

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего
образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к практическим работам

по дисциплине «**Встраиваемые программируемые системы**»
для студентов направления подготовки

11.03.04 Электроника и нанoeлектроника
Направленность (профиль)

Интеллектуальные программируемые системы и устройства

Ставрополь 2026 г.

Введение

Целью дисциплины является формирование набора требуемых компетенций будущего бакалавра по направлению подготовки 11.03.04 Электроника и нанoeлектроника.

Задачи дисциплины: сформировать знания, умения и навыки по требуемым компетенциям.

Выполнение практических работ способствует получению навыков, согласно заявленным для дисциплины компетенциям, на базе теоретического материала, изучаемого на лекционных занятиях.

Содержание

Практическая работа 1. Изучение основ работы в интегрированной среде разработки Microsoft Visual Studio и основ программирования на языке Си	5
Практическая работа 2. Изучение основных операторов языка Си	32
Практическая работа 3. Изучение базовых операций с массивами, строками, правил задания и работы с функциями	53
Практическая работа 4. Изучение основ программирования в интегрированной среде разработки Arduino на базе отладочной платформы Arduino	78
Практическая работа 5. Программирование электронных узлов и устройств на базе отладочной платформы Arduino	103
Практическая работа 6. Изучение основ программирования в визуальном графическом редакторе STM32CubeMX и интегрированной среде разработки Keil uVision на базе отладочной платформы STM32	114
Практическая работа 7. Программирование электронных узлов и устройств на базе отладочной платформы STM32	147
Практическая работа 8. Изучение основ моделирования в многоязычной среде моделирования HDL – ModelSim	157
Практическая работа 9. Изучение основ моделирования в среде программирования Altera Quartus II	187

ПРАКТИЧЕСКАЯ РАБОТА 1

Тема: Изучение основ работы в интегрированной среде разработки Microsoft Visual Studio и основ программирования на языке Си.

Цель работы: закрепление знаний, полученных на лекциях по разделу 4 (лекция 7). В ходе выполнения лабораторной работы необходимо:

- создать консольный проект;
- написать программу вычисления арифметических операций;
- написать программу работы с директивами препроцессора;
- написать программу работы с функциями ввода/вывода.

Формируемые компетенции: ОПК-9.

Теоретическая часть

Си (англ. C) – компилируемый статически типизированный язык программирования общего назначения, разработанный в 1969 – 1973 годах сотрудником Bell Labs Деннисом Ритчи как развитие языка Би.

Си – это универсальный язык программирования с компактным способом записи выражений, современными механизмами управления структурами данных и богатым набором операторов. Си не является ни языком «очень высокого уровня», ни «большим» языком, не рассчитан он и на какую-то конкретную область применения. Однако благодаря широким возможностям и универсальности для решения многих задач он удобнее и эффективнее, чем предположительно более мощные языки.

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом

кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Язык программирования Си оказал существенное влияние на развитие индустрии программного обеспечения, а его синтаксис стал основой для таких языков программирования, как C++, C#, Java и Objective-C. В настоящее время язык перенесен на множество платформ, на этом языке разрабатывается программное обеспечение для различных электронных устройств на основе микроконтроллеров и микропроцессоров. Именно поэтому важным является изучение и получение навыков программирования на языке Си.

Оборудование и материалы

Лабораторная работа выполняется с использованием Microsoft Visual Studio.

Указания по технике безопасности

Перед проведением лабораторной работы следует изучить инструкцию по технике безопасности при работе на компьютере. Также следует соблюдать правила и требования, установленные для данного компьютерного класса.

Задания

1 Задание на подготовку к выполнению работы.

1.1 После установки данного ПО на рабочем столе (или в меню пуск) появится иконка для запуска с названием «Microsoft Visual Studio». При создании приложения в Visual Studio необходимо сначала создать проект и решение.

Для создания консольного приложения необходимо в строке меню выбрать **Файл, Создать, Проект** (рисунок 1.1).

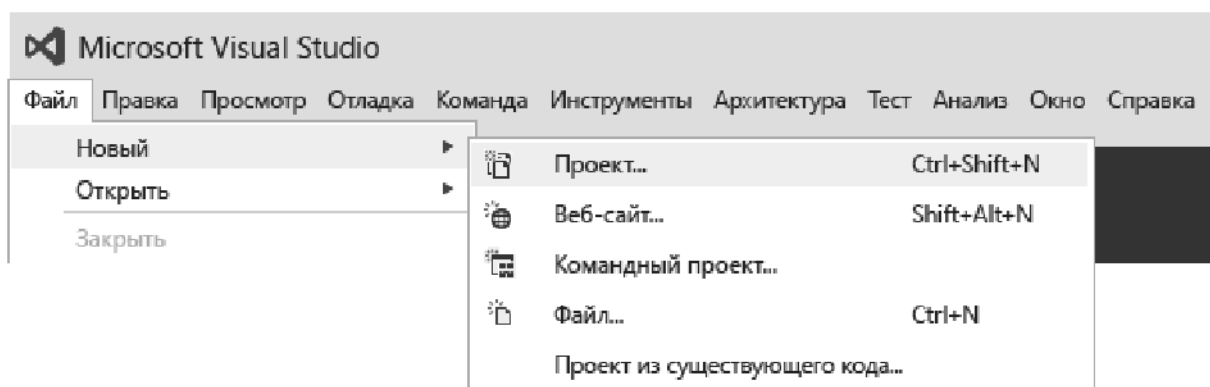


Рисунок 1.1

Далее необходимо в категории **Visual C++** выбрать шаблон **Консольное приложение Win32** и дать название проекту (рисунок 1.2). Также необходимо указать путь сохранения файла проекта. Проекты можно называть Lab1.1, Lab2.2 и т. д. по номеру выполняемой лабораторной работы и номеру задания в ней.

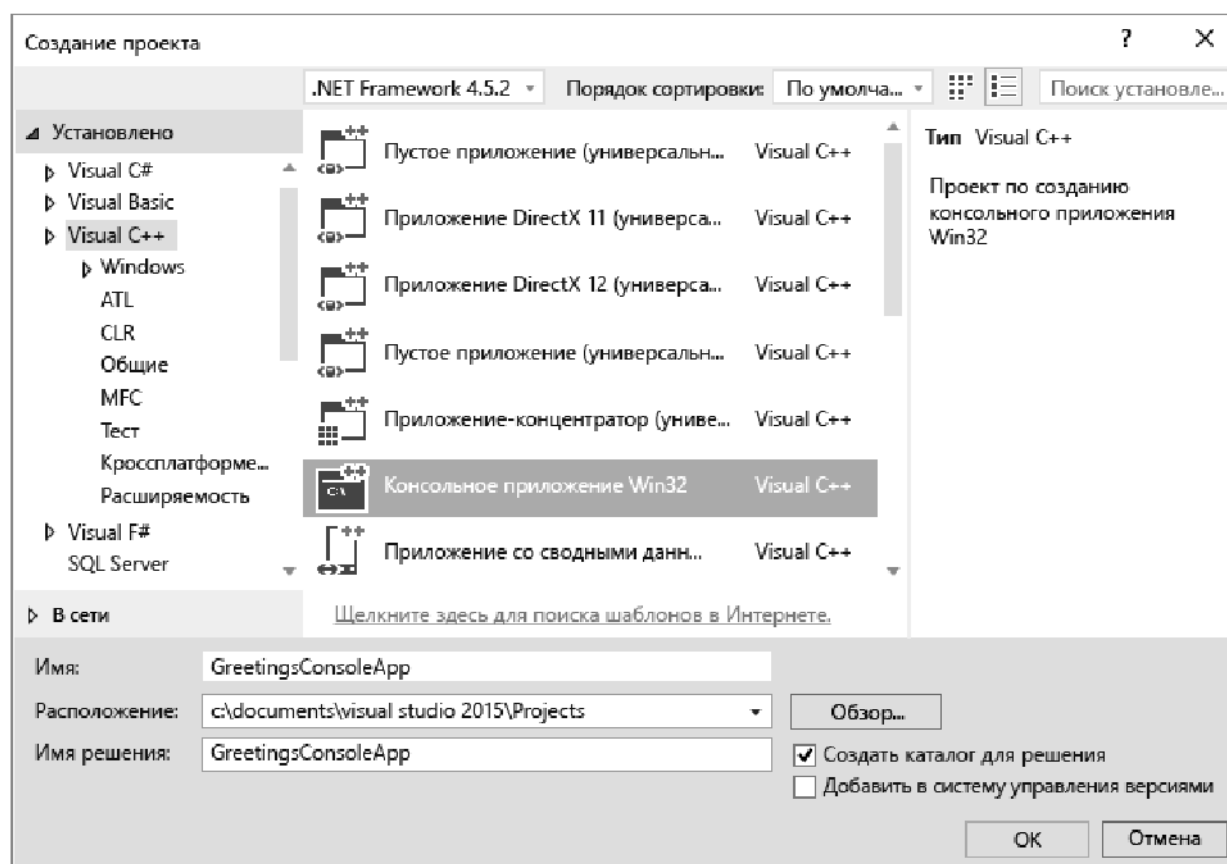


Рисунок 1.2

Когда появится мастер приложений Win32, нажать кнопку **Готово** (рисунок 1.3).

Созданный проект и решение с основными файлами для консольного приложения Win32 создадутся и автоматически загрузятся в **Обозреватель решений**. Файл *.cpp откроется в редакторе кода. В **Обозревателе решений** отображаются необходимые элементы (рисунок 1.4).

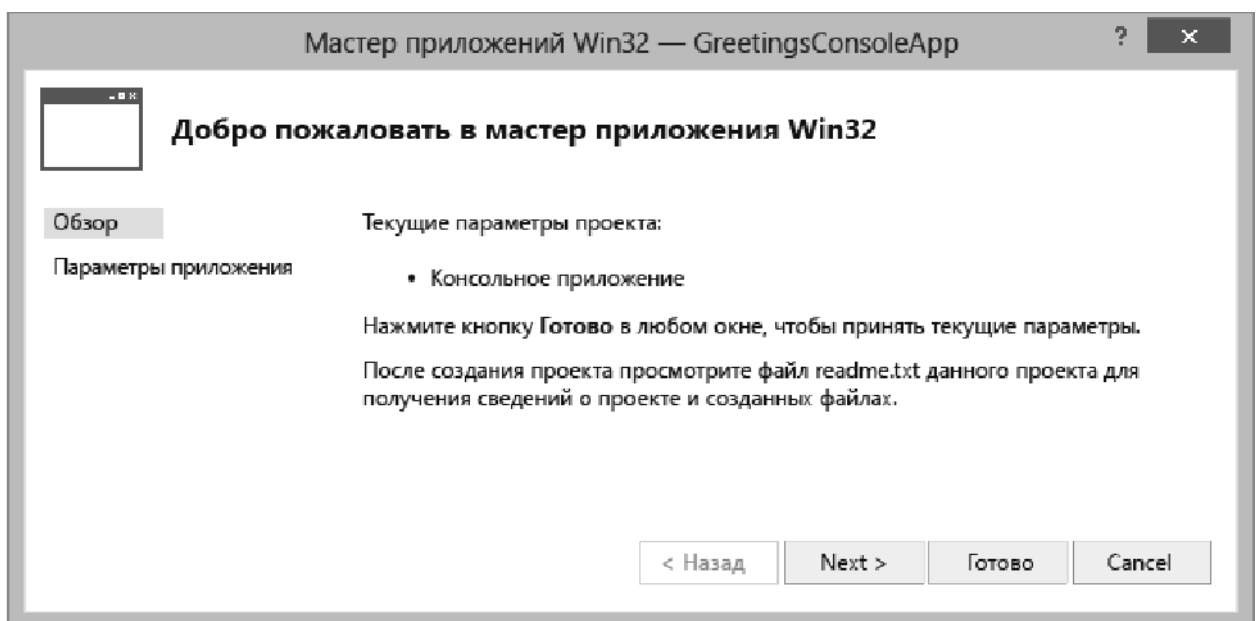


Рисунок 1.3

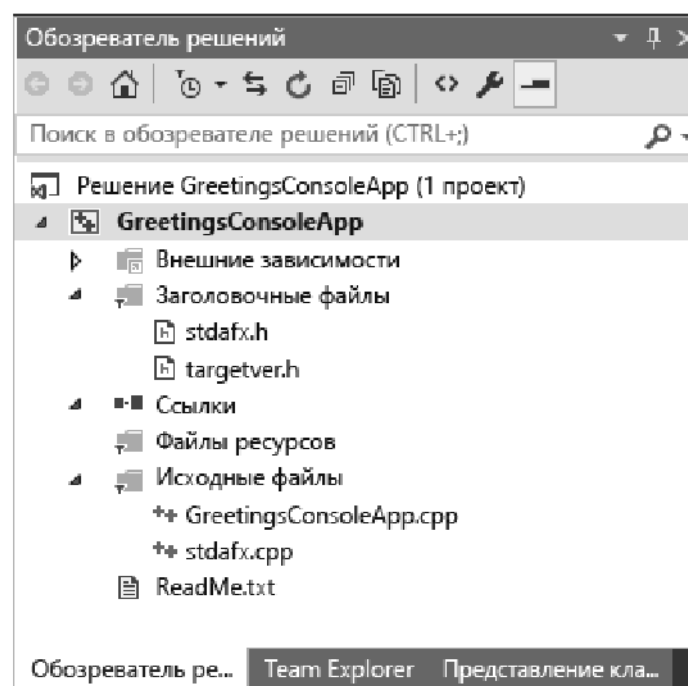


Рисунок 1.4

Слева в окне отображен список файлов проекта, из которых интерес представляет лишь созданный файл *.cpp, т. к. в нем пишется непосредственно программа на языке Си. Справа открыт файл *.cpp, в котором используется стандартный заголовочный файл проекта stdafx.h и

главная функция программы `_tmain()` (аналог функции `main()`). Вся логика программы должна заключаться внутри функции `_tmain()`.

Для того чтобы сохранить проект на внешнем носителе, например, USB Flash, необходимо открыть ранее указанную папку при создании проекта и в ней скопировать каталог с названием проекта, в данном случае – это каталог с именем лабораторной работы и примера.

Чтобы загрузить ранее созданный проект в среду Visual Studio достаточно открыть пункт меню `File->Open->Project/Solution` и в диалоговом окне указать файл проекта, находящегося в папке проекта, в данном случае – это папка с именем лабораторной работы и/или примера.

Для добавления кода в приложение необходимо добавить код для отображения слова «Hello» в окне консоли создать тестовый файл `GreetingsConsoleApp.cpp` и в нем ввести пустую строку перед строкой `return 0;`, а затем ввести в нее следующий код (рисунок 1.5):

```
cout << "Hello\n";
```

Рисунок 1.5

Красная волнистая линия появится под `cout`. При наведении на нее отобразится сообщение об ошибке (рисунок 1.6).

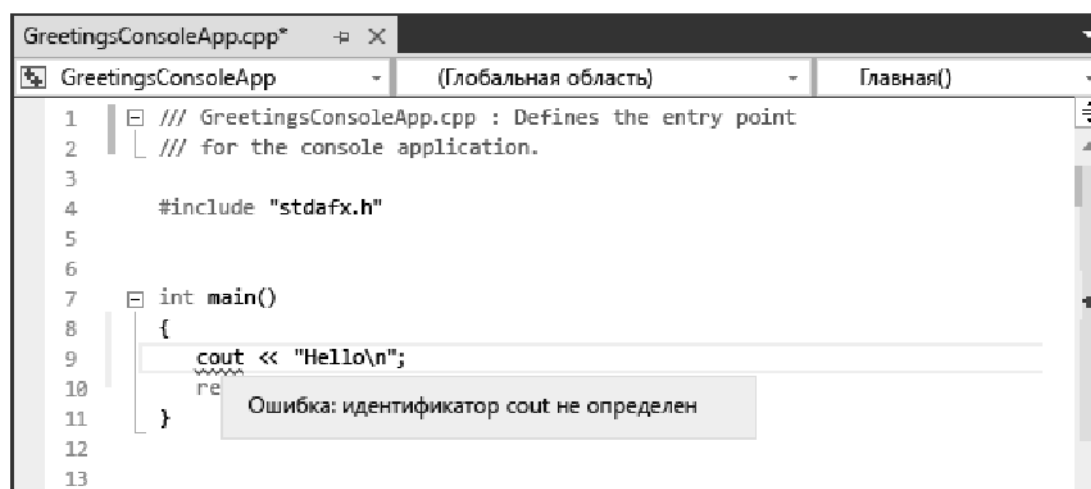


Рисунок 1.6

Сообщение об ошибке также отобразится в окне **Список ошибок**. Можно отобразить это окно, выбрав в строке меню **Вид, Список ошибок**. `cout` включается в файл заголовка `<iostream>`.

Для включения заголовка `iostream` введите следующий код после `#include "stdafx.h"` (рисунок 1.7):

```
#include <iostream> using namespace std;
```

Рисунок 1.7

После вставки этого кода появилось окно, предлагающее рекомендации для символов, которые были введены. Это поле является частью технологии C++ IntelliSense, обеспечивающей подсказки по коду, в том числе отображение членов класса или интерфейса и сведения о параметрах. Кроме того, можно использовать фрагменты кода в виде определенных блоков кода.

Красная волнистая линия под `cout` исчезнет после исправления ошибки. Необходимо сохранить изменения в файле (рисунок 1.8).

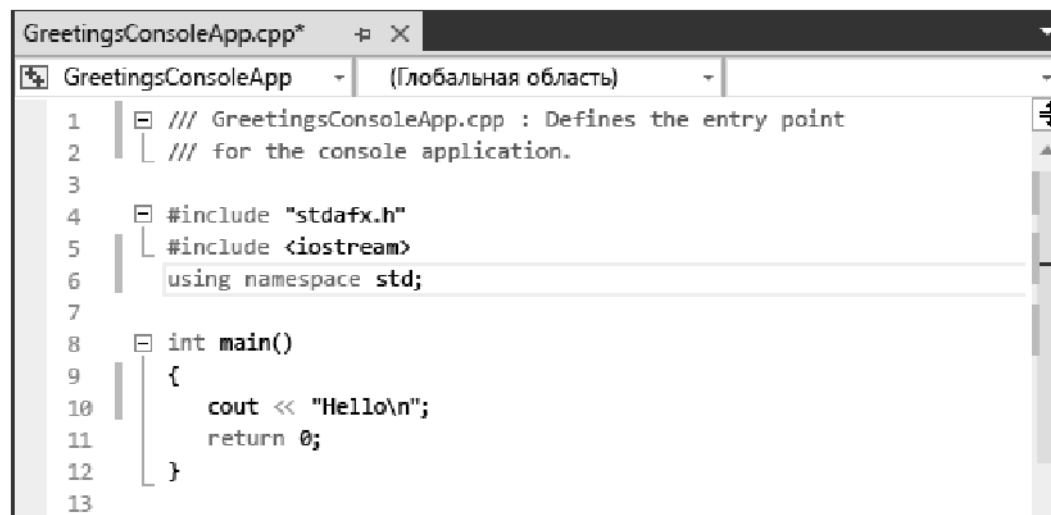


Рисунок 1.8

С помощью отладки GreetingsConsoleApp необходимо проверить, отображается ли слово Hello в окне консоли, для этого нужно запустить отладчик (рисунок 1.9).

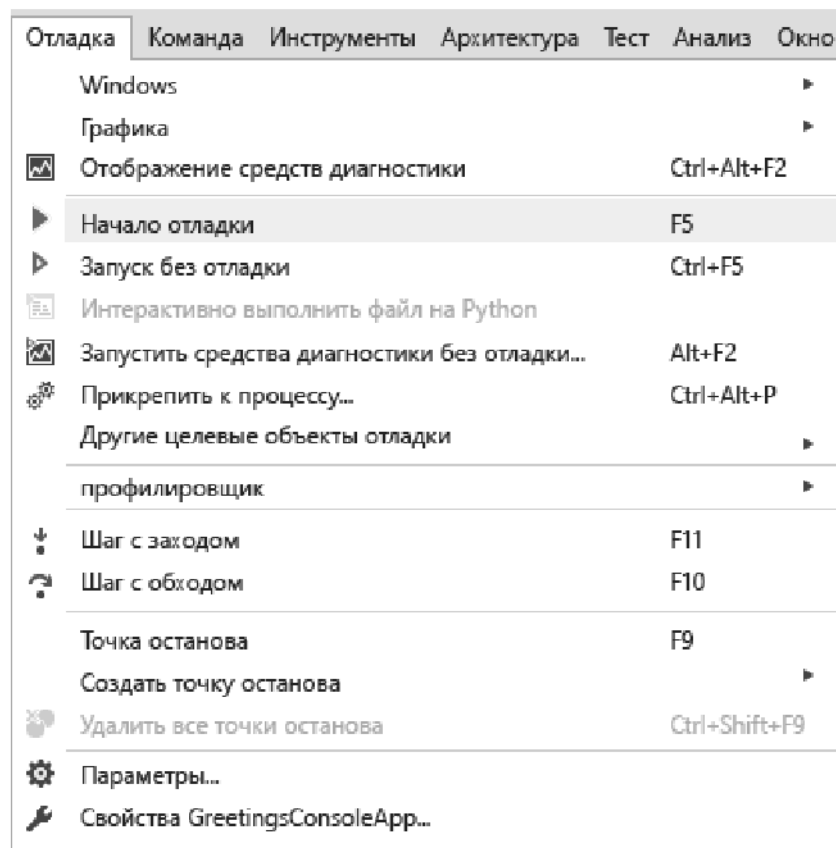


Рисунок 1.9

Отладчик запустится и выполнит код. Окно консоли (отдельное окно, подобное командной строке) отображается в течение нескольких секунд, но при остановке отладчика быстро закрывается. Чтобы просмотреть текст, необходимо установить точку останова выполнения программы. Добавьте точку останова из меню в строке `return 0;`. Для установки точки останова можно также просто щелкнуть область слева (рисунок 1.10).

Рядом со строкой кода в крайнем левом поле окна редактора появится красный кружок. Нажмите клавишу F5, чтобы начать отладку. Запускается отладчик, и появляется окно консоли, в котором выводится слово **Hello**. Для останова процесса отладки нажмите SHIFT + F5.

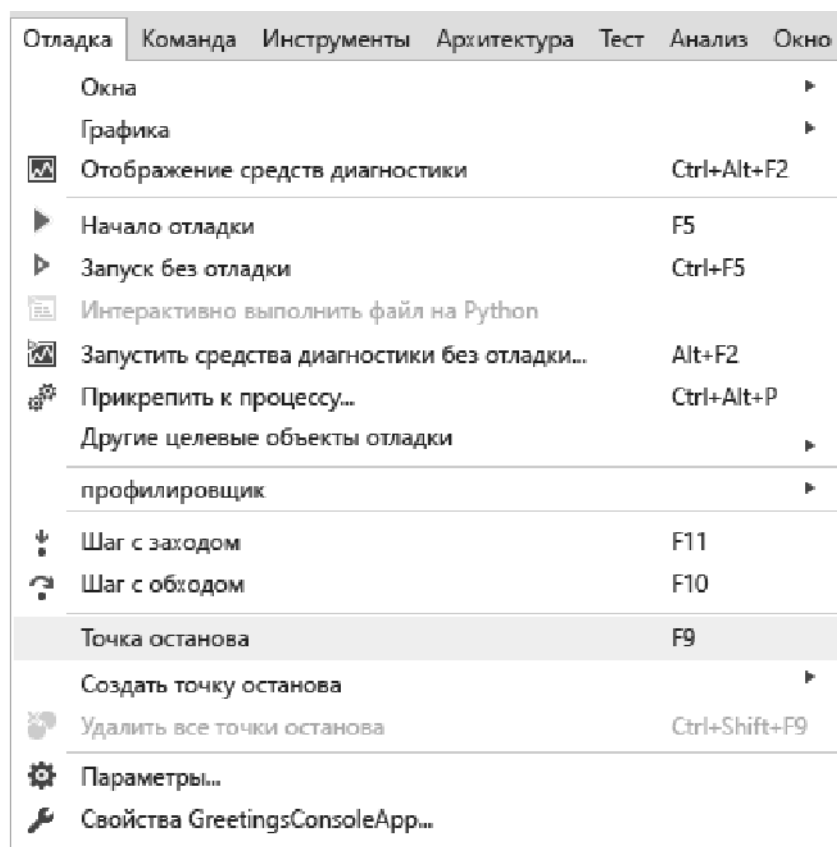


Рисунок 1.10

На завершающем этапе, после проверки работоспособности проекта, производится сборка окончательной версии приложения:

- сначала необходимо провести очистку файлов решения: из строки меню удалите промежуточные файлы и выходные файлы, созданные во время предыдущих сборок (рисунок 1.11);

- измените конфигурацию сборки для GreetingsConsoleApp с **Отладка** на **Выпуск** (рисунок 1.12);

- постройте решение (рисунок 1.13).

Таким образом можно создавать проекты для каждой лабораторной работы и задачи, сохранять, загружать и выполнять их.

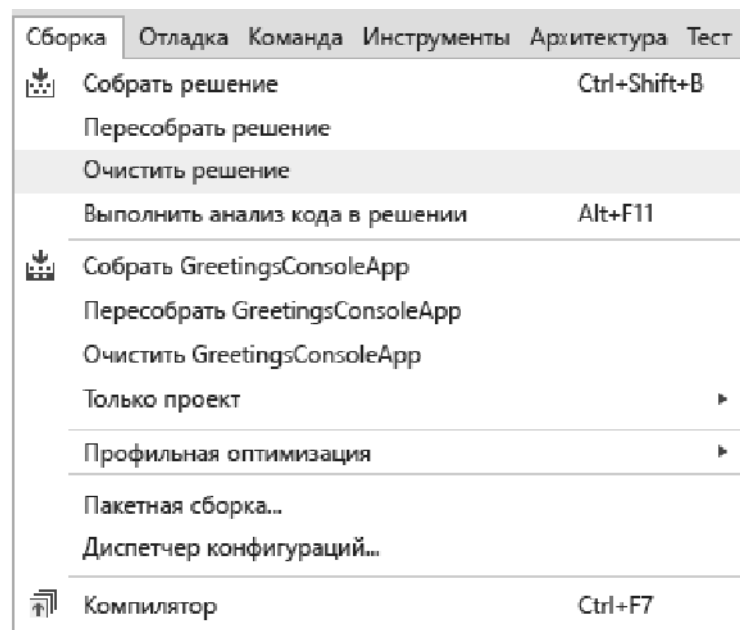


Рисунок 1.11

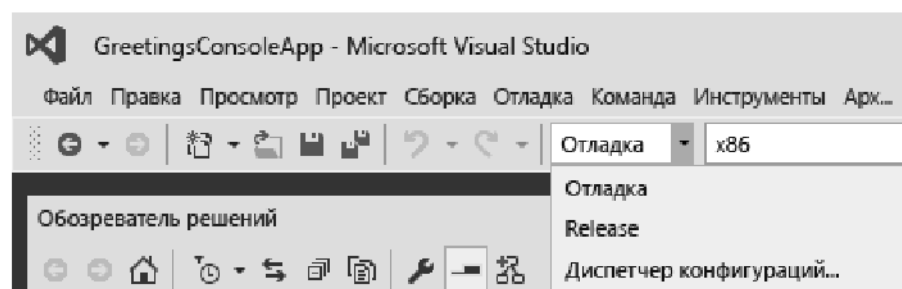


Рисунок 1.12

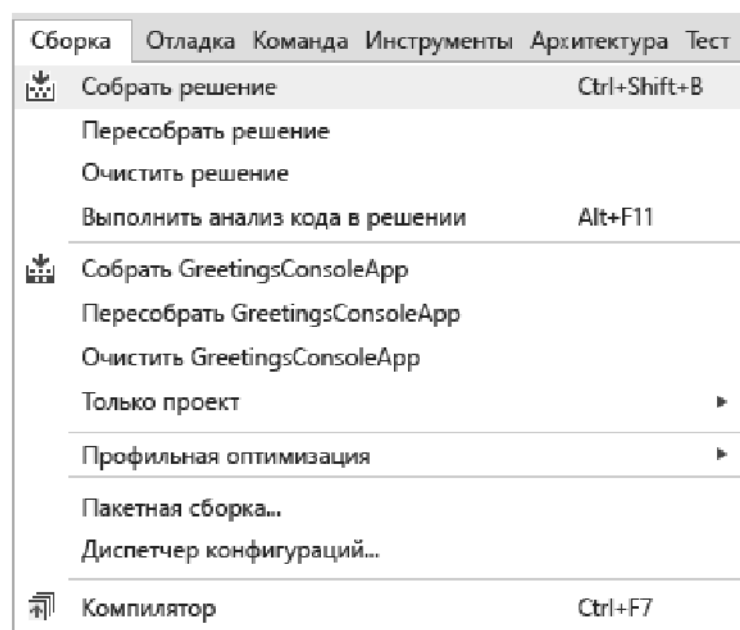


Рисунок 1.13

1.2 Представление данных в языке Си.

Для того чтобы иметь возможность работать с тем или иным типом данных необходимо задать переменную соответствующего типа. Это осуществляется с использованием следующего синтаксиса:

<тип переменной> <имя_переменной>;

например, строка объявляет целочисленную переменную с именем arg:

```
int arg;
```

Основные базовые типы данных приведены в таблице 1.1.

Таблица 1.1

Тип	Описание
int	Целочисленный (обычно 32 бита)
short	Целочисленный (обычно 16 бит)
float	Символьный тип 8 бит
double	Вещественный тип 32 бита
char	Вещественный тип 64 бита

При выборе имени переменной целесообразно использовать осмысленные имена. При определении имени можно использовать как верхний, так и нижний регистры букв латинского алфавита. Причем первым символом обязательно должна быть буква или символ подчеркивания ‘_’.

Рассмотрим несколько примеров:

Правильные имена	Неправильные имена
arg	&arg
cnt	\$cnt
bottom_x	bottom-x
Arg	2Arg
don_t	don't

В приведенных примерах переменные `arg` и `Arg` считаются разными, т. к. язык Си при объявлении переменных различает большой и малый регистры.

В отличие от многих языков программирования высокого уровня, в языке Си переменные могут объявляться в любом месте текста программы.

1.3 Выполнение арифметических операций.

В языке Си довольно просто реализуются элементарные математические операции: сложения, вычитания, умножения и деления. Допустим, что в программе заданы две переменные:

```
int a, b;
```

с начальными значениями

```
a=4;
```

```
b=8;
```

тогда операции сложения, вычитания, умножения и деления будут выглядеть:

```
int c;
```

```
c = a+b; //сложение двух переменных
```

```
c = a-b; //вычитание
```

```
c = a*b; //умножение
```

```
c = a/b; //деление
```

Представленные операции можно выполнять не только с переменными, но и с конкретными числами, например:

```
c = 10+5;
```

```
c = 8*4;
```

```
float d;
```

```
d = 7/2;
```

Результатом первых двух арифметических операций будут числа 15 и 32 соответственно, но при выполнении операции деления в переменную `d` будет записано число 3, а не 3,5. Это связано с тем, что число 7 в языке C++

будет интерпретироваться как целочисленная величина, которая не может содержать дробной части. Поэтому полученная дробная часть 0,5 будет отброшена. Для реализации корректного деления одного числа на другое следует использовать такую запись:

```
d = 7.0/2;
```

или

```
d = (float )7/2;
```

В первом случае вещественное число делится на два и результат (вещественный) присваивается вещественной переменной `d`. Во втором варианте выполняется приведение типов: целое число 7 приводится к вещественному типу `float`, а затем делится на 2. Второй вариант удобен, когда выполняется деление одной целочисленной переменной на другую:

```
int a,b;
```

```
a = 7; b = 2;
```

```
d = a/b;
```

В результате значение `d` будет равно 3, но если записать

```
d = (float )a/b;
```

то получим значение 3,5. Здесь следует также отметить, что если переменная `d` является целочисленной, то результат деления всегда будет записан с отброшенной дробной частью.

В заключение рассмотрения работы с арифметическими операциями отметим, что приоритет операций умножения и деления выше приоритета операций сложения и вычитания. Это означает, что сначала выполняются операции умножения и деления и только затем операции сложения и вычитания. Следующий пример демонстрирует приоритет арифметических операций:

```
double n=2, SCALE = 1.2;
```

```
double arg = 25.0 + 60.0*n/SCALE;
```

В приведенном примере сначала будет выполнена операция умножения, затем деления и, наконец, сложения. То есть порядок вычисления соответствует математическим правилам. Для того чтобы изменить порядок вычисления (поменять приоритеты) используются круглые скобки как показано ниже

```
double arg = (25.0 + 60.0)*n/SCALE;
```

Здесь сначала выполняется операция сложения и только затем операции умножения и деления.

Для простоты программирования в языке С реализованы компактные операторы инкремента и декремента, т.е. увеличения и уменьшения значения переменной на 1 соответственно. Данные операторы могут быть записаны в виде:

```
i++; // операция инкремента
```

```
++i; // операция инкремента
```

```
i--; // операция декремента
```

```
--i; // операция декремента
```

Разницу между первой и второй формами записи данных операторов можно продемонстрировать на следующем примере:

```
int i=10, j=10;
```

```
int a = i++; //значение a = 10; i = 11;
```

```
int b = ++j; //значение b = 11; j = 11;
```

Из полученных результатов видно, что если оператор инкремента стоит после имени переменной, то сначала выполняется операция присваивания и только затем операция инкремента. Во втором случае наоборот, операция инкремента реализуется до присвоения результата другой переменной. Поэтому значение a = 10, а значение b = 11.

1.4 Директивы препроцессора.

Почти все программы на языке Си используют специальные команды для компилятора, которые называются директивами. В общем случае

директива – это указание компилятору языка Си выполнить то или иное действие в момент компиляции программы. Существует строго определенный набор возможных директив, который включает в себя следующие определения:

`#define, #elif, #else, #endif, #if, #ifdef, #ifndef, #include, #undef.`

Директива `#define` используется для задания констант, ключевых слов, операторов и выражений, используемых в программе. Общий синтаксис данной директивы имеет следующий вид:

`#define <идентификатор> <текст>`

или

`#define <идентификатор> (<список параметров>) <текст>`

Следует заметить, что символ `;` после директив не ставится.

Рассмотрим примеры вариантов использования директивы `#define`.

Листинг 1. Примеры использования директивы `#define`.

```
#include <stdio.h>

#define TWO 2

#define FOUR TWO*TWO

#define PX printf("X равно %d.\n", x)

#define FMT «X равно %d.\n»

#define SQUARE(X) X*X

int main()
{
    int x = TWO;

    PX;

    x = FOUR;

    printf(FMT, x);

    x = SQUARE(3);

    PX;
```

```
return 0;  
}
```

После выполнения этой программы на мониторе появится три строки:

X равно 2.

X равно 4.

X равно 9.

Директива `#undef` отменяет определение, введенное ранее директивой `#define`. Предположим, что на каком-либо участке программы нужно отменить определение константы `FOUR`. Это достигается следующей командой:

```
#undef FOUR
```

Интересной особенностью данной директивы является возможность переопределения значения ранее введенной константы. Действительно, повторное использование директивы `#define` для ранее введенной константы `FOUR` невозможно, т.к. это приведет к сообщению об ошибке в момент компиляции программы. Но если отменить определение константы `FOUR` с помощью директивы `#undef`, то появляется возможность повторного использования директивы `#define` для константы `FOUR`.

Для того чтобы иметь возможность выполнять условную компиляцию, используется группа директив `#if`, `#ifdef`, `#ifndef`, `#elif`, `#else` и `#endif`. Приведенная ниже программа выполняет подключение библиотек в зависимости от установленных констант.

```
#if defined(GRAPH)  
#include <graphics.h> //подключение графической библиотеки  
#elif defined(TEXT)  
#include <conio.h>    //подключение текстовой библиотеки  
#else  
#include <io.h>       //подключение библиотеки ввода-вывода
```

```
#endif
```

Данная программа работает следующим образом. Если ранее была задана константа с именем GRAPH через директиву #define, то будет подключена графическая библиотека с помощью директивы #include. Если идентификатор GRAPH не определен, но имеется определение TEXT, то будет использоваться библиотека текстового ввода/вывода. Иначе, при отсутствии каких-либо определений, подключается библиотека ввода/вывода. Вместо словосочетания #if defined часто используют сокращенные обозначения #ifdef и #ifndef и выше приведенную программу можно переписать в виде:

```
#ifdef GRAPH
#include <graphics.h> //подключение графической библиотеки
#endif
#ifdef TEXT
#include <conio.h> //подключение текстовой библиотеки
#else
#include <io.h> //подключение библиотеки ввода-вывода
#endif
```

Отличие директивы #if от директив #ifdef и #ifndef заключается в возможности проверки более разнообразных условий, а не только существует или нет какие-либо константы. Например, с помощью директивы #if можно проводить такую проверку:

```
#if SIZE == 1
#include <math.h> // подключение математической библиотеки
#elif SIZE > 1
#include <array.h> // подключение библиотеки обработки
// массивов
#endif
```

В приведенном примере подключается либо математическая библиотека, либо библиотека обработки массивов, в зависимости от значения

константы SIZE.

Используемая в приведенных примерах директива `#include` позволяет добавлять в программу ранее написанные программы и сохраненные в виде файлов. Например, строка

```
#include <stdio.h>
```

указывает препроцессору добавить содержимое файла `stdio.h` вместо приведенной строки. Это дает большую гибкость, легкость программирования и наглядность создаваемого текста программы.

1.5 Функции ввода/вывода `printf()` и `scanf()`.

Функция `printf()` позволяет выводить информацию на экран при программировании в консольном режиме. Данная функция определена в библиотеке `stdio.h` и имеет следующий синтаксис:

```
int printf( const char *format [, argument]... );
```

Здесь первый аргумент `*format` определяет строку, которая выводится на экран и может содержать специальные управляющие символы для вывода переменных. Затем, следует список необязательных аргументов, которые поясняются ниже. Функция возвращает либо число отображенных символов, либо отрицательное число в случае своей некорректной работы.

В самой простой реализации функция `printf()` просто выводит заданную строку на экран монитора:

```
printf("Привет мир.");
```

Однако с ее помощью можно выводить переменные разного типа: начиная с числовых и заканчивая строковыми. Для выполнения этой операции используются специальные управляющие символы, которые называются спецификаторами и которые начинаются с символа `%`. Следующий пример демонстрирует вывод целочисленной переменной `num` на экран монитора с помощью функции `printf()`:

```
int num;  
num = 5;
```

```
printf(“%d”, num);
```

В первых двух строках данной программы задается переменная с именем `num` типа `int`. В третьей строке выполняется вывод переменной на экран. Работа функции `printf()` выглядит следующим образом. Сначала функция анализирует строку, которую необходимо вывести на экран. В данном случае это «%d». Если в этой строке встречается спецификатор, то на его место записывается значение переменной, которая является вторым аргументом функции `printf()`. В результате, вместо исходной строки «%d» на экране появится строка «5», т. е. будет выведено число 5.

Следует отметить, что спецификатор «%d» выводит только целочисленные типы переменных, например `int`. Для вывода других типов следует использовать другие спецификаторы. Ниже перечислены основные виды спецификаторов:

%c – одиночный символ

%d – десятичное целое число со знаком

%f – число с плавающей точкой (десятичное представление)

%s – строка символов (для строковых переменных)

%u – десятичное целое без знака

%% – печать знака процента

С помощью функции `printf()` можно выводить сразу несколько переменных. Для этого используется следующая конструкция:

```
int num_i;  
float num_f;  
num_i = 5;  
num_f = 10.5;  
printf(“num_i = %d, num_f = %f”, num_i, num_f);
```

Результат выполнения программы будет выглядеть так:

```
num_i = 5, num_f = 10.5
```

Кроме спецификаторов в функции `printf()` используются управляющие символы, такие как перевод строки `\n`, табуляции `\t` и др. Например, если в ранее рассмотренном примере необходимо вывести значения переменных не в строчку, а в столбик, то необходимо переписать функцию `printf()` следующим образом:

```
printf("num_i = %d,\n num_f = %f", num_i, num_f);
```

Аналогично используется и символ табуляции.

Для ввода информации с клавиатуры удобно использовать функцию `scanf()` библиотеки `stdio.h`, которая имеет следующий синтаксис:

```
int scanf( const char *format [,argument]... );
```

Здесь, как и для функции `printf()`, переменная `*format` определяет форматную строку для определения типа вводимых данных и может содержать те же спецификаторы что и функция `printf()`. Затем, следует список необязательных аргументов. Работа функции `scanf()` демонстрируется на листинге 2.

Листинг 2. Пример использования функции `scanf()`.

```
#include <stdio.h>

int main()
{
    int age;
    float weight;
    printf("Введите информацию о Вашем возрасте: ");
    scanf("%d", &age);
    printf("Введите информацию о Вашем весе: ");
    scanf("%f", &weight);
    printf("Ваш возраст = %d, Ваш вес = %f", age, weight);
    return 0;
}
```

Основным отличием применения функции `scanf()` от функции `printf()` является знак `&` перед именем переменной, в которую записываются результаты ввода.

Функция `scanf()` может работать сразу с несколькими переменными. Предположим, что необходимо ввести два целых числа с клавиатуры. Формально для этого можно дважды вызвать функцию `scanf()`, однако лучше воспользоваться такой конструкцией:

```
scanf(" %d, %d ", &n, &m);
```

Функция `scanf()` интерпретирует это так, как будто ожидает, что пользователь введет число, затем – запятую, а затем – второе число.

Функция `scanf()` возвращает число успешно считанных элементов. Если операции считывания не происходило, что бывает в том случае, когда вместо ожидаемого цифрового значения вводится какая-либо буква, то возвращаемое значение равно 0.

2 Задание на выполнение работы.

2.1 Создать консольный проект согласно описанию, приведенному в п. 1.1. В качестве выводимой информации использовать собственные фамилию, имя и отчество. Во втором проекте вывести дополнительно дату и место рождения с новой строки. Результаты проектов сохранить в файлах.

2.2 Написать программу вычислений в соответствии с заданным преподавателем вариантом (числовые параметры задаются самостоятельно). Исходные данные приведены в таблице 1.2.

2.3 Написать программу работы с директивами препроцессора в соответствии с заданным преподавателем вариантом. Исходные данные приведены в таблице 1.3.

2.4 Написать программу с использованием функций `printf()` и `scanf()` в соответствии с заданным преподавателем вариантом. Исходные данные приведены в таблице 1.3.

Таблица 1.2

Вариант	Тип переменных	Вычислить
1	Вещественный Целочисленный	Периметр прямоугольника Площадь круга
2	Вещественный Целочисленный	Площадь прямоугольника Длину окружности
3	Вещественный Целочисленный	Площадь треугольника Объем параллелепипеда
4	Вещественный Целочисленный	Высоту параллелепипеда Евклидовое расстояние между двумя точками: $d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$
5	Вещественный Целочисленный	$(a+b)^3$ $(a+b)^2$
6	Вещественный Целочисленный	$\frac{1}{a} + \frac{2}{a} + \dots + \frac{5}{a}$ $f(x) = kx + b$, при $x = 1, 2, \dots, 5$
7	Вещественный Целочисленный	$(a-b)^2$ Площадь круга
8	Вещественный Целочисленный	Периметр прямоугольника Площадь параллелограмма
9	Вещественный Целочисленный	10 % от числа 456 $a_1b_1 + a_2b_2 + \dots + a_5b_5$
10	Вещественный Целочисленный	Длину окружности $f(x) = x^2 + b$, при $x = 1, 2, \dots, 5$
11	Вещественный Целочисленный	Периметр треугольника $(a+b)^2 - (a-b)$
12	Вещественный Целочисленный	Периметр трапеции $f(x) = kx - b^2$, при $x = 1, 2, \dots, 5$
13	Вещественный Целочисленный	Неизвестный угол равнобедренного треугольника по двум сторонам и углу между ними Высоту прямоугольного треугольника по известным сторонам
14	Вещественный Целочисленный	Высоту прямоугольного треугольника по составным отрезкам гипотенузы $a_1b_1 + (a_2b_2)^2 + \dots + (a_5b_5)^5$

15	Вещественный Целочисленный	$f(x) = \frac{kx^2 - b}{2}$, при $x = 1, 2, \dots, 5$ Катет прямоугольного треугольника
16	Вещественный Целочисленный	$(a+b)^2 * (a-b)$ Объем прямоугольника
17	Вещественный Целочисленный	$(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2$ Объем правильного тетраэдра
18	Вещественный Целочисленный	Полную площадь поверхности цилиндра $f(x) = \frac{kx^2 - b}{a^2}$, при $x = 1, 2, \dots, 4$
19	Вещественный Целочисленный	Объем конуса $(a+b)^3 + (a+b)^2 + (a+b)$
20	Вещественный Целочисленный	Периметр эллипса $S = \frac{\pi ab}{2} - \frac{b}{a}$

Таблица 1.3

Вариант	Программирование директив препроцессора	Программирование функций printf() и scanf()
1	Программа вычисления $a + b$ с использованием директивы #define	Ввести два вещественных значения и вывести их произведение на экран монитора
2	С помощью директив #if, #else, #elif осуществить выбор строк программы для вычисления либо $2(a+b)$, либо ab	Ввести два целочисленных значения и вывести их частное на экран монитора
3	Задать константы M и N и вычислить $\frac{(aM + bN)}{MN}$	Ввести два вещественных значения и вывести их на экран с точностью до сотых
4	С помощью директивы #define вычислить x^2 , при $x = 1, 2, \dots, 5$	Ввести два целочисленных значения и вывести их разность на экран монитора
5	Задать константы $M1, M2, \dots, M5$ и вычислить $M1 + 2 * M2 + 3 * M3 + 4 * M4 + 5 * M5$	Ввести целочисленное и вещественное значения и вывести их сумму на экран монитора
6	С помощью директивы #define вычислить	Ввести два вещественных значения и вывести их на экран с

	$kx + b$, при $x = 1, 2, \dots, 5$	точностью до тысячных
7	С помощью директив #if, #else, #elif осуществлять выбор строк программы для вычисления либо $(a + b)^2$, либо $(a - b)^2$	Ввести ширину и высоту прямоугольника, вычислить его периметр и вывести результат на экран монитора
8	С помощью директивы #define вычислить x^3 , при $x = -2, -1, \dots, 2$	Ввести ширину и высоту прямоугольника, вычислить его площадь и вывести результат на экран монитора
9	Задать константы M и N и вычислить $(M + N)^2 / 2$	Ввести длину основания и высоту равнобедренного прямоугольника, вычислить его площадь и вывести результат на экран монитора
10	С помощью директивы #define вычислить $(x + y)^2$, при $x, y = 1, 2, \dots, 5$	Ввести длину основания и высоту равнобедренного прямоугольника, вычислить его периметр и вывести результат на экран монитора
11	С помощью директив #if, #else, #elif осуществить выбор строк программы для вычисления либо x^2 , при $x = 1, 2, \dots, 5$, либо $(x + y)^2$, при $x, y = 1, 2, \dots, 5$	Ввести значение периметра квадрата, рассчитать длину его диагонали и вывести результат на экран монитора
12	Задать константы M , N и P и вычислить $(M + N)^3 / (2 * P)$	Ввести значение периметра квадрата, рассчитать его площадь и вывести результат на экран монитора
13	С помощью директив #if, #else, #elif осуществлять выбор строк программы для вычисления либо $kx^2 + b$, при $x = 1, 2, \dots, 5$, либо $kx^3 + 2b$, при $x = 1, 2, \dots, 5$	Ввести значения диагоналей ромба, рассчитать значение его стороны и вывести результат на экран монитора
14	Задать константы M и S и вычислить $\frac{(aM + b)}{M(S + 2M)}$	Ввести длину стороны ромба и одной диагонали, рассчитать значение второй диагонали и вывести результат на экран монитора
15	Задать константы $M1, M2, S1, S2$ и вычислить $M1 + 2 * S2 + 3 * M2 + 5 * S1$	Ввести значение двух диагоналей ромба, рассчитать его площадь и вывести результат на экран монитора
16	С помощью директив #if, #else,	Ввести значение радиуса шара,

	<code>#elif</code> осуществить выбор строк программы для вычисления либо $(a+b)^3 - 0,5a$, либо $(0,5a + 0,2b)^{-2} - 10$	рассчитать его объем и вывести результат на экран монитора
17	Задать константы R , Q и S и вычислить $\frac{\sqrt{aR+b}}{Q^2(3S+2R)}$	Ввести значение радиуса сферы и высоты сегмента сферы, рассчитать объем сегмента сферы с заданной высотой и вывести результат на экран монитора
18	Программа вычисления $(a+b)^3 - 0,5a$ с использованием директивы <code>#define</code>	Ввести значения длин сторон трапеции, рассчитать длины диагоналей и вывести результат на экран монитора
19	Программа вычисления $(x+y)^4$, при $x, y = 1, 2, 3$ с использованием директивы <code>#define</code>	Ввести значения радиуса и длины образующей прямого кругового конуса, рассчитать общую площадь поверхности и вывести результат на экран монитора
20	С помощью директивы <code>#define</code> вычислить $(2x + \sqrt{y})^3$, при $x, y = 1, 2, \dots, 6$	Ввести значения радиуса основы и высоты цилиндра, рассчитать объем цилиндра и вывести результат на экран монитора

3 Задание на обработку данных.

3.1 Сделать выводы о полученных результатах работы написанных программ.

Содержание отчета

Отчет по лабораторной работе выполняется на листах формат А4 согласно требований ГОСТ и ЕСКД и должен содержать следующие пункты:

- а) титульный лист с названием работы, дисциплины, кода группы, ф.и.о. исполнителя и преподавателя;
- б) цель и содержание работы;
- в) описание ПО, необходимого для выполнения работы;

- г) написанные программы приведенных заданий;
- д) результаты исследований работы программ;
- е) выводы по результатам работы.

Контрольные вопросы

1. Приведите примеры правильных имен переменных.
2. Чему будет равна переменная *c* в строке программы `float c = 7 / 2`?
3. Приведите примеры неправильных имен переменных.
4. Как записывается оператор умножения в языке Си?
5. Как изменится значение переменной *i* в строчке программы `i = i + 1`?
6. Что такое операция декремента?
7. Приведите пример использования функции `printf()` для вывода значений двух целочисленных переменных на экран.
8. Запишите функцию `scanf()` для ввода символа с клавиатуры.
9. Запишите директиву `#define` для задания константы с именем `LENGTH` равной 10.
10. Приведите пример макроса, позволяющего возводить число в квадрат.
11. С помощью каких директив можно выполнять условную компиляцию программы?
12. Запишите функцию `printf()` для вывода вещественной переменной с точностью до сотых.

Список литературы, рекомендованный к использованию по данной теме

Основная учебная литература

Дополнительная литература

ПРАКТИЧЕСКАЯ РАБОТА 2

Тема: Изучение основных операторов языка Си.

Цель работы: закрепление знаний, полученных на лекциях по разделу 4 (лекция 7). В ходе выполнения лабораторной работы необходимо:

- изучить особенности использования условных операторов;
- изучить особенности использования операторов циклов;
- написать программу с использованием условных операторов;
- написать программу с использованием операторов циклов.

Формируемые компетенции: ОПК-9.

Теоретическая часть

Способность управлять программным потоком позволяет делать выборочное выполнение отдельных участков кода, – это весьма ценная особенность программирования. Условный оператор позволяет выполнять или не выполнять определенные участки кода, в зависимости от того является ли истинным или ложным условие этого оператора. Одно из самых важных назначений условного оператора это то, что он позволяет программе совершить действие на выбор, в зависимости от того, какие данные ввел пользователь.

Без условного оператора программа бы срабатывала раз за разом одинаково, независимо от того, какие входные данные поступали от пользователя. Если же использовать операторы выбора, то результат работы программы может быть намного интереснее, так как он будет зависеть напрямую от входных данных пользователя.

Таким образом, условные операторы позволяют строить разветвляющийся алгоритм. Разветвляющимся называется такой алгоритм, в котором выбирается один из нескольких возможных вариантов вычислительного процесса. Каждый подобный путь называется ветвью

алгоритма.

Циклом называется блок кода, который для решения задачи требуется повторить несколько раз. Циклы используются для многократного повторения частей программного кода. Возможность повторения определенных фрагментов кода – это одна из основных и в тоже время важных задач, которые приходится решать программисту. Большинство программ используют циклы. То есть в таких задачах необходимо выполнять постоянно операции чтения и записи, и для того чтобы не дублировать один и тот же код на помощь приходят циклы. Циклы достаточно просто объявляются в коде, однако они выполняют сложные задачи, всего лишь простым повторением.

Именно поэтому важным является изучение условных операторов и операторов циклов, получение навыков программирования с их использованием.

Оборудование и материалы

Лабораторная работа выполняется с использованием Microsoft Visual Studio.

Указания по технике безопасности

Перед проведением лабораторной работы следует изучить инструкцию по технике безопасности при работе на компьютере. Также следует соблюдать правила и требования, установленные для данного компьютерного класса.

Задания

1 Задание на подготовку к выполнению работы.

1.1 Условный оператор if.

Чаще всего для проверки условия используется условный оператор if. Условный оператор if может использоваться в форме полной или неполной развилки (рисунок 2.1).

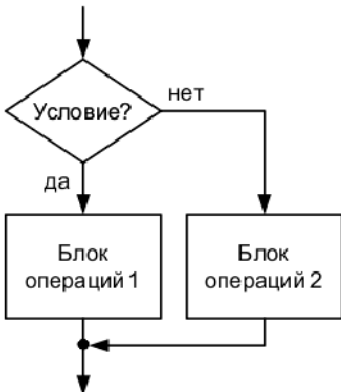
Неполная развилка	Полная развилка
<pre>1 if (Условие) 2 { 3 БлокОпераций1; 4 }</pre>	<pre>1 if (Условие) 2 { 3 БлокОпераций1; 4 } 5 else 6 { 7 БлокОпераций2; 8 }</pre>
	

Рисунок 2.1 – Условный оператор if

Следует отметить, что «Условие» является условным выражением, в котором выполняется проверка некоторого условия. В таблице 2.1 представлены варианты простых логических выражений оператора if.

В случае неполной развилки если «Условие» истинно, то «БлокОпераций1» выполняется, если «Условие» ложно, то «БлокОпераций1» не выполняется. В случае полной развилки если «Условие» истинно, то выполняется «БлокОпераций1», иначе выполняется «БлокОпераций2». «БлокОпераций» может состоять из одной операции. В этом случае наличие фигурных скобок, ограничивающих блок, необязательно.

Таблица 2.1

Логическое выражение	Комментарии
<code>if(a < b)</code>	Истинно, если переменная <code>a</code> меньше переменной <code>b</code> и ложно в противном случае.
<code>if(a > b)</code>	Истинно, если переменная <code>a</code> больше переменной <code>b</code> и ложно в противном случае.
<code>if(a == b)</code>	Истинно, если переменная <code>a</code> равна переменной <code>b</code> и ложно в противном случае.
<code>if(a <= b)</code>	Истинно, если переменная <code>a</code> меньше либо равна переменной <code>b</code> и ложно в противном случае.
<code>if(a >= b)</code>	Истинно, если переменная <code>a</code> больше либо равна переменной <code>b</code> и ложно в противном случае.
<code>if(a != b)</code>	Истинно, если переменная <code>a</code> не равна переменной <code>b</code> и ложно в противном случае.
<code>if(a)</code>	Истинно, если переменная <code>a</code> не равна нулю, и ложно в противном случае.

Рассмотрим пример использования оператора ветвления `if`. Следующая программа позволяет определять знак введенной переменной:

```
#include <stdio.h>

int main()
{
    float x;
    printf("Введите число: ");
    scanf("%f",&x);
    if(x < 0)
        printf("Введенное число %f является отрицательным.\n", x);
    if(x >= 0)
        printf("Введенное число %f является неотрицательным.\n", x);
    return 0;
}
```

Анализ приведенного текста программы показывает, что два условных оператора можно заменить одним, используя конструкцию:

```
if (выражение)
<оператор1>
else
<оператор2>
```

которая интерпретируется таким образом. Если «выражение» истинно, то выполняется «оператор1», иначе выполняется «оператор2».

В случаях, когда при выполнении какого-либо условия необходимо записать более одного оператора, необходимо использовать фигурные скобки, т. е. использовать конструкцию вида:

```
if (выражение)
{
<список операторов>
}
else
{
<список операторов>
}
```

Следует отметить, что после ключевого слова `else` формально можно поставить еще один оператор условия `if`, в результате получим еще более гибкую конструкцию условных переходов:

```
if(выражение1) <оператор1>
else if(выражение2) <опреатор2>
else <оператор3>
```

До сих пор рассматривались простые условия типа $x < 0$. Вместе с тем оператор `if` позволяет реализовывать более сложные условные переходы. В языке C имеются три логические операции:

&& - логическое «И»;

|| - логическое «ИЛИ»;

! – логическое «НЕ».

На основе этих трех логических операций можно сформировать более сложные условия. Например, если имеются три переменные `expr1`, `expr2` и `expr3`, то они могут составлять логические конструкции, представленные в таблице 2.2.

Таблица 2.2

Выражение	Комментарии
<code>if(expr1 > expr2 && expr2 < expr3)</code>	Истинно, если значение переменной <code>expr1</code> больше значения переменной <code>expr2</code> И значение переменной <code>expr2</code> меньше значения переменной <code>expr3</code> .
<code>if(expr1 <= expr2 expr1 >= expr3)</code>	Истинно, если значение переменной <code>expr1</code> меньше либо равно значения переменной <code>expr2</code> ИЛИ значение переменной <code>expr2</code> больше либо равно значения переменной <code>expr3</code> .
<code>if(expr1 &&expr2&& !expr3)</code>	Истинно, если истинное значение <code>expr 1</code> И истинно значение <code>expr2</code> И ложно значение <code>expr3</code> .
<code>if(!expr1 !expr2&&expr3)</code>	Истинно, если ложно значение <code>expr1</code> ИЛИ ложно значение <code>expr2</code> И истинно значение <code>expr3</code> .

Подобно операциям умножения и сложения в математике, логические операции «И», «ИЛИ», «НЕ», также имеют свои приоритеты. Самый высокий приоритет имеет операция «НЕ», т. е. такая операция выполняется в первую очередь. Более низкий приоритет у операции «И», и наконец самый малый приоритет у операции «ИЛИ».

1.2 Условный оператор switch.

Условная операция `if` облегчает написание программ, в которых

необходимо производить выбор между небольшим числом возможных вариантов. Однако иногда в программе необходимо осуществить выбор одного варианта из множества возможных. Формально для этого можно воспользоваться конструкцией `if else if ... else`. Однако во многих случаях оказывается более удобным применять оператор `switch` языка C. Синтаксис данного оператора следующий:

```
switch(переменная)
{
case константа1:
<операторы>
case константа2:
<операторы>
...
default:
<операторы>
}
```

Данный оператор последовательно проверяет на равенство переменной константам, стоящим после ключевого слова `case`. Если ни одна из констант не равна значению переменной, то выполняются операторы, находящиеся после слова `default`. Оператор `switch` имеет следующую особенность. Допустим, значение переменной равно значению константы1 и выполняются операторы, стоящие после первого ключевого слова `case`. После этого выполнение программы продолжится проверкой переменной на равенство константы2, что часто приводит к неоправданным затратам ресурсов ЭВМ. Во избежание такой ситуации следует использовать оператор `break` для перехода программы к следующему оператору после `switch`.

Рассмотрим пример программирования условного оператора `switch`:

```
#include <stdio.h>

int main()
```

```

{
int x;
printf("Введите число: ");
scanf("%d",&x);
switch(x)
{
case 1 : printf("Введено число 1\n");break;
case 2 : printf("Введено число 2\n"); break;
default : printf("Введено другое число\n");
}
char ch;
printf("Введите символ: ");
scanf("%c",&ch);
switch(ch)
{
case 'a' : printf("Введен символ a\n"); break;
case 'b' : printf("Введен символ b\n"); break;
default : printf("Введен другой символ\n");
}
return 0;
}

```

Данный пример демонстрирует два разных варианта использования оператора switch. В первом случае выполняется анализ введенной цифры, во втором – анализ введенного символа. Следует отметить, что данный оператор может производить выбор только на основании равенства своего аргумента одному из перечисленных значений case, т. е. проверка выражений типа $x < 0$ в данном случае невозможна.

1.3 Оператор цикла while.

Это тип цикла с предусловием. С помощью данного оператора

реализуется цикл, который выполняется до тех пор, пока истинно условие цикла. Синтаксис данного оператора следующий:

```
while(<условие>)  
{  
<тело цикла>  
}
```

Приведем пример реализации данного цикла, в котором выполняется

суммирование элементов ряда $S = \sum_{i=0}^{\infty} i$ пока $S < N$:

```
int N=20, i = 0;  
long S = 0L;  
while(S < N)  
{  
    S=S+i;  
    i++;  
}
```

В данном примере реализуется цикл `while` с условием $i < N$. Так как начальное значение переменной $i=0$, а $N=20$, то условие истинно и выполняется тело цикла, в котором осуществляется суммирование переменной i и увеличение ее на 1. Очевидно, что на 20 итерации значение $i=20$, условие станет ложным и цикл будет завершен. Продемонстрируем гибкость языка C, изменив данный пример следующим образом:

```
int N=20, i = 0;  
long S = 0L;  
while((S=S+i++) < N);
```

В данном случае при проверке условия сначала выполняются операторы, стоящие в скобках, где и осуществляется суммирование элементов ряда и только, затем, проверяется условие. Результат выполнения обоих вариантов программ одинаковый и $S=21$. Однако последняя

конструкция бывает удобной при реализации опроса клавиатуры, например, с помощью функции `scanf()`:

```
int num;
while(scanf("%d",&mun) == 1)
{
    printf("Вы ввели значение %d\n",num);
}
```

Данный цикл будет работать, пока пользователь вводит целочисленные значения и останавливается, если введена буква или вещественное число.

Следует отметить, что цикл `while` можно принудительно завершить даже при истинном условии цикла. Это достигается путем использования оператора `break`. Перепишем предыдущий пример так, чтобы цикл завершался, если пользователь введет число 0.

```
int num;
while(scanf("%d",&mun) == 1)
{
    if(num == 0) break;
    printf("Вы ввели значение %d\n",num);
}
```

Цикл завершается сразу после использования оператора `break`, т. е. в приведенном примере, при вводе с клавиатуры нуля функция `printf()` выполняться не будет и программа перейдет на следующий оператор после `while`. Того же результата можно добиться, если использовать составное условие в цикле:

```
int num;
while(scanf("%d",&mun) == 1 && num != 0)
{
    printf("Вы ввели значение %d\n",num);
}
```

```
}
```

Таким образом, в качестве условия возможны такие же конструкции, что и в операторе `if`.

1.4 Оператор цикла `for`.

`for` – параметрический цикл (цикл с фиксированным числом повторений). Для организации такого цикла необходимо осуществить три операции:

- инициализация – присваивание параметру цикла начального значения;
- условие – проверка условия повторения цикла, чаще всего – сравнение величины параметра с некоторым граничным значением;
- модификация – изменение значения параметра для следующего прохождения тела цикла.

Работа оператора цикла `for` подобна оператору `while` с той лишь разницей, что оператор `for` подразумевает изменение значения некоторой переменной и проверки ее на истинность. Работа данного оператора продолжается до тех пор, пока истинно условие цикла. Синтаксис оператора `for` следующий:

```
for(<инициализация счетчика>;<условие>;<изменение значения  
счетчика>)  
{  
<тело цикла>  
}
```

Рассмотрим особенность реализации данного оператора на примере вывода таблицы кодов ASCII символов:

```
char ch;  
for(ch = 'a'; ch <= 'z'; ch++)  
    printf("Значение ASCII для %c - %d.\n",ch,ch);
```

В данном примере в качестве счетчика цикла выступает переменная `ch`, которая инициализируется символом `'a'`. Это означает, что в переменную `ch`

заносится число 97 – код символа ‘a’. Именно так символы представляются в памяти компьютера. Код символа ‘z’ – 122, и все малые буквы латинского алфавита имеют коды в диапазоне [97; 122]. Поэтому, увеличивая значение `ch` на единицу, получаем код следующей буквы, которая выводится с помощью функции `printf()`. Учитывая все вышесказанное, этот же пример можно записать следующим образом:

```
for(char ch = 97; ch <= 122; ch++)  
    printf(“Значение ASCII для %c - %d.\n”,ch,ch);
```

Здесь следует отметить, что переменная `ch` объявлена внутри оператора `for`. Это особенность языка C – возможность объявлять переменные в любом месте программы.

Существует много особенностей реализации данного оператора, отметим основные из них, которые могут заметно повысить скорость написания программ. Следующим примером продемонстрируем особенности изменения значения счетчика цикла:

```
int line_cnt = 1;  
double debet;  
for(debet = 100.0; debet < 150.0; debet = debet*1.1,  
    line_cnt++)  
    printf(“%d. Ваш долг теперь равен %.2f.\n”,line_cnt, debet);
```

Следующий фрагмент программы демонстрирует возможность программирования сложного условия внутри цикла:

```
int exit = 1;  
for(int num = 0; num < 100 && !exit; num += 1)  
{  
    scanf(“%d",&mov);  
    if(mov == 0) exit = 0;  
    printf(“Произведение num*mov = %d.\n”,num*mov);  
}
```

Оператор for с одним условием:

```
int i=0;  
for(;i < 100;) i++;
```

и без условия:

```
int i=0;  
for(;;) {i++; if(i > 100) break;}
```

В последнем примере оператор break служит для выхода из цикла for, т. к. он будет работать «вечно» не имея никаких условий.

1.5 Оператор цикла do... while.

Цикл do...while – это цикл с постусловием, где истинность выражения, проверяющего Условие, проверяется после выполнения Блока Операций, заключенного в фигурные скобки. Тело цикла выполняется до тех пор, пока выражение, проверяющее Условие, не станет ложным, то есть тело цикла с постусловием выполнится хотя бы один раз.

Использовать цикл do...while лучше в тех случаях, когда должна быть выполнена хотя бы одна итерация, либо когда инициализация объектов, участвующих в проверке условия, происходит внутри тела цикла.

Бывают ситуации, когда целесообразно выполнять проверку условия после того, как будут выполнены операторы, стоящие внутри цикла. Это достигается путем использования операторов do while, которые реализуют цикл с постусловием. Следующий пример демонстрирует реализацию такого цикла:

```
const int secret_code = 13;  
int code_ent;  
do  
{  
printf(“Введите секретный код: ”);  
scanf(“%d”,&code_ent);  
}
```

```
while(code_ent != secret_code);
```

Из приведенного примера видно, что цикл с постусловием работает до тех пор, пока истинно условие, т.е. в данном случае пока значение введенного кода будет отличаться от значения секретного кода. Также следует обратить внимание на то, что после ключевого слова `while` должна стоять точка с запятой. При реализации данного цикла можно использовать составные условия, подобно циклу `while`, а также принудительно выходить из цикла с помощью оператора `break`.

1.6 Все рассмотренные выше операторы циклов допускают использование любых других операторов языка Си внутри цикла, в том числе и операторов цикла. Это значит, что внутри одного цикла может находиться другой, что приводит к реализации вложенных циклов. Вложенные циклы необходимы для решения большого числа задач, например, вычисления двойных, тройных и т.д. сумм, просмотр элементов двумерного массива и многих других задач.

2 Задание на выполнение работы.

2.1 Написать программу с использованием условного оператора `if` в соответствии с заданным преподавателем вариантом (числовые параметры при необходимости задаются самостоятельно). Исходные данные приведены в таблице 2.3.

2.2 Написать программу с использованием условного оператора `switch` в соответствии с заданным преподавателем вариантом (числовые параметры при необходимости задаются самостоятельно). Исходные данные приведены в таблице 2.3.

2.3 Написать программу с использованием операторов цикла `while` и `for` в соответствии с заданным преподавателем вариантом (числовые параметры при необходимости задаются самостоятельно). Исходные данные приведены в таблице 2.4.

2.4 Написать программу с использованием оператора цикла do...while в соответствии с заданным преподавателем вариантом (числовые параметры при необходимости задаются самостоятельно). Исходные данные приведены в таблице 2.4.

Таблица 2.3

Вариант	Оператор if	Оператор switch
1	Написать программу вычисления модуля введенного числа	Написать программу перевода введенного символа от a до f в верхний регистр
2	Написать программу проверки попадания введенного числа в диапазон от -2 до 2	Написать программу перевода введенного символа от A до F в нижний регистр
3	Написать программу проверки не вхождения введенного числа в диапазон от 0 до 5	Написать программу замены введенного символа от 0 до 9 соответствующим числом
4	Написать программу проверки на положительность введенного числа	Написать программу замены введенного числа от 0 до 9 соответствующим символом
5	Написать программу проверки на отрицательность введенного числа	Написать программу замены введенного числа от 0 до 5 соответствующим символом, а все другие значения заменять буквой z
6	Написать программу определения знака введенного числа	Написать программу замены введенного символа от 0 до 5 соответствующим числом, а все другие символы заменять числом -1
7	Написать программу проверки попадания введенного числа в диапазон от -6 до -2	Написать программу перевода введенного символа от a до f в верхний регистр, а другие символы заменять на Z
8	Написать программу проверки не вхождения введенного числа в диапазон от -5 до -1	Написать программу перевода введенного символа от A до F в нижний регистр, а все другие символы заменять на x
9	Написать программу вычисления суммы модулей двух введенных чисел	Написать программу сравнения введенного числа со значениями 0, 4, 8, 9 и 30

10	Написать программу вычисления $1/a$ с проверкой $a \neq 0$	Написать программу сравнения введенного символа с a, s, d, j и e
11	Написать программу вычисления разности модулей трех введенных чисел	Написать программу, которая проверяет знание начала и конца Второй мировой войны: a. 01.09.1939 -- 2.09.1945 b. 01.09.1939 -- 9.05.1945 c. 22.06.1941 -- 9.05.1945 d. 22.06.1941 -- 2.09.1945
12	Написать программу расчета суммы пяти введенных чисел	Заданы три числа: a, b, c . Определить, могут ли они быть сторонами треугольника, и если да, то определить его тип: равносторонний, равнобедренный, равнососторонний. Условия существования треугольника: $a \leq b + c$, $b \leq a + c$, $c \leq a + b$.
13	Написать программу вычисления $(1+x)/x^2$ с проверкой $x \neq 0$	Треугольник задан длинами своих сторон: a, b, c. Определить, является ли он тупоугольным, прямоугольным или остроугольным.
14	Написать программу нахождения максимального из шести введенных чисел	Даны три числа a, b, c . Найти произведение наибольшего из этих чисел на наименьшее.
15	Написать программу изменения знака записанных шести чисел на положительный	Путник двигался t_1 часов со скоростью v_1 , затем t_2 часов - со скоростью v_2 и t_3 часов - со скоростью v_3 . За какое время он одолел первую половину пути, после чего запланировал привал?
16	Написать программу изменения знака записанных шести чисел на отрицательный	Как известно, число делится на 3 тогда и только тогда, когда сумма его цифр делится на 3. Проверить этот признак на примере заданного трехзначного числа.
17	Написать программу вычисления модуля суммы	Среди заданных целых чисел a, b, c найти пары кратных.

	введенных трех чисел	
18	Написать программу проверки попадания введенного числа в диапазон $[-6; 5)$	Лежит ли точка $M(x,y)$ внутри треугольника, заданного координатами своих, вершин $A(xa, ya)$, $B(xb, yb)$, $C(xc, yc)$ на плоскости?
19	Написать программу проверки не вхождения введенного числа в диапазон $(-6; 5]$	Определить, пройдет ли кирпич со сторонами a и b сквозь прямоугольное отверстие со сторонами r и s . Стороны отверстия должны быть параллельны граням кирпича.
20	Написать программу вычисления модуля разности введенных трех чисел	Четырехугольник $ABCD$ задан координатами своих вершин на плоскости: $A(xa, ya)$, $B(xb, yb)$, $C(xc, yc)$, $D(xd, yd)$. Определить тип четырехугольника: прямоугольник, параллелограмм, трапеция, произвольный четырехугольник.

Таблица 2.4

Вариант	Операторы циклов while и for	Оператор цикла do while
1	Вычислить $\sum_{i=1}^{50} 1/i^2$ с использованием оператора for	Написать программу ввода произвольных чисел до тех пор, пока не будет введено число 0
2	Вычислить $f(x) = kx + b$, при $x = 1, 2, \dots, 100$ с использованием оператора while	Написать программу ввода произвольных символов до тех пор, пока не будет введен символ q
3	Вычислить $\sum_{i=1}^{50} \sum_{j=1}^{30} i + j$ с помощью вложенных циклов for	Написать программу подсчета суммы 10 чисел, вводимых с клавиатуры
4	Вычислить $S = \sum_{i=1}^{\infty} i$ пока $S < 50$ с помощью цикла while	Написать программу вычисления произведения 5 чисел, введенных с клавиатуры
5	Вычислить $S = \sum_{i=1}^{\infty} i^2$ пока $S < 100$ с помощью цикла for	Написать программу вычисления модулей введенных чисел до тех пор, пока пользователь не введет 0

6	Вычислить $\sum_{i=1}^{50} \sum_{j=1}^{10} 1 / (i + j)$ с помощью вложенных циклов while	Написать программу определения знака введенных чисел до тех пор, пока пользователь не введет 0
7	Вычислить $f(x) = x^2 + b$, при $x = -10, -9, \dots, 10$ с использованием оператора for	Написать программу определения минимального введенного числа из 10 чисел
8	Вычислить $\sum_{i=1}^{10} 1 / i^3, i \neq 0$ с использованием оператора for	Написать программу определения максимального введенного числа из 5 чисел
9	Вычислить $\sum_{i=-10}^{20} \sum_{j=0}^{10} 1 / (i + j)^2$ с помощью вложенных циклов for	Написать программу определения минимального среди положительных введенных 10 чисел
10	Вычислить $f(x) = 1 / x, x \neq 0$ при $x = -10, -9, \dots, 10$ с использованием оператора for	Написать программу определения максимального среди отрицательных введенных 7 чисел
11	Подсчитать количество натуральных чисел n ($102 < n < 987$), в которых все три цифры различны	Вводить различные числа. Когда будет введен 0, то программа выходит из цикла
12	Вводим любое число с клавиатуры и нужно определить является ли оно простым	Написать программу, спрашивающую у студента его отметку по экзамену по дисциплине и реагирующую на нее подходящим текстом
13	Определить, является ли натуральное число n степенью числа 3	Выводить на экран степени числа два до 1000
14	Дано натуральное число n . Получить все его натуральные делители	В компьютер вводятся числа. После ввода каждого числа нужно печатать, сколько среди них уже введено отрицательных. Закончить ввод, когда отрицательных станет больше пяти
15	Присвойте переменной star значение 0. Пока значение star не достигнет 55 выводите на экран в строку по одной	Найти сумму чисел от 1 до 1000

	звездочке (*)	
16	Используя внешний и вложенный циклы while организуйте вывод таблицы умножения на экран	Написать программу вычисления суммы $S=1+2+3+\dots+50$
17	Даны целые числа K и N ($N > 0$). Вывести N раз число K	Электронные часы показывают время в формате от 00:00 до 23:59. Подсчитать сколько раз за сутки случается так, что слева от двоеточия показывается симметричная комбинация для той, что справа от двоеточия (например, 02:20, 11:11 или 15:51)
18	Даны два целых числа A и B ($A < B$). Вывести в порядке возрастания все целые числа, расположенные между A и B (включая сами числа A и B), а также количество N этих чисел	Выведите на экран первые 11 членов последовательности Фибоначчи. Напоминаем, что первый и второй члены последовательности равны единицам, а каждый следующий – сумме двух предыдущих
19	Даны два целых числа A и B ($A < B$). Найти сумму квадратов всех целых чисел от A до B включительно	Создать цикл ровно из 256 шагов, тело которого, даже если оно пусто, выполняется ровно 256 раз. В частном случае такой цикл может вывести все целые числа от 0 до 255 включительно
20	Дано целое число N (> 0). Если оно является степенью числа 3, то вывести True, если не является — вывести False	Вводить с клавиатуры целые числа и суммировать, пока не встретится число 0. Сумму вывести на экран

3 Задание на обработку данных.

3.1 Сделать выводы о полученных результатах работы написанных программ.

Содержание отчета

Отчет по лабораторной работе выполняется на листах формат А4 согласно требований ГОСТ и ЕСКД и должен содержать следующие пункты:

- а) титульный лист с названием работы, дисциплины, кода группы, ф.и.о. исполнителя и преподавателя;
- б) цель и содержание работы;
- в) описание ПО, необходимого для выполнения работы;
- г) написанные программы приведенных заданий;
- д) результаты исследований работы программ;
- е) выводы по результатам работы.

Контрольные вопросы

1. Запишите условный оператор if для определения знака переменной var.
2. В каких случаях следует использовать оператор switch?
3. Используя условный оператор, выполните проверку на принадлежность значения переменной диапазону [10; 20).
4. Приведите программу замены малых латинских букв большими с использованием оператора switch.
5. Как записывается логическое равенство в операторе if?
6. Приведите обозначение логического знака «не равно».
7. Какими символами обозначаются логические операции И и ИЛИ в условном операторе if?
8. В чем отличия между операторами while и do while?
9. Дайте понятие вложенных циклов?
10. Что такое цикл с предусловием?
11. Что такое цикл с постусловием?
12. Условие остановки цикла while?
13. Для каких целей используются циклы в программировании?

14. Перечислите операторы циклов в языке С.

Список литературы, рекомендованный к использованию по данной теме

Основная учебная литература

Дополнительная литература

1.

ПРАКТИЧЕСКАЯ РАБОТА 3

Тема: Изучение базовых операций с массивами, строками, правил задания и работы с функциями.

Цель работы: закрепление знаний, полученных на лекциях по разделу 4 (лекция 7). В ходе выполнения лабораторной работы необходимо:

- изучить базовые операции работы с одномерными и двумерными массивами;
- изучить базовые операции работы со строками;
- изучить правила задания и работы с функциями.

Формируемые компетенции: ОПК-9.

Теоретическая часть

Массив – это упорядоченный набор данных, для хранения данных одного типа, идентифицируемых с помощью одного или нескольких индексов. В простейшем случае массив имеет постоянную длину и хранит единицы данных одного и того же типа.

Количество используемых индексов массива может быть различным. Массивы с одним индексом называют одномерными, с двумя – двумерными и т. д. Одномерный массив нестрого соответствует вектору в математике, двумерный – матрице. Чаще всего применяются массивы с одним или двумя индексами, реже – с тремя, ещё большее количество индексов встречается крайне редко.

Поддержка индексных массивов (свой синтаксис объявления, функции для работы с элементами и т. д.) есть в большинстве высокоуровневых языков программирования, включая С. Максимально допустимая размерность массива, типы и диапазоны значений индексов, ограничения на типы элементов определяются языком программирования и/или конкретным транслятором.

В языках программирования, допускающих объявления программистом собственных типов, как правило, существует возможность создания типа «массив». В определении такого типа может указываться размер, тип элемента, диапазон значений и типы индексов. В дальнейшем возможно определение переменных созданного типа. Все такие переменные-массивы имеют одну структуру. Некоторые языки поддерживают для переменных-массивов операции присваивания (когда одной операцией всем элементам массива присваиваются значения соответствующих элементов другого массива).

Любой язык программирования содержит средства представления и обработки текстовой информации. Другое дело, что обычно программист наряду с символами имеет дело с типом данных (формой представления) – строкой, причем особенности ее организации скрыты, а для работы предоставлен стандартный набор функций. В С, наоборот, форма представления строки является открытой, а программист работает с ней «на низком уровне».

В компьютере все значения хранятся в виде чисел. И строки тоже, там нет никаких символов и букв. Строка представляет собой массив чисел. Каждое число соответствует определённому символу, который берётся из таблицы кодировки. При выводе на экран символ отображается определённым образом.

Для хранения строк используются массивы типа `char` – числовой тип, он хранит один байт данных. Но в соответствии с таблицей кодировки каждое из этих чисел связано с символом. И в обратную сторону – каждый символ определяется своим порядковым номером в таблице кодировки.

В предыдущих лабораторных работах неоднократно использовались различные функции подключаемых библиотек. Вместе с тем существующих функций языка С недостаточно для написания собственных программ и возникает необходимость создания своих функций. В связи с этим нужно

понимать, в каких случаях целесообразно создавать свои функции. Обычно это делается для избавления много раз писать один и тот же код в программе. Например, если часто выполняются действия копирования одной строки в другую, то такую операцию лучше определить в виде функции и использовать ее по мере необходимости.

Именно поэтому важным является изучение базовых операций с массивами, строками, а также правил задания и работы с функциями, получение навыков программирования с их использованием.

Оборудование и материалы

Лабораторная работа выполняется с использованием Microsoft Visual Studio.

Указания по технике безопасности

Перед проведением лабораторной работы следует изучить инструкцию по технике безопасности при работе на компьютере. Также следует соблюдать правила и требования, установленные для данного компьютерного класса.

Задания

1 Задание на подготовку к выполнению работы.

1.1 Массивы.

Представление данных в виде отдельных переменных не всегда достаточно при программировании реальных задач. Например, для представления поведения сигнала во времени или хранения информации об изображении удобно использовать специальный тип данных – массивы.

Одномерные массивы можно ассоциировать с компонентами вектора, а двумерные – с матрицами. В общем случае массив – это набор элементов данных одного типа, для объявления которого используется следующий синтаксис:

```
<тип данных> <имя массива>[число элементов];
```

Например,

```
int array_int[100]; //одномерный массив 100 целочисленных
// элементов
```

```
double array_d[25]; //одномерный массив 25 вещественных
// элементов
```

Как видно из примеров, объявление массивов отличается от объявления обычных переменных наличием квадратных скобок []. Также имена массивов выбираются по тем же правилам, что и имена переменных. Обращение к отдельному элементу массива осуществляется по номеру его индекса. Следующий фрагмент программы демонстрирует запись в массив значений линейной функции $f(x) = kx + b$ и вывода значений на экран:

```
double k=0.5, b = 10.0;
double f[100];
for(int x=0; x < 100; x++)
{
    f[x] = k*x+b;
    printf("%.2f ",f[x]);
}
```

В языке C предусмотрена возможность инициализации массива в момент его объявления, например, таким образом:

```
int powers[4] = {1, 2, 4, 6};
```

В этом случае элементу powers[0] присваивается значение 1, powers[1] – 2, и т. д. Особенностью инициализации массивов является то, что их размер можно задавать только константами, а не переменными. Например,

следующая программа приведет к ошибке при компиляции:

```
int N=100;  
float array_f[N]; //ошибка, так нельзя
```

Поэтому при объявлении массивов обычно используют такой подход:

```
#include <stdio.h>  
  
#define N 100  
  
int main()  
{  
    float array_f[N];  
    return 0;  
}
```

Следует отметить, что при инициализации массивов число их элементов должно совпадать с его размерностью. Рассмотрим вариант, когда число элементов при инициализации будет меньше размерности массива:

```
#define SIZE 4  
  
int data[SIZE]={512, 1024};  
for(int i = 0; i < SIZE; i++)  
    printf("%d, \n", data[i]);
```

Результат работы программы будет следующим:

```
512, 1024, 0, 0
```

Из полученного результата видно, что неинициализированные элементы массива принимаются равными нулю. В случаях, когда число элементов при инициализации превышает размерность массива, то при компиляции произойдет ошибка. Поэтому, когда наперед неизвестно число элементов, целесообразно использовать такую конструкцию языка C++:

```
int data[] = {2, 16, 32, 64, 128, 256};
```

В результате инициализируется одномерный массив размерностью 6 элементов.

Для хранения некоторых видов информации, например, изображений

удобно пользоваться двумерными массивами. Объявление двумерных массивов осуществляется следующим образом:

```
int array2D[100][20]; //двумерный массив 100x20 элементов
```

Нумерация элементов также начинается с нуля, т.е. `array2D[0][0]` соответствует первому элементу, `array2D[0][1]` – элементу первой строки, второго столбца и т. д. Для начальной инициализации двумерного массива может использоваться следующая конструкция:

```
long array2D[3][2] = {{1, 2}, {3, 4}, {5, 6}};
```

или

```
long array2D[][] = {{1, 2}, {3, 4}, {5, 6}};
```

В общем случае можно задать массив любой размерности и правила работы с ними аналогичны правилам работы с одномерными и двумерными массивами.

1.2 Строки.

В языке C нет специального типа данных для строковых переменных. Для этих целей используются массивы символов (тип `char`). Следующий пример демонстрирует использование строк в программе:

```
char str_1[100] = {'П', 'р', 'и', 'в', 'е', 'т', '\0'};
```

```
char str_2[100] = "Привет";
```

```
char str_3[] = "Привет";
```

```
printf("%s\n%s\n%s\n", str_1, str_2, str_3);
```

В приведенном примере показаны три способа инициализации строковых переменных. Первый способ является классическим объявлением массива, второй и третий используются специально для строк. Причем в последнем случае, компилятор сам определяет нужную длину массива для записи строки.

Анализируя первый и второй способы инициализации массива символов возникает вопрос: каким образом язык C++ «знает» где заканчивается строка?

Действительно, массив `str_2` содержит 100 элементов, а массив `str_3` меньше 100, тем не менее длина строки и в первом и во втором случаях одна и та же. Такой эффект достигается за счет использования специальных управляющих кодов, которые говорят где заканчивается строка или где используется перенос внутри одной строки и т. п. В частности символ `'\0'` означает в языке C++ конец строки и все символы после него игнорируются как символы строки.

Следующий пример показывает особенность использования данного специального символа.

```
char str1[10] = {'Л', 'е', 'к', 'ц', 'и', 'я', '\0'};  
char str2[10] = {'Л', 'е', 'к', 'ц', '\0', 'и', 'я' };  
char str3[10] = {'Л', 'е', '\0', 'к', 'ц', 'и', 'я' };  
printf("%s\n%s\n%s\n", str1, str2, str3);
```

Результатом работы данного кода будет вывод следующих трех строк:

Лекция

Лекц

Ле

Из этого примера видно как символ конца строки `'\0'` влияет на длину строк. Таким образом, чтобы подсчитать длину строки (число символов) необходимо считать символы до тех пор, пока не встретится символ `'\0'` или не будет достигнут конец массива. Функция вычисления размера строк уже реализована в стандартной библиотеке языка C `string.h` и имеет следующий синтаксис:

```
int strlen(char* str);
```

где `char* str` – указатель на строку (об указателях речь пойдет ниже).

Следующая программа показывает использование функции `strlen()`.

Пример использования функции `strlen()`.

```
#include <stdio.h>
```

```
#include <string.h>
```

```

int main(void) {
char str[] = "Привет мир!";
int length = strlen(str);
printf("Длина строки = %d.\n",length);
return 0;
}

```

Результатом работы программы будет вывод на экран числа 11. Учитывая, что первый символ имеет нулевой индекс, то можно заметить, что данная функция считает и символ '\0'.

Теперь рассмотрим правила присваивания одной строковой переменной другой. Допустим, объявлены две строки:

```

char str1[] = "Это первая строка";
char str2[] = "Это вторая строка";

```

и необходимо выполнить оператор присваивания:

```

str1 = str2;

```

При такой записи оператора присваивания компилятор выдаст сообщение об ошибке. Для того чтобы выполнить копирование необходимо перебирать по порядку элементы одного массива и присваивать их другому массиву. Данная функция реализована в библиотеке языка C string.h и имеет следующее определение:

```

char* strcpy(char* dest, char* src);

```

Она выполняет копирование строки src в строку dest и возвращает строку dest. В следующем примере показано использование функции strcpy().

Пример использования функции strcpy().

```

#include <stdio.h>
#include <string.h>
int main(void) {
char src[] = "Привет мир!";
char dest[100];

```

```
strcpy(dest,src);
printf("%s\n",dest);
return 0;
}
```

Кроме операций вычисления длины строки и копирования строк важной является операция сравнения двух строк между собой. В языке С две строки считаются одинаковыми, если равны их длины и элементы одной строки равны соответствующим элементам другой. Функция сравнения двух строк имеет вид:

```
int strcmp(char* str1, char* str2);
```

и реализована в библиотеке `string.h`. Данная функция возвращает нуль, если строки `str1` и `str2` равны и не нуль в противном случае. Приведем пример использования данной функции:

```
char str1[] = "Это первая строка";
char str2[] = "Это вторая строка";
if(strcmp(str1,str2) == 0) printf("Строка %s равна строке %s\n",str1,str2);
else printf("Строка %s не равна строке %s\n",str1,str2);
```

В языке С имеется несколько функций, позволяющих вводить строки с клавиатуры. Самой распространенной из них является ранее рассмотренная функция `scanf()`, которой в качестве параметра передается ссылка на массив символов:

```
char str[100];
scanf("%s",str);
```

В результате выполнения этого кода, переменная `str` будет содержать введенную пользователем последовательность символов. Кроме функции `scanf()` также часто используют функцию `gets()` библиотеки `stdio.h`, которая в качестве аргумента принимает ссылку на массив символов:

```
gest(str);
```

Данная функция считывает символы до тех пор, пока пользователь не

нажмет клавишу Enter, т. е. введет символ перевода строки '\n'. Затем она записывает вместо символа '\n' символ '\0' и передает строку вызывающей программе.

Для вывода строк на экран помимо функции printf() можно использовать также функцию puts() библиотеки stdio.h, которая более проста в использовании. Следующий пример демонстрирует применение данной функции.

```
#define DEF "Заданная строка"
char str[] = "Это первая строка";
puts(str);
puts(DEF);
puts(&str[4]);
```

Результат работы следующий:

Это первая строка

Заданная строка

первая строка

Еще одной удобной функцией работы со строками является функция sprintf() библиотеки stdio.h. Ее действие аналогично рассмотренной ранее функции printf() с той лишь разницей, что результат вывода заносится в строковую переменную, а не на экран:

```
int age;
char name[100], str[100];
printf("Введите Ваше имя: ");
scanf("%s",name);
printf("Введите Ваш возраст: ");
scanf("%d",&age);
sprintf(str,"Здравствуйте %s. Ваш возраст %d лет",name,age);
puts(str);
```

В результате массив str будет содержать строку «Здравствуйте ... Ваш

возраст...».

Анализ последнего примера показывает, что с помощью функции `sprintf()` можно преобразовывать числовые переменные в строковые, объединять несколько строк в одну и т. п.

1.3 Функции.

Для объявления функции используется следующий синтаксис:

<тип> <имя функции> ([список параметров]) { <тело функции> }

Тип определяет возвращаемый тип функции. Имя функции служит для ее вызова в программе и ее правило определения совпадает с правилом определения имен переменных. Список параметров необходим для передачи функции каких-либо данных при ее вызове. Тело функции – это набор операторов, которые выполняются при ее вызове. Следующий пример показывает правило определения пользовательских функций.

Пример задания функции.

```
double square(double x)
{
    x = x*x;
    return x;
}

int main()
{
    double arg = 5;
    double sq1=square(arg);
    double sq2=square(3);
    return 0;
}
```

В данном примере задается функция с именем `square`, которая принимает один входной параметр типа `double`, возводит его в квадрат и возвращает вычисленное значение вызывающей программе с помощью

оператора `return`. Следует отметить, что работа функции завершается при вызове оператора `return`. Даже если после этого оператора будут находиться другие операторы, то они выполняться не будут. Например,

```
int square(int x)
{
    x = x*x;
    return x;
    printf("%d",x);
}
```

при вызове данной функции оператор `printf()` не будет выполнен никогда, т. к. оператор `return` завершит работу функции `square`. Оператор `return` является обязательным, если функция возвращает какие-либо значения. Если же она имеет тип `void`, т. е. ничего не возвращает, то оператор `return` может не использоваться.

Пользуясь рассмотренными правилами, можно создавать множество своих функций. При этом важно, чтобы объявление функции было раньше ее использования в программе подобно переменным. Именно поэтому во всех примерах объявление функций осуществляется до функции `main()`, в которой они вызываются.

Функция может принимать произвольное число аргументов, но возвращает только один или не одного (тип `void`). Для задания нескольких аргументов функции используется следующая конструкция:

```
void show(int x,int y,int z) {}
```

Здесь следует обратить внимание на то, что каждой переменной в списке аргументов функции предшествует ее тип. В отличие от объявления обычных переменных. Поэтому следующая программная строка приведет к сообщению об ошибке на этапе компиляции:

```
void show(int x, y, z) {} //неверное объявление
```

Если число пользовательских функций велико (50 и выше), то

возникает неудобство в их визуальном представлении в общем тексте программы. Действительно, имея список из 100 разных функций с их реализациями, в них становится сложно ориентироваться и вносить необходимые изменения. Для решения данной проблемы в языке С при создании своих функций можно пользоваться правилом: сначала задаются объявления функции, а затем их реализации.

Язык С позволяет задавать функции с одинаковыми именами, но разными типами входных аргументов. Следующий пример демонстрирует удобство использования таких функций при их вызове.

Пример использования перегруженных функций.

```
#include <stdio.h>

double abs(double arg);
float abs(float arg);
int abs(int arg);
int main()
{
    double a_d = -5.6;
    float a_f = -3.2;
    int a_i;
    a_d = abs(a_d);
    a_f = abs(a_f);
    a_i = abs(-8);
    return 0;
}

double abs(double arg)
{
    if(arg < 0) arg = arg*(-1);
    return arg;
}
```

```

float abs(float arg)
{
return (arg < 0) ? -arg : arg;
}
int abs(int arg)
{
return (arg < 0) ? -arg : arg;
}

```

В представленной программе задаются три функции с именем `abs` и разными входными и выходными аргументами для вычисления модуля числа. Благодаря такому объявлению при вычислении модуля разных типов переменных в функции `main()` используется вызов функции с одним и тем же именем `abs`. При этом компилятор в зависимости от типа переменной автоматически выберет нужную функцию. Такой подход к объявлению функций называется перегрузкой.

В языке C можно задавать значения аргументов функции, которые будут использоваться по умолчанию, т. е. если программист не введет свое значение. Приведенный ниже фрагмент программы демонстрирует правило использования аргументов по умолчанию.

```

void some_func(int a = 1, int b = 2, int c = 3)
{
printf("a = %d, b = %d, c = %d\n",a,b,c);
}

```

Благодаря начальной инициализации значений переменных, функция `some_func()` может быть вызвана с разным набором аргументов:

```

int main(void)
{
show_func(10);
show_func(10,20);
}

```

```

show_func(10,20,30);
return 0;
}

```

В результате, на экране появятся следующие строки:

```

a = 1, b = 2, c = 3
a = 10, b = 2, c = 3
a = 10, b = 20, c = 3
a = 10, b = 20, c = 30

```

Из полученного результата видно, что по умолчанию значения аргументов равны установленным значениям при определении функции. В случае ввода новых значений, переменные *a*, *b* и *c* соответственно меняют свои значения на введенные.

При использовании значений аргументов по умолчанию следует пользоваться правилом: аргументы со значениями по умолчанию должны находиться в списке аргументов функции последними. Следующий пример показывает правильные и неправильные объявления функций:

```

void my_func(int a, int b = 1, int c = 1); //правильное объявление
void my_func(int a, int b, int c = 1); //правильное объявление
void my_func(int a=1, int b, int c = 1); //неправильное объявление
void my_func(int a, int b = 1, int c); //неправильное объявление

```

В языке C допускается чтобы функция вызывала саму себя. Этот процесс называется рекурсией. В некоторых задачах программирования такой подход позволяет заметно упростить создаваемый программный код. Рассмотрим данный процесс на следующем примере.

Пример использования рекурсивных функций.

```

#include <stdio.h>

void up_and_down(int );

int main(void)
{

```

```

up_and_down(1);
return 0;
}
void up_and_down(int n)
{
printf("Уровень вниз %d\n",n);
if(n < 4) up_and_down(n+1);
printf("Уровень вверх %d\n",n);
}

```

Результатом работы этой программы будет вывод на экран следующих строк:

```

Уровень вниз 1
Уровень вниз 2
Уровень вниз 3
Уровень вниз 4
Уровень вверх 4
Уровень вверх 3
Уровень вверх 2
Уровень вверх 1

```

Полученный результат работы программы объясняется следующим образом. Вначале функция `main()` вызывает функцию `up_and_down()` с аргументом 1. В результате аргумент `n` данной функции принимает значение 1 и функция `printf()` печатает первую строку. Затем выполняется проверка и если $n < 4$, то снова вызывается функция `up_and_down()` с аргументом на 1 больше $n+1$. В результате вновь вызванная функция печатает вторую строку. Данный процесс продолжается до тех пор, пока значение аргумента не станет равным 4. В этом случае оператор `if` не сработает и вызовется функция `printf()`, которая печатает пятую строку «Уровень вверх 4». Затем функция завершает свою работу и управление передается функции, которая вызывала

данную функцию. Это функция `up_and_down()` с аргументом `n=3`, которая также продолжает свою работу и переходит к оператору `printf()`, который печатает 6 строку «Уровень вверх 3». Этот процесс продолжается до тех пор, пока не будет достигнут исходный уровень, т. е. первый вызов функции `up_and_down()` и управление вновь будет передано функции `main()`, которая завершит работу программы.

2 Задание на выполнение работы.

2.1.1 Написать программу работы с одномерным массивом в соответствии с заданным преподавателем вариантом (числовые параметры при необходимости задаются самостоятельно). Исходные данные приведены в таблице 3.1.

2.1.2 Написать программу с двумерным массивом в соответствии с номером заданным преподавателем вариантом (числовые параметры при необходимости задаются самостоятельно). Исходные данные приведены в таблице 3.1.

Таблица 3.1

Вариант	Одномерный массив	Двумерный массив
1	Записать в массив значения функции $f(x) = kx + b$ при $x = 1, 2, \dots, 100$ и вывести его на экран	Занести в массив значения функции $f(x, y) = x + y$, $0 \leq x \leq 20$, $0 \leq y \leq 10$ и вывести его на экран
2	Записать в массив значения функции $f(x) = a \cdot \sin(x / 100)$, при $x = 1, 2, \dots, 100$ и вывести его на экран	Написать программу ввода в массив 5x4 элемента чисел и поиска в нем максимального значения
3	Написать программу ввода в массив 20 чисел и поиска в нем максимального значения	Занести в массив значения функции $f(x, y) = 1 / (x + y)$, $0 \leq x \leq 30$, $1 \leq y \leq 20$ и вывести его на экран
4	Записать в массив значения функции $f(x) = a \cdot \cos(x / 50)$, при $x = 1, 2, \dots, 100$ и вывести его на	Написать программу ввода в массив 6x3 элемента чисел и поиска в нем минимального

	экран	значения
5	Написать программу ввода в массив 10 чисел и поиска в нем минимального значения	Занести в массив значения функции $f(x,y) = (x+y)^2$, $0 \leq x \leq 5$, $0 \leq y \leq 3$ и вывести его на экран
6	Записать в массив значения функции $f(x) = x^2 + b$, при $x = 1, 2, \dots, 10$ и вывести его на экран	Написать программу ввода в массив 6x5 элементов чисел и вычисления суммы элементов полученного массива
7	Написать программу ввода в массив 20 чисел и вычисления суммы элементов полученного массива	Занести в массив значения функции $f(x,y) = 1/\left((x+y)^2 + 1\right)$, $0 \leq x \leq 5$, $0 \leq y \leq 10$ и вывести его на экран
8	Написать программу ввода в массив 5 чисел и вычисления произведения элементов полученного массива	Написать программу ввода в массив 3x3 элемента чисел и вычисления произведения элементов полученного массива
9	Записать в массив значения функции $f(x) = 1/x + b$, при $x = 1, 2, \dots, 50$ и вывести его на экран	Занести в массив значения функции $f(x,y) = x - y$, $0 \leq x \leq 20$, $0 \leq y \leq 10$ и вывести его на экран
10	Написать программу ввода в массив 10 чисел и поиска в нем модуля максимального значения	Написать программу ввода в массив 4x4 элементов чисел и поиска в нем модуля максимального значения
11	Написать программу ввода в массив 7 чисел и вычисления разности элементов полученного массива	Занести в массив значения функции $f(x,y) = x - y^2 + 2$, $0 \leq x \leq 20$, $0 \leq y \leq 5$ и вывести его на экран
12	Записать в массив значения функции $f(x) = (1+b)/(x+b)$, при $x = 1, 2, \dots, 40$ и вывести его на экран	Написать программу ввода в массив 5x5 элементов чисел и поиска в нем среднего значения
13	Написать программу ввода в массив 10 чисел и поиска в нем среднего значения	Занести в массив значения функции $f(x,y) = 5 \cdot \left((x^2 + y)^2 - 15\right)$,

		$0 \leq x \leq 5, 0 \leq y \leq 5$ и вывести его на экран
14	Записать в массив значения функции $f(x) = x^2 - 0,5 \cdot b^2$, при $x = 1, 2, \dots, 8$ и вывести его на экран	Написать программу ввода в массив 5x5 элемента чисел и вычисления частного элементов полученного массива
15	Написать программу ввода в массив 10 чисел и вычисления суммы квадратов его элементов	Занести в массив значения функции $f(x, y) = (x^{0,5} - y)^2$, $0 \leq x \leq 5, 0 \leq y \leq 7$ и вывести его на экран
16	Записать в массив значения функции $f(x) = a^2 \cdot \sin(x \cdot 20)$, при $x = 1, 2, \dots, 70$ и вывести его на экран	Написать программу ввода в массив 7x3 элементов чисел и вычисления разности элементов полученного массива
17	Написать программу ввода в массив 20 чисел и поиска в нем максимального и минимального значений	Занести в массив значения функции $f(x, y) = (x^3 - y) + 5$, $0 \leq x \leq 15, 0 \leq y \leq 10$ и вывести его на экран
18	Записать в массив значения функции $f(x) = x^2 + b$ при $x = 1, 2, \dots, 85$ и вывести его на экран	Написать программу ввода в массив 6x4 элементов чисел и поиска в нем суммы модулей
19	Написать программу ввода в массив 20 чисел и поиска в нем четных значений	Занести в массив значения функции $f(x, y) = (x - y)^{0,4} + b$, $0 \leq x \leq 20, 0 \leq y \leq 40$ и вывести его на экран
20	Записать в массив значения функции $f(x) = (x + a)^2$ при $x = 1, 2, \dots, 60$ и вывести его на экран	Написать программу ввода в массив 3x5 элементов чисел и поиска в нем минимального и максимального значений

2.2 Написать две программы по работе со строками в соответствии с заданным преподавателем вариантом (строковые параметры при необходимости задаются самостоятельно). Исходные данные приведены в таблице 3.2.

Таблица 3.2

Вариант	1 -е задание	2-е задание
1	Написать программу поэлементного копирования строки «Hello World» в другой символьный массив	Написать программу объединения трех строк «Лабораторная», «работа» и «№3» в четвертую строку с текстом: «Лабораторная работа №3» без использования функции <code>sprintf()</code>
2	Написать программу замены во введенной строке малых букв а на заданные	Написать программу удаления букв и из введенной строки
3	Написать программу подсчета букв е во введенной строке	Написать программу добавления слова «hello» после первого слова введенной строки *)
4	Написать программу удаления букв о из введенного слова	Написать программу сравнения двух строк с помощью функции <code>strcmp()</code>
5	Написать программу добавления пробела после каждой буквы а введенной строки	Написать программу замены во введенной строке заглавных букв О на малые
6	Написать программу подсчета числа слов в строке *)	Написать программу подсчета букв 'и' во введенной строке
7	Написать программу выделения первого слова из введенной строки *) и отображение его на экране	Написать программу удаления всех пробелов из введенной строки
8	Написать программу выделения последнего слова из введенной строки *) и отображение его на экране	Написать программу копирования первой половины введенной строки в другую строку
9	Написать программу вывода введенного слова задом на перед	Написать программу сравнения первых половин двух введенных строк
10	Написать программу удаления последнего слова из строки *)	Написать программу замещения первой половины строки второй, а второй – первой
11	Написать программу замены всех прописных букв на строчные	Написать программу добавления слова «Уважаемый» перед первым словом введенных Ф.И.О *)
12	Написать программу подсчета букв о во введенной строке	Написать программу сравнения двух строк с помощью функции <code>strcmp()</code>

13	Написать программу удаления каждой второй буквы о из введенного предложения	Написать программу объединения четырех любых строк поздравления в четвертую строку без использования функции <code>sprintf()</code>
14	Написать программу удаления всех пробелов и знаков пунктуации, кроме точки	Написать программу удаления всех знаков переноса из введенной строки
15	Написать программу, которая делает все гласные буквы прописными	Написать программу сравнения вторых половин двух введенных строк
16	Написать программу, которая удаляет все согласные буквы	Написать программу замены во введенной строке малых букв а на заглавные
17	Написать программу вывода введенного слова кириллицей при вводе слова на латинице	Написать программу сравнения двух строк из не менее, чем восьми смысловых слов, с помощью функции <code>strcmp()</code>
18	Написать программу удаления первого и последнего слова из строки *)	Написать программу замены во введенной строке знаков пунктуации на пробелы
19	Написать программу добавления точки после каждой буквы введенной строки	Написать программу удаления всех знаков пунктуации из введенного текста, длиной, не менее трех предложений
20	Написать программу удаления точки после каждой буквы введенной строки	Написать программу сравнения трех строк с помощью функции <code>strcmp()</code>
* считается, что слова разделяются пробелом.		

2.3 Написать две программы по работе со строками в соответствии с заданным преподавателем вариантом (числовые параметры при необходимости задаются самостоятельно). Исходные данные приведены в таблице 3.3.

Таблица 3.3

Вариант	1-е задание	2-е задание
---------	-------------	-------------

1	Написать функцию вычисления площади прямоугольника	Используя перегрузку функций, написать программу определения знака переменных разного типа
2	Написать функцию вычисления периметра прямоугольника	С помощью рекурсивной функции осуществить вывод на экран элементов одномерного массива
3	Написать функцию вычисления длины окружности	Используя перегрузку функций, написать программу вычисления суммы элементов массива разных типов
4	Написать функцию вычисления площади круга	С помощью рекурсивной функции осуществить поиск максимального элемента одномерного массива
5	Написать функцию вычисления объема параллелепипеда	Используя перегрузку функций, написать программу определения максимального значения элемента массива разного типа
6	Написать функцию вычисления евклидова расстояния между двумя точками	С помощью рекурсивной функции осуществить поиск минимального элемента одномерного массива
7	Написать функцию вычисления суммы элементов массива	Используя перегрузку функций, написать программу определения минимального значения элемента массива разного типа
8	Написать функцию нахождения максимального значения элемента массива	С помощью рекурсивной функции вычислить сумму элементов одномерного массива
9	Написать функцию нахождения минимального значения элемента массива	Используя перегрузку функций, написать программу вычисления произведения двух переменных разного типа
10	Написать функцию вычисления произведения элементов массива	С помощью рекурсивной функции вычислить среднее арифметическое элементов одномерного массива
11	Написать функцию вычисления разности элементов массива	С помощью рекурсивной функции осуществить вывод на

		экран положительных элементов одномерного массива
12	Написать функцию вычисления периметра 8 угольника	Используя перегрузку функций, написать программу определения знака переменных разного типа
13	Написать функцию вычисления суммы квадратов элементов массива	С помощью рекурсивной функции вычислить разность элементов одномерного массива
14	Написать функцию нахождения четных элементов массива	Используя перегрузку функций, написать программу определения первого положительного значения элемента массива разного типа
15	Написать функцию нахождения отрицательных элементов массива	С помощью рекурсивной функции осуществить вывод на экран отрицательных элементов одномерного массива
16	Написать функцию нахождения элементов массива, равных нулю	Используя перегрузку функций, написать программу определения первого отрицательного элемента массива разного типа
17	Написать функцию нахождения катета прямоугольного треугольника по катету и гипотенузе	Используя перегрузку функций, написать программу нахождения минимального и максимального элементов массива
18	Написать функцию нахождения нечетных элементов массива	Используя перегрузку функций, написать программу определения знака переменных разного типа
19	Написать функцию нахождения синуса угла по противолежащему катету и гипотенузе	С помощью рекурсивной функции осуществить вывод на экран элементов одномерного массива
20	Написать функцию вычисления объема сферы	С помощью рекурсивной функции осуществить поиск элемента, большего, чем число 3 одномерного массива

3 Задание на обработку данных.

3.1 Сделать выводы о полученных результатах работы написанных программ.

Содержание отчета

Отчет по лабораторной работе выполняется на листах формат А4 согласно требований ГОСТ и ЕСКД и должен содержать следующие пункты:

- а) титульный лист с названием работы, дисциплины, кода группы, ф.и.о. исполнителя и преподавателя;
- б) цель и содержание работы;
- в) описание ПО, необходимого для выполнения работы;
- г) написанные программы приведенных заданий;
- д) результаты исследований работы программ;
- е) выводы по результатам работы.

Контрольные вопросы

1. Каким образом задаются одномерные массивы в языке С?
2. Запишите массив целых чисел с начальными значениями 1, 2 и 3.
3. Каким образом задаются двумерные массивы в языке С?
4. В чем преимущества массивов перед переменными?
5. Как записать значение в элемент массива?
6. Как отобразить элементы массива на экране монитора?
7. Как задаются строки в программе на языке С?
8. Для чего предназначена функция `strcpy()` и в какой библиотеке она определена?
9. Запишите возможные способы начальной инициализации строки.
10. Какой управляющий символ соответствует концу строки?
11. Что выполняет функция `strcmp()`?

12. Какую роль играют структуры в программировании?
13. Что возвращает функция `strlen()`?
14. Запишите прототип функции, которая принимает два целочисленных аргумента и возвращает вещественное число.
15. Допустим, даны три функции:
- ```
int abs(int x);
float abs(float x);
long abs(long x);
```
- Какая из этих трех функций будет вызвана в строке *float a = abs(-6);*?
16. Запишите функцию возведения числа в квадрат.
17. Дайте понятие рекурсии.
18. В каких задачах целесообразно использовать рекурсивные функции?
19. Приведите функцию с тремя аргументами, один из которых задан со значением по умолчанию.

**Список литературы, рекомендованный к использованию по данной теме**

### **Основная учебная литература**

### **Дополнительная литература**

- 1.

## ПРАКТИЧЕСКАЯ РАБОТА 4

**Тема:** Изучение основ программирования в интегрированной среде разработки Arduino на базе отладочной платформы Arduino.

**Цель работы:** закрепление знаний, полученных на лекциях по разделам 2 (лекции 2, 3 и 4) и 4 (лекция 8). В ходе выполнения лабораторной работы необходимо:

- изучить аппаратные средства платформы Arduino и основы программирования;
- установить драйвер и программное обеспечение – программную оболочку Arduino IDE;
- изучить интерфейс программного обеспечения;
- подключить плату Arduino и выполнить необходимые настройки аппаратного и программного обеспечения;
- выполнить тестирование аппаратной и программной части с использованием стандартного скетча;
- написать программы для работы с простейшими устройствами согласно вариантам.

**Формируемые компетенции:** ОПК-9.

### Теоретическая часть

Arduino – это инструмент для проектирования электронных устройств (электронный конструктор) более плотно взаимодействующих с окружающей физической средой, чем стандартные персональные компьютеры, которые фактически не выходят за рамки виртуальности. Это платформа, предназначенная для «physical computing» с открытым программным кодом, построенная на простой печатной плате с современной средой для написания программного обеспечения.

Arduino применяется для создания электронных устройств с

возможностью приема сигналов от различных цифровых и аналоговых датчиков, которые могут быть подключены к нему, и управления различными исполнительными устройствами. Проекты устройств, основанные на Arduino, могут работать самостоятельно или взаимодействовать с программным обеспечением на компьютере. Среда разработки программ с открытым исходным текстом доступна для бесплатного скачивания.

Язык программирования Arduino является стандартным C++ (используются компиляторы семейства GNU Compiler Collection) с некоторыми особенностями, облегчающими новичкам написание первой работающей программы.

Программы, написанные программистом Arduino, называются наброски (или иногда скетчи – транслитерация от англ. sketch) и сохраняются в файлах с расширением \*.ino. Эти файлы перед компиляцией обрабатываются препроцессором Ардуино. Также существует возможность создавать и подключать к проекту стандартные файлы C++.

Обязательную в C++ функцию main() препроцессор Arduino создает сам, вставляя туда необходимые «черновые» действия.

Программист должен написать две обязательные для Arduino функции setup() и loop(). Первая вызывается однократно при старте, вторая выполняется в бесконечном цикле.

В текст своей программы (скетча) программист не обязан вставлять заголовочные файлы используемых стандартных библиотек. Эти заголовочные файлы добавит препроцессор Arduino в соответствии с конфигурацией проекта. Однако пользовательские библиотеки нужно указывать.

Менеджер проекта Arduino IDE имеет нестандартный механизм добавления библиотек. Библиотеки в виде исходных текстов на стандартном C++ добавляются в специальную папку в рабочем каталоге IDE. При этом

название библиотеки добавляется в список библиотек в меню IDE. Программист отмечает нужные библиотеки, и они вносятся в список компиляции.

Arduino IDE не предлагает никаких настроек компилятора и минимизирует другие настройки, что упрощает начало работы для новичков и уменьшает риск возникновения проблем.

## **Оборудование и материалы**

Лабораторная работа выполняется с использованием аппаратной платформы Arduino и программной оболочки Arduino IDE.

## **Указания по технике безопасности**

Перед проведением лабораторной работы следует изучить инструкцию по технике безопасности:

- при работе с аппаратной платформой Arduino IDE;
- при работе на компьютере.

**ВНИМАНИЕ!** Все работы по сборке схемы, изменению порядка внешних соединений, подключению измерительных приборов должны производиться ТОЛЬКО при отключенном напряжении источника питания и/или отключенном кабеле USB, соединяющем плату с ПК.

Включение напряжения питания необходимо производить только после проверки преподавателем правильности коммутаций.

При подключении измерительных приборов необходимо внимательно и заранее устанавливать пределы измерения, во избежание повреждения приборов.

Также следует соблюдать правила и требования, установленные для данного компьютерного класса.

## Задания

1 Задание на подготовку к выполнению работы.

1.1 Изучение аппаратных средств платформы Arduino.

Используемый в работе Arduino Uno контроллер построен на микроконтроллере ATmega328. Платформа имеет 14 цифровых вход/выходов (6 из которых могут использоваться как выходы ШИМ), 6 аналоговых входов, кварцевый генератор 16 МГц, разъем USB, силовой разъем, разъем ICSP и кнопку перезагрузки. Для работы необходимо подключить платформу к компьютеру посредством кабеля USB, либо подать питание при помощи адаптера AC/DC или батареи.

Схема электрическая принципиальная платы Arduino Uno приведена на рисунке 4.1, а диаграмма выводов – на рисунке 4.2.

1.2 Язык программирования Arduino.

Язык программирования устройств Arduino основан на C/C++ и скомпонован с библиотекой AVR Libc и позволяет использовать любые ее функции.

Язык Arduino можно разделить на четыре раздела: операторы, данные (переменные и константы), функции и библиотеки.

Операторы:

- setup();
- loop();
- управляющие операторы;
- синтаксис;
- арифметические операторы;
- операторы сравнения;
- логические операторы;
- указатели доступа;



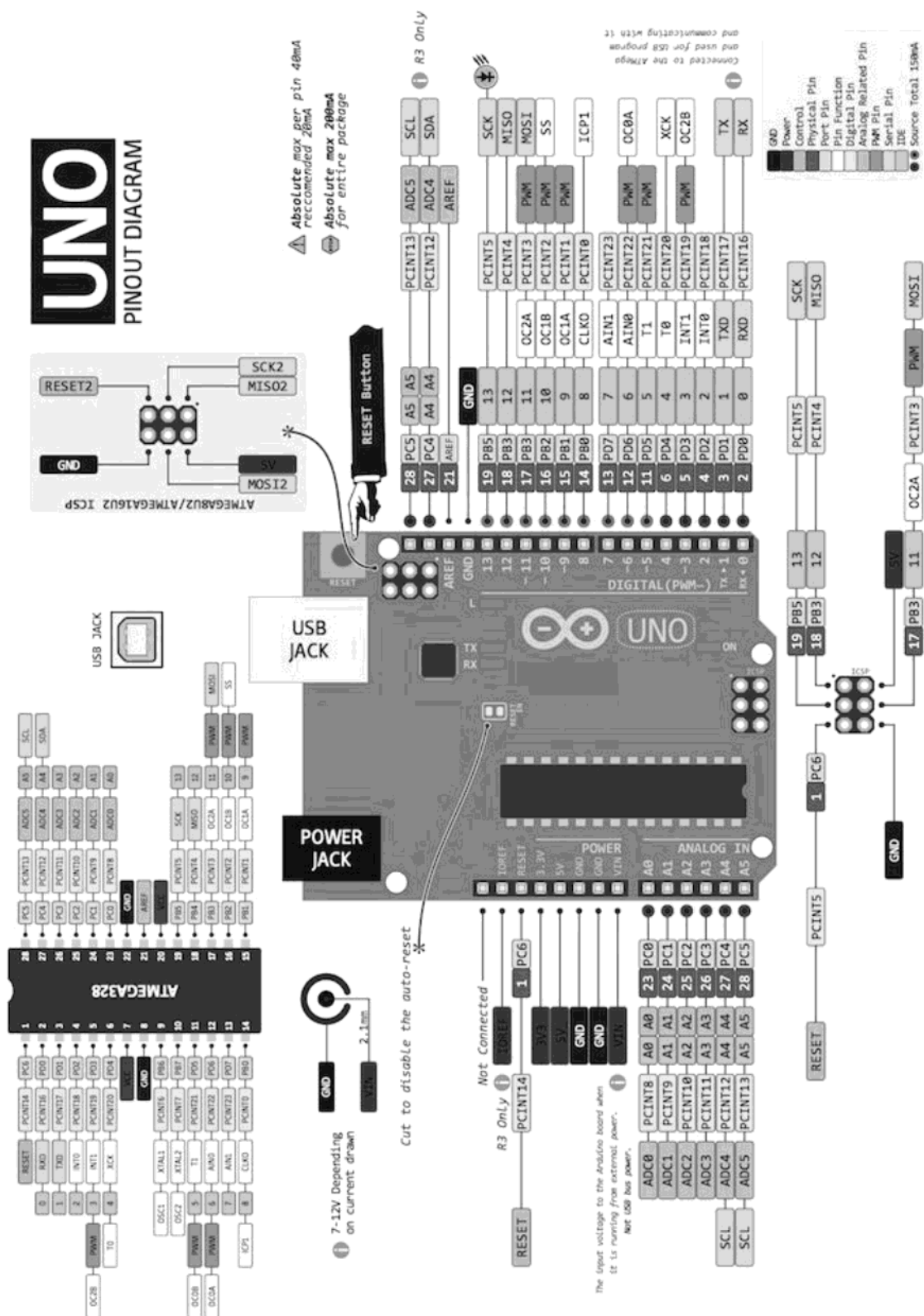


Рисунок 4.2 – Диаграмма выводов платы Arduino Uno

- битовые операторы;
- составные операторы.

Данные:

- константы;
- типы данных;
- преобразователи типов;
- область видимости переменной и спецификаторы;
- вспомогательная функция.

Функции:

- цифровой ввод/вывод;
- аналоговый ввод/вывод;
- расширенный ввод/вывод;
- время;
- математические вычисления;
- тригонометрия;
- случайные числа;
- биты и байты;
- внешние прерывания;
- прерывания.

Библиотеки:

- EEPROM;
- SD;
- SPI;
- SoftwareSerial;
- Wire;
- вспомогательные классы.

Более подробно описание операторов, данных (переменных и констант), функций и библиотек приведено в справке программного обеспечения Arduino IDE: Помощь -> Найти в Справочнике.

### 1.3 Инсталляция драйвера и программного обеспечения.

Для этого этапа потребуется кабель стандарта USB (с разъемами типа USB-A и USB-B).

Найдите последнюю версию на официальной странице (<https://www.arduino.cc/en/Main/Software>). После окончания загрузки распакуйте скачанный файл. Убедитесь, что не нарушена структура папок. Откройте папку двойным кликом на ней. В ней должны быть несколько файлов и подкаталогов.

Arduino Uno получает питание автоматически от любого USB-подключения к компьютеру или другому источнику питания. Подсоедините плату Arduino к компьютеру, используя USB-кабель. Должен загореться зеленый светодиод питания, помеченный PWR.

Установка драйверов для Arduino Uno на Windows7, Vista или XP:

- подключите плату и подождите, пока Windows начнет процесс установки драйвера. Через некоторое время процесс закончится безрезультатно;

- нажмите на кнопку ПУСК и откройте Панель управления. В панели управления перейдите на вкладку Система и безопасность (System and Security). Затем выберите Система. Когда откроется окно Система, выберите Диспетчер устройств (Device Manager);

- обратите внимание на порты (COM и LPT). Вы увидите открытый порт под названием «Arduino UNO (COMxx)»;

- щелкните на названии «Arduino UNO (COMxx)» правой кнопкой мышки и выберите опцию «Обновить драйвер» (Update Driver Software);

- кликните «Browse my computer for Driver software». Для завершения найдите и выберите файл драйвера для Uno – «ArduinoUNO.inf», расположенный в папке Drivers программного обеспечения для Arduino (не в подкаталоге «FTDI USB Drivers»). На этом Windows закончит установку драйвера.

#### 1.4 Изучение интерфейса программного обеспечения.

Запустить среду разработки Arduino, дважды щелкнув на приложении для Arduino.

Среда разработки Arduino состоит из встроенного текстового редактора программного кода, области сообщений, окна вывода текста (консоли), панели инструментов с кнопками часто используемых команд и нескольких меню (рисунок 4.3). Для загрузки программ и связи среда разработки подключается к аппаратной части Arduino.

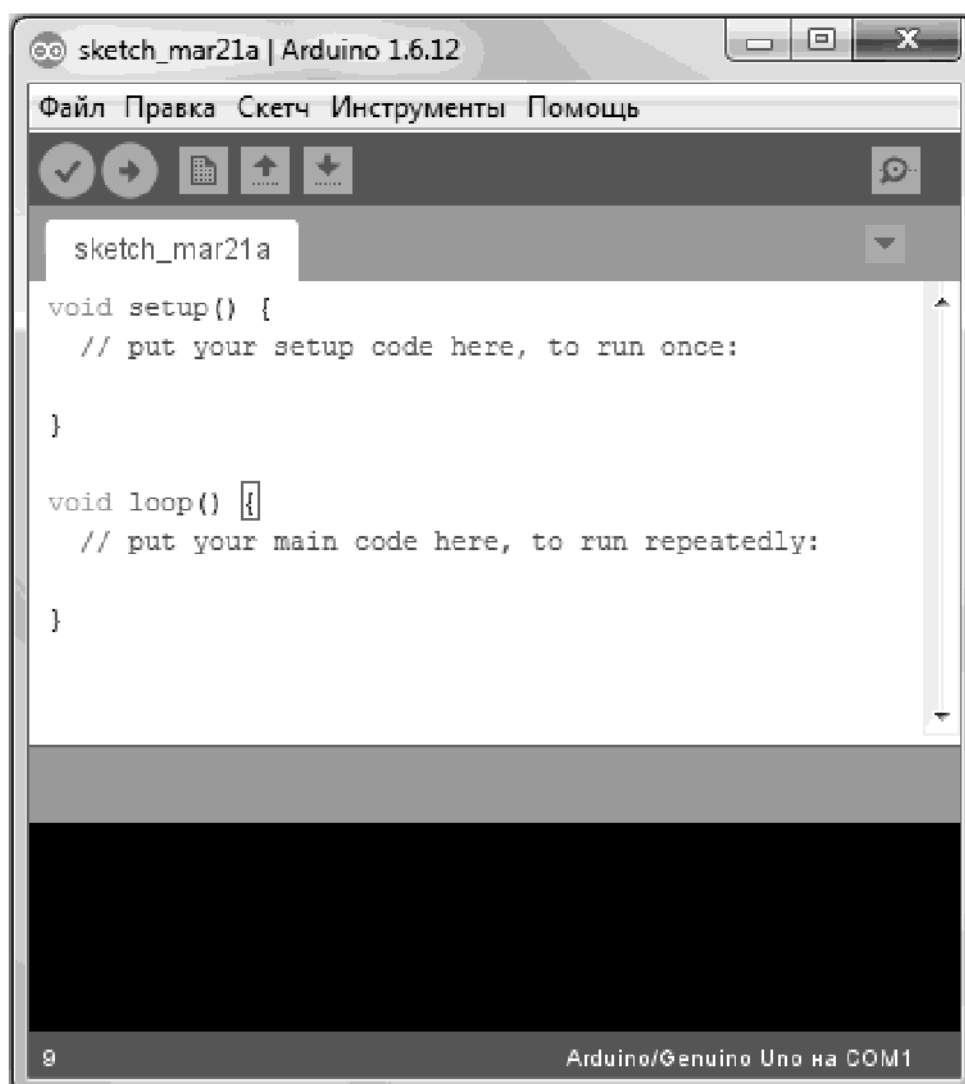


Рисунок 4.3 – Интерфейс программного обеспечения Arduino IDE

Программа, написанная в среде Arduino, называется скетч. Скетч пишется в текстовом редакторе, имеющем инструменты вырезки/вставки,

поиска/замены текста. Во время сохранения и экспорта проекта в области сообщений появляются пояснения, также могут отображаться возникшие ошибки. Окно вывода текста (консоль) показывает сообщения Arduino, включающие полные отчеты об ошибках и другую информацию. Кнопки панели инструментов позволяют проверить и записать программу, создать, открыть и сохранить скетч, открыть мониторинг последовательной шины (*Verify/Compile, Stop, New, Open, Save, Upload to I/O Board, Serial Monitor*).

Дополнительные команды сгруппированы в пять меню: Файл, Правка, Скетч, Инструменты, Помощь. Доступность меню определяется работой, выполняемой в данный момент.

Изучить указанные меню самостоятельно. В отчете представить описание пунктов меню.

## 2 Задание на выполнение работы.

2.1 Подключить плату Arduino к ПК и выполнить необходимые настройки аппаратного и программного обеспечения.

На первом этапе необходимо выбрать вид подключенной платы: Инструменты -> Плата: (рисунок 4.4).

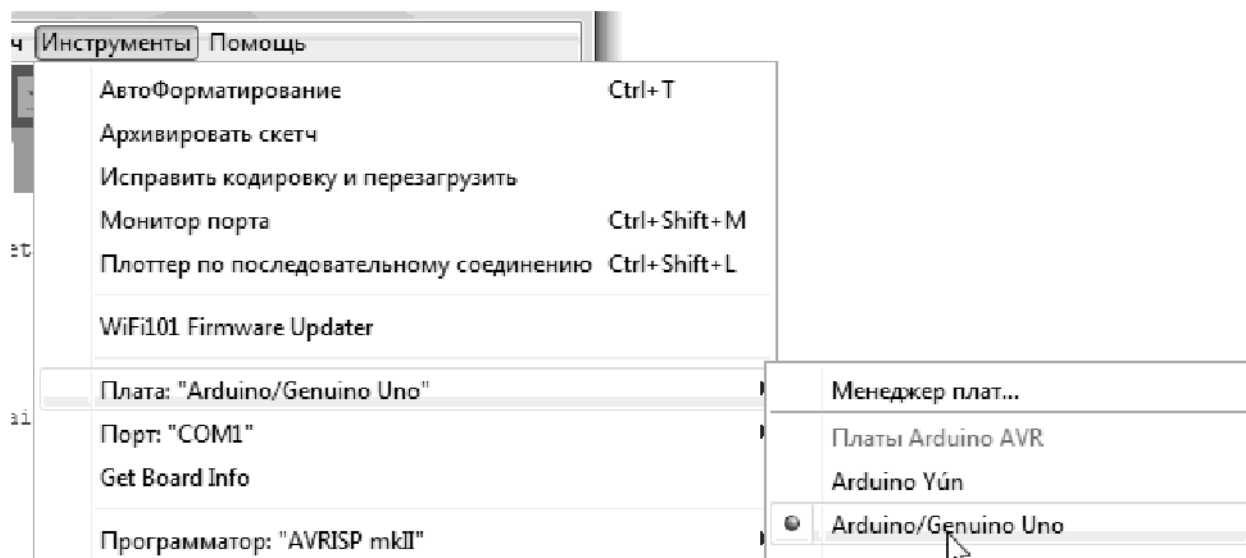


Рисунок 4.4 – Выбор подключенной платы

На следующем этапе необходимо выбрать порт, к которому

подключена плата: Инструменты -> Порт: (рисунок 4.5). Номер порта, указанного на рисунке выше, может быть любым и зависит от системы. На этом настройка аппаратного и программного обеспечения завершена.

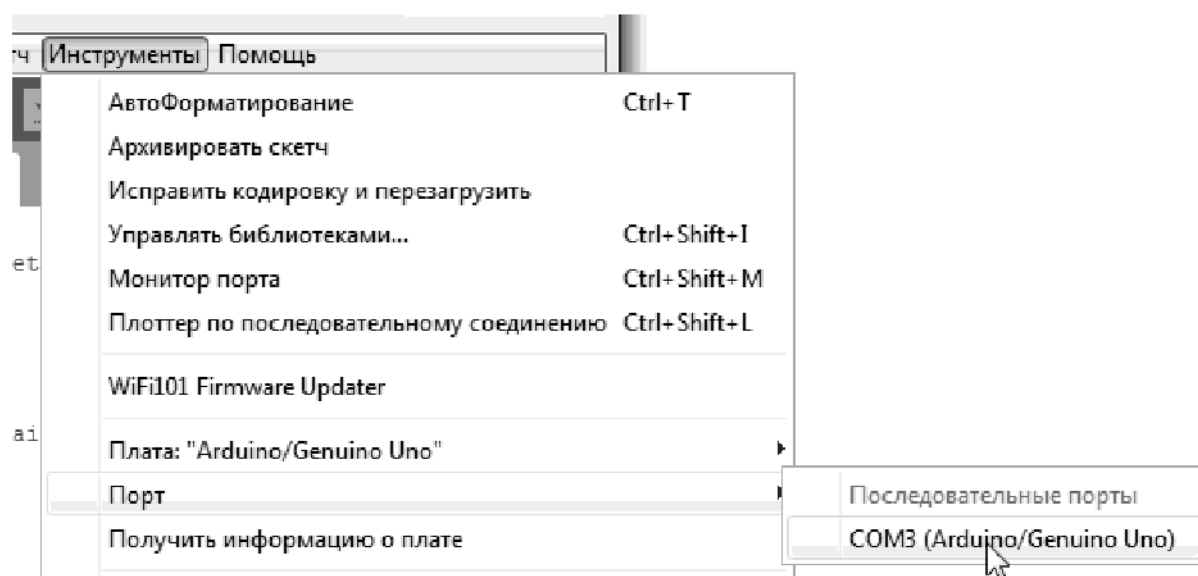


Рисунок 4.5 – Выбор порта, к которому подключена плата

2.2 Выполнить тестирование аппаратной и программной части с использованием стандартного скетча.

Arduino IDE содержит большой список готовых примеров. Для тестирования выберем самый распространенный – «Blink» (рисунок 4.6).

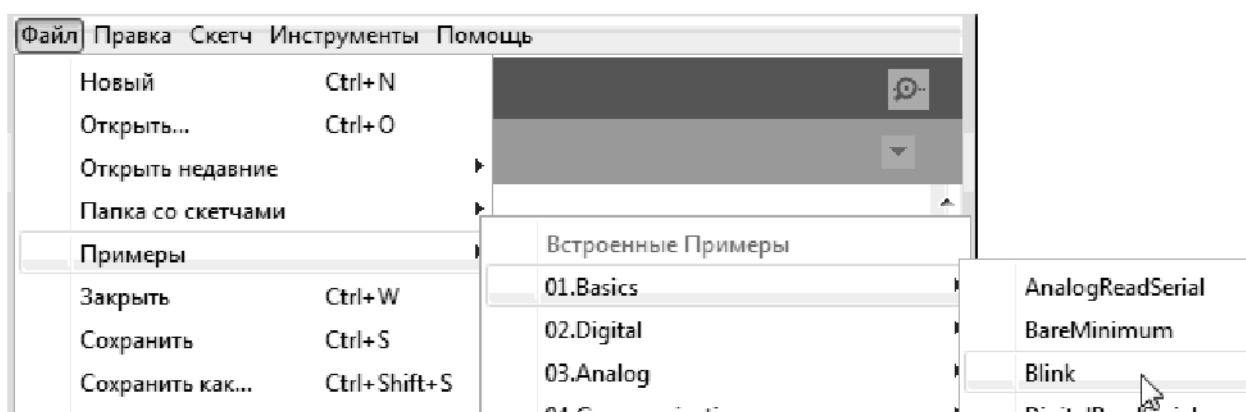


Рисунок 4.6 – Загрузка скетча «Blink»

Далее необходимо нажать кнопку «Загрузка». Во время загрузки в нижней части окна IDE появятся ползунок загрузки и сообщения. Вначале

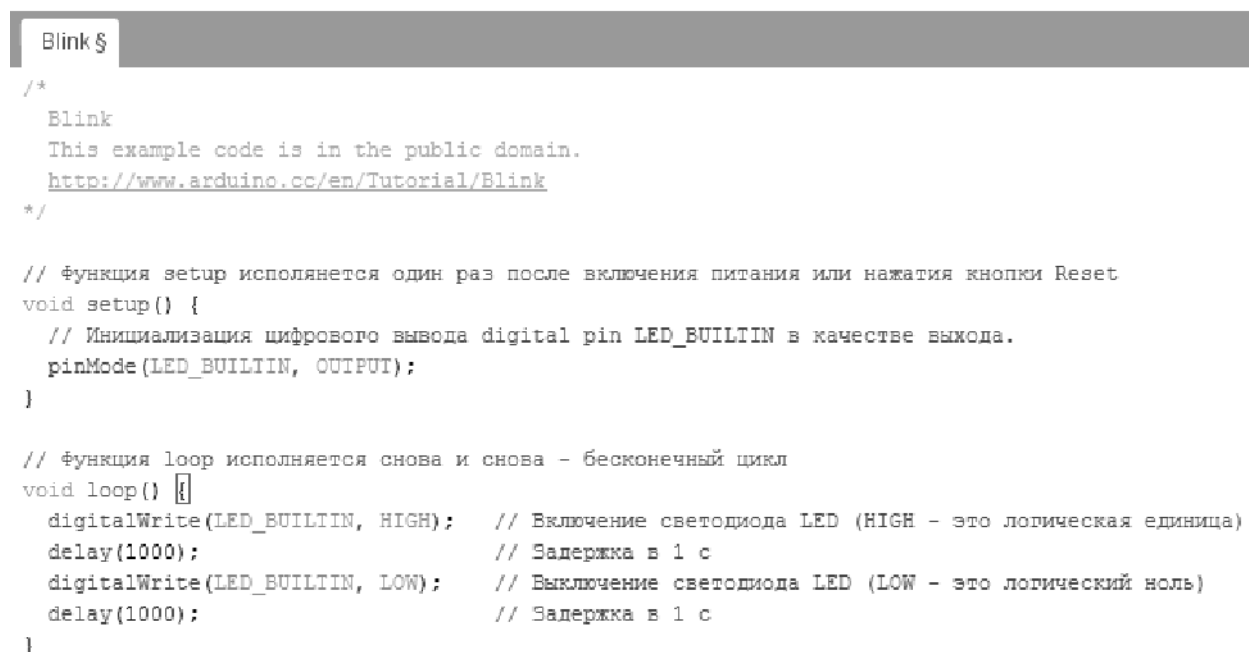
появляется фраза «Компилирование» («Compiling sketch»), что означает процесс конвертирования скетча в формат, необходимый для загрузки.

Дальше статус сменится на «Загрузка» («Uploading»). В этот момент светодиоды на плате начнут мигать, так как начнется записи скетча в микроконтроллер.

В конце статус сменится на «Загрузка завершена» («Done uploading»). В сообщении, которое появится в текстовой строке отобразится информация о том, что загруженный скетч занимает 1,084 байта из 32,256 доступных.

Если в скетче будут обнаружены ошибки, это будет отображено в нижней части окна. Необходимо устранить ошибки и попытаться записать скетч в плату снова. Рекомендуется перед прошивкой выполнять компиляцию скетча нажатием на кнопку «Проверить» («Verify»).

При успешной загрузке плата Arduino перезагрузится и «L» светодиод начнет мигать. На рисунке 4.7 приведен скрин скетча с подробными комментариями.



```
Blink §
/*
 Blink
 This example code is in the public domain.
 http://www.arduino.cc/en/Tutorial/Blink
*/

// функция setup исполняется один раз после включения питания или нажатия кнопки Reset
void setup() {
 // Инициализация цифрового вывода digital pin LED_BUILTIN в качестве выхода.
 pinMode(LED_BUILTIN, OUTPUT);
}

// функция loop выполняется снова и снова - бесконечный цикл
void loop() {
 digitalWrite(LED_BUILTIN, HIGH); // Включение светодиода LED (HIGH - это логическая единица)
 delay(1000); // Задержка в 1 с
 digitalWrite(LED_BUILTIN, LOW); // Выключение светодиода LED (LOW - это логический ноль)
 delay(1000); // Задержка в 1 с
}
```

Рисунок 4.7 – Скрин стандартного скетча «Blink»

Далее, во всех заданиях, необходимо в самом первом комментарии

указывать свою фамилию, имя и отчество. В качестве подтверждения выполнения задания необходимо в отчет приводить:

- скрин написанного скетча или его непосредственный текст;
- скрины результатов выполнения, если это возможно;
- фото результатов выполнения и собранной схемы (если требуется по заданию);
- текст всех скетчей (проекты) должны быть сохранены в программном обеспечении Arduino IDE для демонстрации при защите.

Изменить скетч примера «Blink» так, чтобы свечение светодиода длилось 0,5 с, а пауза 1,5 с. Выполнить загрузку и тестирование. Сохранить результаты выполнения как описано выше.

### 2.3 Написать программы для работы с простейшими устройствами.

#### 2.3.1 Исследование применения цифровых входов/выходов.

##### 2.3.1.1 Мигающий светодиод.

Для реализации примера необходимо на макетной плате собрать схему электрическую принципиальную (рисунок 4.8). Номера цифрового вывода, к которому подключен светодиод, соответствует номеру варианта. Сопротивление резистора – не менее 300 Ом.

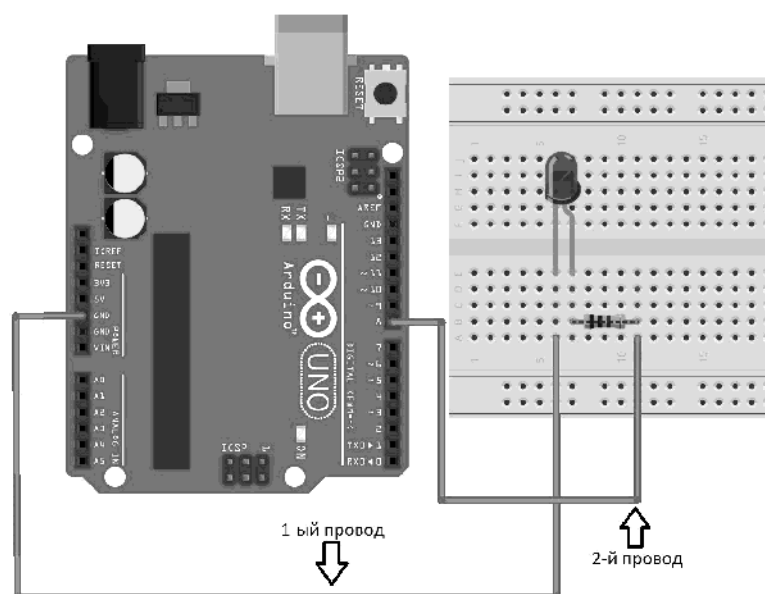


Рисунок 4.8

Написать скетч включения светодиода с миганием по алгоритму «0,5 с (светится); 0,5 с (не светится); n с (светится); n с (не светится); 1 с (светится); 1 с (не светится)» (где n – номер варианта).

Выполнить загрузку скетча и тестирование. Сохранить результаты выполнения как описано в п. 2.2.

#### 2.3.1.2 Кнопка.

Для реализации примера необходимо на макетной плате собрать схему электрическую принципиальную (рисунок 4.9). Номера цифрового вывода для подключения кнопки задает преподаватель, а номер, к которому подключен светодиод, соответствует номеру варианта. Сопротивление резистора в цепи кнопки – 10 кОм. Сопротивление резистора в цепи светодиода – не менее 300 Ом.

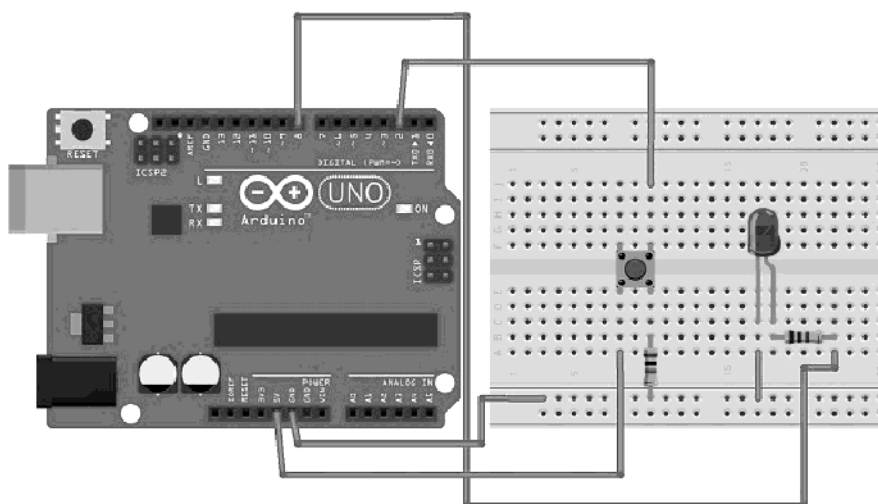


Рисунок 4.9

Написать скетч, чтобы светодиод светился, пока нажата кнопка, и выключался после ее отпускания.

Выполнить загрузку скетча и тестирование. Сохранить результаты выполнения как описано в п. 2.2.

Модифицировать скетч таким образом, чтобы светодиод светился при первом нажатии кнопки. После второго нажатия – светодиод должен выключиться. Далее – работа устройства повторяется.

Выполнить загрузку скетча и тестирование. Сохранить результаты выполнения как описано в п. 2.2.

В рассмотренных примерах очень велик шанс неверного срабатывания устройства, то есть не так, как задумано в алгоритме. Это происходит из-за того, что при нажатии кнопки возникает явление, – «дребезг» контактов. Переходный процесс переключения кнопки показан на рисунке 4.10.

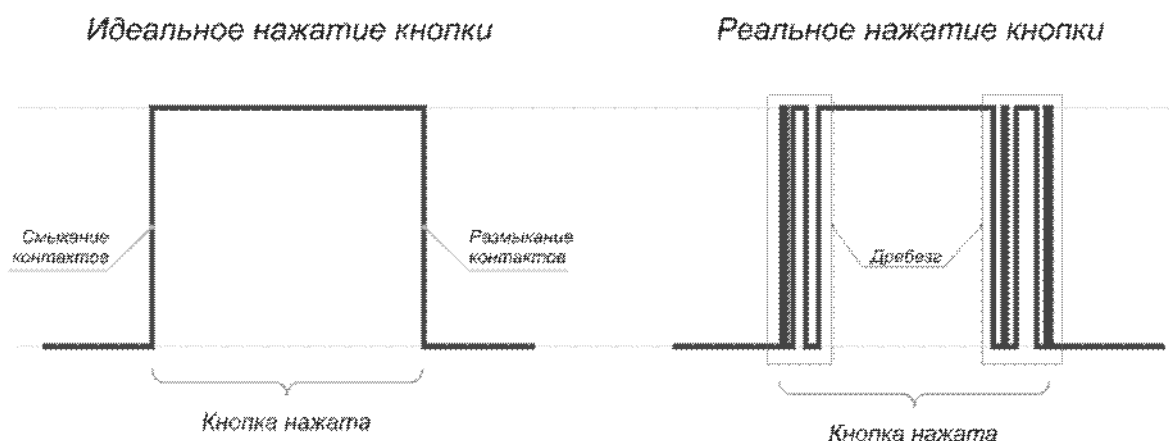


Рисунок 4.10 – Переходный процесс при нажатии кнопки

Переходные процессы протекают очень быстро и исчезают за доли – единицы миллисекунд. Поэтому мы редко их замечаем. Но, обрабатывая сигнал от кнопки в цифровых устройствах, мы вполне можем столкнуться с такими переходными эффектами и должны их учитывать при программировании. Учет сводится к борьбе с последствиями. Возможны два метода решения проблемы: аппаратный и программный.

Далее будет рассмотрен программный способ борьбы с «дребезгом» контактов с помощью функции `delay()` и функции `millis()`. Фрагмент программного кода с использованием функции `delay()` приведен на рисунке 4.11. В примере реализована пауза в 10 мс для ожидания завершения переходного процесса.

Используя алгоритм представленного фрагмента кода модернизировать первые два скетча п. 2.3.1.2. Выполнить загрузку скетча и тестирование.

Сохранить результаты выполнения как описано в п. 2.2.

```
int currentValue, prevValue;
void loop() {
 currentValue = digitalRead(PIN_BUTTON);
 if (currentValue != prevValue) {
 // Что-то изменилось, здесь возможна зона неопределенности
 // Делаем задержку
 delay(10);
 // А вот теперь спокойно считываем значение, считая, что нестабильность исчезла
 currentValue = digitalRead(PIN_BUTTON);
 }

 prevValue = currentValue;
 Serial.println(currentValue);
}
```

Рисунок 4.11 – Борьба с «дребезгом» контактов посредством функции delay()

Самостоятельно разработать программный код для борьбы с «дребезгом» контактов с помощью функции millis(). Используя алгоритм модернизировать первые два скетча п. 2.3.1.2. Выполнить загрузку скетча и тестирование. Сохранить результаты выполнения как описано в п. 2.2.

### 2.3.2 Исследование применения аналоговых входов.

#### 2.3.2.1 Измерение уровня освещенности.

Аналоговые входы могут только принимать сигнал и измерять напряжение поступающего сигнала. В изучаемой плате Arduino установлен 10-битный аналогово-цифровой преобразователь. Это означает, что 0 В считывается как значение «0», а 5 В считаются как значение «1023». То есть аналоговые входы измеряют, подаваемое на них напряжение, с точностью до 0,005 В.

Для измерения уровня освещенности будет использован фоторезистор. Для реализации примера необходимо на макетной плате собрать схему электрическую принципиальную (рисунок 4.12). Номер аналогового входа для подключения фоторезистора задает преподаватель. Сопротивление

резистора в цепи фоторезистора – 10 кОм.

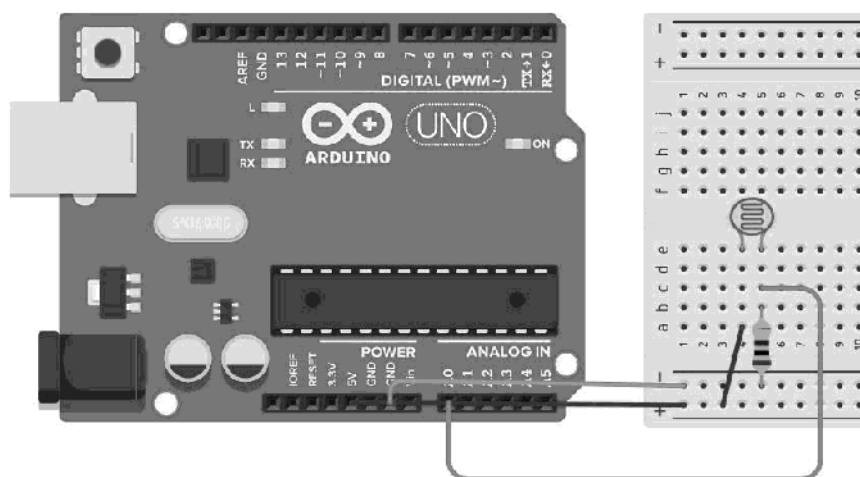


Рисунок 4.12

Написать скетч, который будет выводить в COM-порт значения с аналогового входа, пропорциональные уровню освещенности фоторезистора.

Выполнить загрузку скетча и тестирование. Сохранить результаты выполнения как описано в п. 2.2.

Модернизировать схему измерителя уровня освещенности, как показано на рисунке 4.13. Номера цифрового вывода, к которому подключен светодиод, соответствует номеру варианта. Сопротивление резистора – не менее 300 Ом.

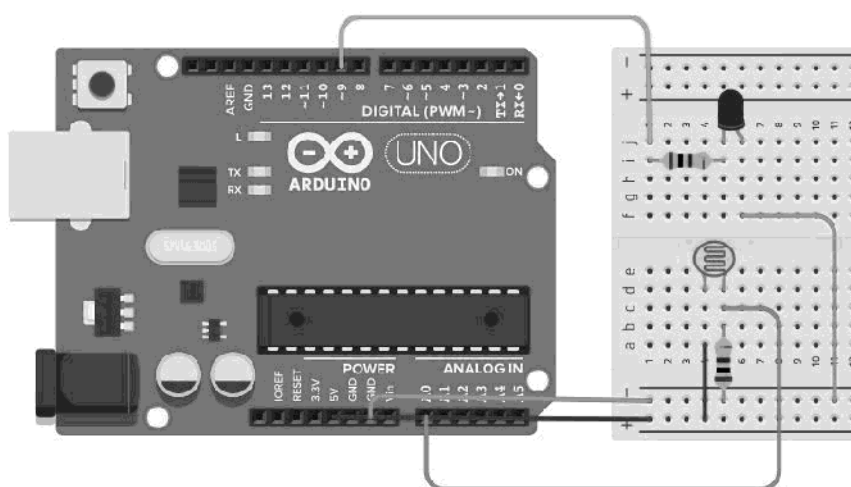


Рисунок 4.13

Написать скетч, который будет управлять яркостью свечения светодиода по следующему алгоритму:

- для четных номеров вариантов при максимальной освещенности фоторезистора светодиод должен быть выключен. При уменьшении освещенности – яркость свечения светодиода должна возрасти до максимального значения при полном затемнении фоторезистора;

- для нечетных номеров вариантов при максимальной освещенности фоторезистора светодиод должен светиться максимально ярко. При уменьшении освещенности – яркость свечения светодиода должна уменьшаться до состояния выключения при полном затемнении фоторезистора.

При этом необходимо учитывать, что разрядность ШИМ изучаемой платы Arduino составляет 8 бит (выходной диапазон от 0 до 255).

Выполнить загрузку скетча и тестирование. Сохранить результаты выполнения как описано в п. 2.2.

#### 2.3.2.2 Измерение напряжения на аналоговом входе.

Для измерения напряжения на аналоговом входе необходимо на макетной плате собрать схему электрическую принципиальную (рисунок 4.14). Номер аналогового входа для подключения переменного резистора задает преподаватель. Его сопротивление – 10 кОм.

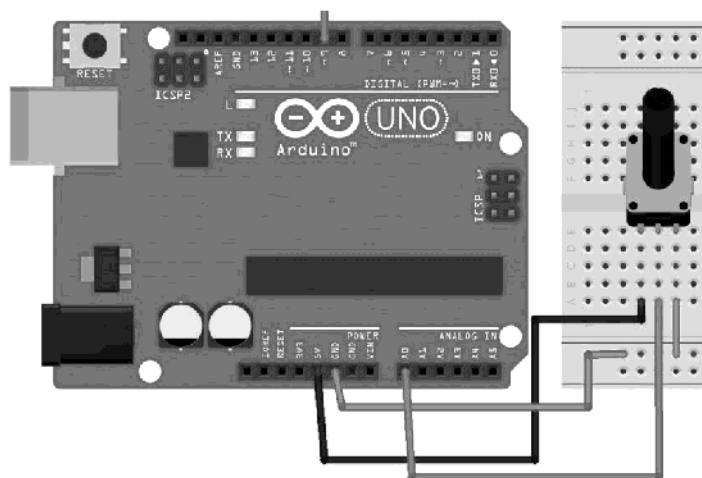


Рисунок 4.14

Написать скетч, который будет выводить в COM-порт значения напряжения на аналоговом входе, устанавливаемое поворотом ручки переменного резистора. При этом необходимо учесть, что в изучаемой плате Arduino установлен 10-битный аналогово-цифровой преобразователь. Это означает, что 0 В считывается как значение «0», а 5 В считываются как значение «1023». То есть аналоговые входы измеряют, подаваемое на них напряжение, с точностью до 0,005 В. Таким образом, для вывода в COM-порт именно значения напряжения необходимо пересчитать полученное с аналогового входа значение в напряжение. Формула должна быть введена в текст счетча.

Выполнить загрузку скетча и тестирование. Сохранить результаты выполнения как описано в п. 2.2.

### 2.3.3 Исследование реализации ШИМ (PWM).

С помощью цифровых выходов плат Arduino можно реализовать ШИМ-управление объектами. Далее реализуем простейший пример управления яркостью светодиода в зависимости от количества нажатий кнопки. Для реализации примера необходимо на макетной плате собрать схему электрическую принципиальную (рисунок 4.15).

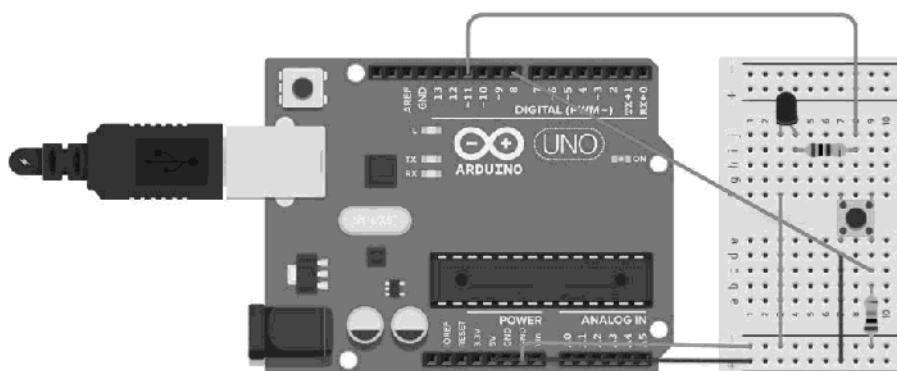


Рисунок 4.15

При каждом нажатии на кнопку яркость свечения светодиода будет изменяться.

Для четных номеров вариантов при включении яркость светодиода должна быть установлена максимальной. При каждом нажатии на кнопку яркость светодиода должна дискретно уменьшаться вплоть до полного выключения.

Для нечетных номеров вариантов при включении яркость светодиода должна быть установлена равной нулю. При каждом нажатии на кнопку яркость светодиода должна дискретно увеличиваться вплоть до максимальной.

При этом необходимо учитывать, что разрядность ШИМ изучаемой платы Arduino составляет 8 бит (выходной диапазон от 0 до 255).

Написать скетч. Выполнить загрузку скетча и тестирование. Сохранить результаты выполнения как описано в п. 2.2.

#### 2.3.4 Исследование метода управления двигателем постоянного тока.

Выводы Arduino, сконфигурированные как цифровой выход могут отдавать до 20 мА в нагрузку при напряжении до 5 В. Для управления нагрузками с более высокими значениями токов и/или напряжений необходимо использование транзистора в ключевом режиме. В данном эксперименте будем управлять скоростью вращения мотора изменением напряжения, подаваемого на транзистор с помощью ШИМ. Для выполнения эксперимента необходимо на макетной плате собрать схему электрическую принципиальную (рисунок 4.16). Номер выхода ШИМ для подключения мотора задает преподаватель.

База транзистора подключена к плате Arduino через резистор сопротивлением 1 кОм. Параллельно двигателю необходимо подключить защитный диод.

Написать скетч, работающий по следующему алгоритму:

- при включении питания (нажатии кнопки Reset) двигатель не вращается в течении 5 с;
- далее он начинает вращаться со скоростью, пропорциональной

максимальной скорости вращения деленной на  $8 \cdot n$  (где  $n$  – номер варианта), в течении 5 с;

- далее он начинает вращаться со скоростью, пропорциональной максимальной скорости вращения деленной на  $3 \cdot n$  (где  $n$  – номер варианта), в течении 5 с;

- далее он начинает вращаться со скоростью, пропорциональной максимальной скорости вращения деленной на  $n$  (где  $n$  – номер варианта), в течении 5 с;

- далее он начинает вращаться с максимальной скоростью в течении 5 с, после чего останавливается на 10 с.

Алгоритм повторяется циклически.

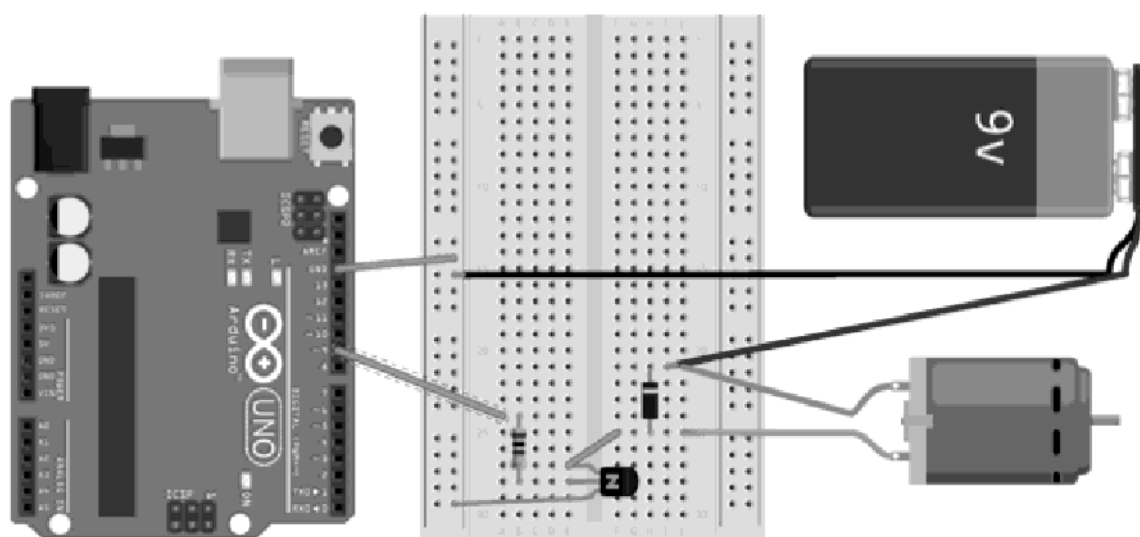


Рисунок 4.16

Выполнить загрузку скетча и тестирование. Сохранить результаты выполнения как описано в п. 2.2.

### 2.3.5 Исследование метода управления сервоприводом.

Сервопривод – это мотор, положением вала которого можно управлять. От обычного мотора он отличается тем, что ему можно точно в градусах задать положение, в которое встанет вал. Сервоприводы используются для моделирования различных механических движений роботов.

В отличие от обычных двигателей постоянного тока, сервоприводы обычно помимо выводов питания (Vcc и GND) имеют еще один дополнительный вывод (сигнальный). Сигнальный вывод используется для подачи управляющего сигнала от микроконтроллера, поворачивая тем самым его вал на необходимый угол.

Для выполнения эксперимента необходимо на макетной плате собрать схему электрическую принципиальную (рисунок 4.17). Номер выхода ШИМ для подключения сервопривода задает преподаватель.

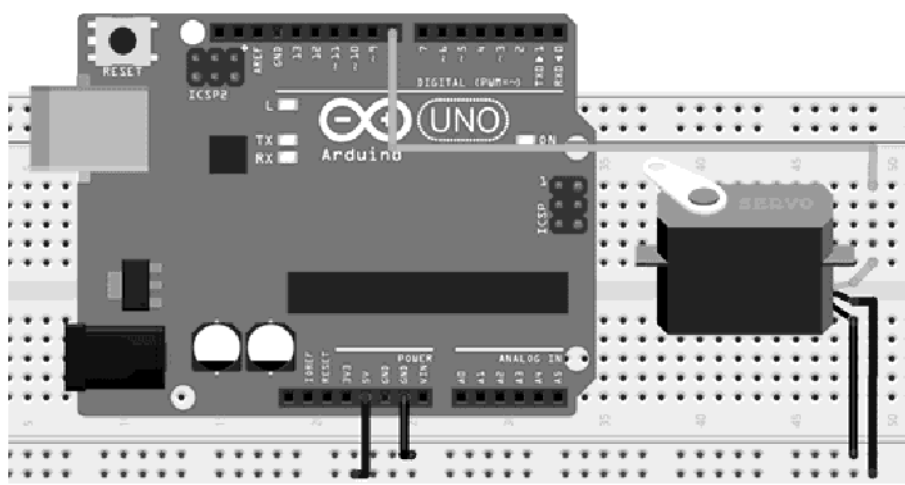


Рисунок 4.17

Скетч для этого проекта довольно прост благодаря функциональной и компактной библиотеке `Servo.h`, разработанной командой Ардуино. Библиотека упрощает поворот вала сервопривода под разными углами с помощью всего одной команды. Библиотека поставляется с предустановленной Arduino IDE. В начале проекта необходимо подключить библиотеку командой `#include <Servo.h>`.

Изучив самостоятельно работу библиотеки `Servo.h`, написать скетч, работающий по следующему алгоритму:

- при включении питания (нажатии кнопки Reset) вал сервопривода должен установиться в нулевое положение;
- через 3 с вал должен повернуться на угол 170 градусов;

- каждые 3 с вал должен поворачиваться обратно к нулевой позиции с шагом в 10 градусов – для каждого четного номера варианта, и с шагом 15 градусов – для каждого нечетного варианта, с паузой между поворотами в 3 с.

Выполнить загрузку скетча и тестирование. Сохранить результаты выполнения как описано в п. 2.2.

2.3.6 Исследование метода вывода символов на 7-сегментный индикатор.

Для реализации примера необходимо на макетной плате собрать схему электрическую принципиальную (рисунок 4.18). Сопротивление резисторов – не менее 300 Ом. Перед подключением необходимо определить тип индикатора: с общим катодом или с общим анодом. На рисунке приведен пример для индикатора с общим катодом. В случае индикатора с общим анодом, его необходимо соединить с гнездом +5V платы Arduino.

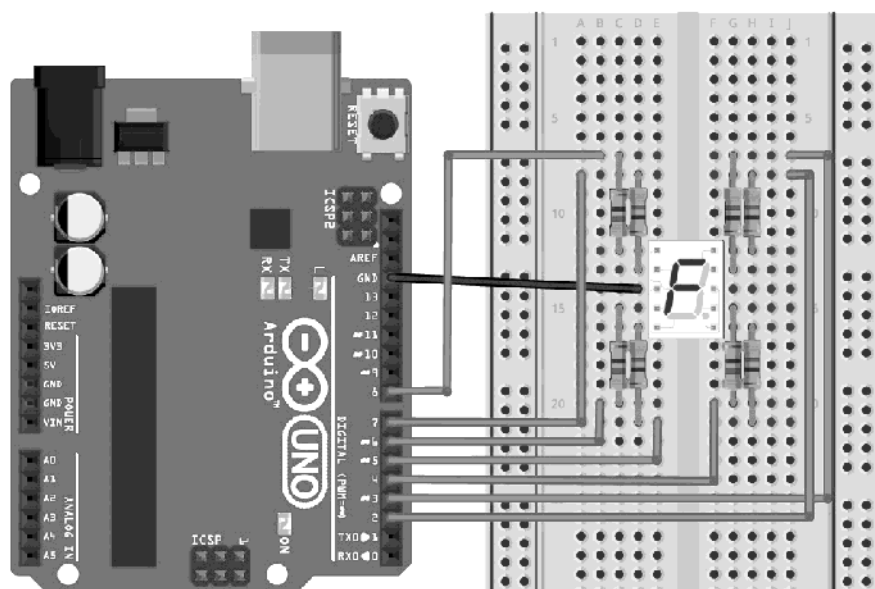


Рисунок 4.18

Написать скетч для вывода на индикатор номера варианта в 16-й системе счисления. Если номер варианта превышает максимальное число для данного индикатора, то необходимо из числа 20 вычесть номер своего

варианта и полученную разность в 16-й системе счисления выводить на индикатор.

Выполнить загрузку скетча и тестирование. Сохранить результаты выполнения как описано в п. 2.2.

Модернизировать скетч таким образом, чтобы показания индикатора мигали с частотой 1 Гц.

Выполнить загрузку скетча и тестирование. Сохранить результаты выполнения как описано в п. 2.2.

### 3 Задание на обработку данных.

3.1 Сделать выводы о работе цифровых входов/выходов.

3.2 Сделать выводы о работе аналоговых входов.

3.3 Сделать выводы о работе PWM-выводов (ШИМ) платы Arduino.

3.4 Сделать выводы об управлении двигателем постоянного тока.

3.5 Сделать выводы об управлении сервоприводом.

3.6 Сделать выводы о выводе информации на светодиодный индикатор.

3.7 Сделать выводы о полученных результатах работы написанных программ.

## **Содержание отчета**

Отчет по лабораторной работе выполняется на листах формат А4 согласно требований ГОСТ и ЕСКД и должен содержать следующие пункты:

а) титульный лист с названием работы, дисциплины, кода группы, ф.и.о. исполнителя и преподавателя;

б) цель и содержание работы;

в) краткое описание аппаратной части и программного обеспечения, необходимого для выполнения работы;

г) написанные программы приведенных заданий (в виде скринов или текста). Комментарии обязательны;

- д) результаты исследований работы скетчей (фото и/или скрины);
- е) выводы по результатам работы.

### **Контрольные вопросы**

1.

### **Список литературы, рекомендованный к использованию по данной теме**

#### **Основная учебная литература**

#### **Дополнительная литература**

## ПРАКТИЧЕСКАЯ РАБОТА 5

**Тема:** Программирование электронных узлов и устройств на базе отладочной платформы Arduino.

**Цель работы:** закрепление знаний, полученных на лекциях по разделам 2 (лекции 2, 3 и 4) и 4 (лекция 8). В ходе выполнения лабораторной работы необходимо:

- произвести обоснованный выбор элементной базы для реализации электронного устройства;
- собрать схему электрическую принципиальную посредством макетирования;
- написать программное обеспечение для электронных узлов и блоков согласно вариантам. Выполнить загрузку и, в случае необходимости, отладку.

**Формируемые компетенции:** ОПК-9.

### Теоретическая часть

Arduino применяется для создания электронных устройств с возможностью приема сигналов от различных цифровых и аналоговых датчиков, которые могут быть подключены к нему, и управления различными исполнительными устройствами. Проекты устройств, основанные на Arduino, могут работать самостоятельно или взаимодействовать с программным обеспечением на компьютере.

Выполнение электронных устройств на платформе Arduino позволяет:

- быстро протестировать схемотехническое решение разработанного устройства и в случае наличия ошибок оперативно их исправить;
- быстро выполнить тестирование и отладку алгоритма работы электронного устройства и в случае наличия ошибок оперативно их исправить.

Далее, разработанный проект может быть выполнен на отдельной

печатной плате, включающей микроконтроллер платформы Arduino, и необходимую периферию.

Именно поэтому важным является изучение методики программирования электронных узлов и устройств на базе отладочной платформы Arduino.

### **Оборудование и материалы**

Лабораторная работа выполняется с использованием аппаратной платформы Arduino, программной оболочки Arduino IDE и дополнительных модулей: датчиков и объектов управления (исполнительных устройств).

### **Указания по технике безопасности**

Перед проведением лабораторной работы следует изучить инструкцию по технике безопасности:

- при работе с аппаратной платформой Arduino IDE;
- при работе на компьютере.

**ВНИМАНИЕ!** Все работы по сборке схемы, изменению порядка внешних соединений, подключению измерительных приборов должны производиться ТОЛЬКО при отключенном напряжении источника питания и/или отключенном кабеле USB, соединяющем плату с ПК.

Включение напряжения питания необходимо производить только после проверки преподавателем правильности коммутаций.

При подключении измерительных приборов необходимо внимательно и заранее устанавливать пределы измерения, во избежание повреждения приборов.

Также следует соблюдать правила и требования, установленные для данного компьютерного класса.

## Задания

1 Задание на подготовку к выполнению работы.

1.1 Выполнить сборку схемы электронного устройства.

Произвести обоснованный выбор элементной базы для реализации электронного устройства. Тип устройства выбирается в соответствии с номером варианта, заданным преподавателем. Исходные данные приведены в таблице 5.1.

Произвести обоснованный выбор портов ввода/вывода платы Arduino.

Подключить необходимые датчики (элементы регулировки и т. д.) и/или объекты управления (устройства индикации и т. д.) непосредственно к плате Arduino или с использованием дополнительной макетной платы (Shield).

Включение напряжения питания и подключение к ПК необходимо производить только после проверки преподавателем правильности реализации схемного решения и произведенных коммутаций.

Таблица 5.1

| Вариант | Устройство                        | Рекомендации                                                                                                                                                                                                                                                                                                                      |
|---------|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | Измеритель температуры            | Цифровой датчик температуры<br>Символьный дисплей 16x2 (в первой строке выводится надпись «Temperature = xx», где xx – числовое значение измеренной температуры. Во второй строке надпись «Normal», когда значение температуры от 0 до 35 градусов Цельсия включительно; «Attention», когда температура выше 35 градусов Цельсия) |
| 2       | Электронный измеритель расстояния | Ультразвуковой датчик<br>Символьный дисплей 16x2 (в первой строке выводится надпись «Distance = xx cm», где xx – числовое значение измеренного расстояния в                                                                                                                                                                       |

|   |                                      |                                                                                                                                                                                                                                                                                                     |
|---|--------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|   |                                      | сантиметрах. Во второй строке надпись «Normal» для расстояния до объекта от 10 до 20 см, «Close» – для расстояния менее 10 см и «Away» – для расстояния более 20 см)                                                                                                                                |
| 3 | Клавиатура управления объектом       | Клавиатура 4x4<br>Символьный дисплей (в первой строке выводится «Code = xxxx», где xxxx – код нажатой кнопки; во второй строке «Symbol = x», где x символ, соответствующий нажатой кнопке                                                                                                           |
| 4 | Система дистанционного управления ИК | ИК-пульт<br>ИК-приемник<br>Графический дисплей (в первой строке выводится «Code = xxxx», где xxxx – код нажатой на пульте кнопки; во второй строке «Symbol = x», где x символ, соответствующий нажатой на пульте кнопке                                                                             |
| 5 | Таймер обратного отсчета             | Сенсорный датчик<br>4-разрядный семисегментный индикатор<br>Счет от 80 до 0. Каждый новый отсчет начинается при касании сенсорного датчика                                                                                                                                                          |
| 6 | Счетчик посещений                    | Инфракрасный датчик препятствий<br>4-разрядный семисегментный индикатор                                                                                                                                                                                                                             |
| 7 | Электронная метеостанция             | Цифровой датчик температуры<br>Цифровой датчик влажности<br>Графический дисплей (в первой строке выводится надпись «Temperature = xx °C», где xx – числовое значение измеренной температуры. Во второй строке выводится надпись «Humidity = yy %», где yy – числовое значение измеренной влажности) |
| 8 | Охранное устройство                  | Модуль датчика вибрации<br>Сенсорный датчик<br>Графический дисплей (в первой строке выводится надпись «OFF». При установлении факта вибрации в эту строку выводится надпись «Vibration».                                                                                                            |

|    |                                                                     |                                                                                                                                                                                                                                                                                                                                                                          |
|----|---------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|    |                                                                     | При касании сенсорного датчика снова выводится надпись «OFF» до момента установления факта следующей вибрации)                                                                                                                                                                                                                                                           |
| 9  | Цифровой индикатор напряжения                                       | Подстроечный резистор<br>Шкала линейная светодиодная<br>При минимальном измеряемом напряжении светится только 1 разряд шкалы, при максимальном – все                                                                                                                                                                                                                     |
| 10 | Измеритель температуры                                              | Цифровой датчик температуры<br>Графический дисплей (в первой строке выводится надпись «Temperature = xx», где xx – числовое значение измеренной температуры. Во второй строке надпись «Normal», когда значение температуры от 0 до 32 градусов Цельсия включительно; «Attention», когда температура выше 32 градусов Цельсия)                                            |
| 11 | Устройство управления скоростью вращения двигателя постоянного тока | Двигатель постоянного тока<br>Подстроечный резистор<br>При минимальном напряжении регулировки – двигатель не вращается, при максимальном – вращается с максимальной скоростью. Напряжение регулировки считывается со среднего вывода резистора (диапазон от 0 до 5 В)                                                                                                    |
| 12 | Устройство управления сервоприводом                                 | Сервопривод с фиксированным углом<br>Энкодер инкрементный<br>При перемещении ручки энкодера на по часовой стрелке сервопривод поворачивается также по часовой стрелке на такое же количество фиксированных шагов, на сколько шагов повернули ручку энкодера. при вращении против часовой стрелки алгоритм работы такой же (сервопривод вращается против часовой стрелки) |
| 13 | Устройство ограничения доступа                                      | Метка RFID<br>Приемник RFID<br>Графический дисплей (в первой строке                                                                                                                                                                                                                                                                                                      |

|    |                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|    |                                   | выводится «UID = x...x», где x...x – идентификационный номер метки; во второй строке «Access is allowed», если UID совпадает с установленным в скетче и «Access is denied», если UID не совпадает с установленным в скетче                                                                                                                                                                                                                                                                                                                                         |
| 14 | Электронный измеритель расстояния | Ультразвуковой датчик<br>Символьный дисплей 16x2 (в первой строке выводится надпись «Distance = xx cm», где xx – числовое значение измеренного расстояния в сантиметрах. Во второй строке надпись «Normal» для расстояния до объекта от 10 до 20 см, «Close» – для расстояния менее 10 см и «Away» – для расстояния более 20 см)                                                                                                                                                                                                                                   |
| 15 | Электронный замок                 | Метка RFID<br>Приемник RFID<br>Сервопривод с фиксированным углом<br>Если UID совпадает с установленным в скетче, то сервопривод поворачивается на угол 90 градусов, Через 5 секунд вал сервопривода возвращается в первоначальное (нулевое) положение. Если UID не совпадает, то вращения вала нет                                                                                                                                                                                                                                                                 |
| 16 | Ультразвуковой датчик парковки    | Ультразвуковой датчик<br>Звукоизлучатель со встроенным генератором частоты<br>При расстоянии до датчика более 60 см, звуковой сигнал не формируется. При расстоянии от 40 до 60 см, звуковой сигнал формируется с частотой 0,25 Гц (время звучания равно времени паузы). При расстоянии от 20 до 40 см, звуковой сигнал формируется с частотой 1 Гц (время звучания равно времени паузы). При расстоянии от 10 до 20 см, звуковой сигнал формируется с частотой 2 Гц (время звучания равно времени паузы). При расстоянии менее 10 см, звуковой сигнал формируется |

|    |                                                           |                                                                                                                                                                                                                                                                                                                                                                         |
|----|-----------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|    |                                                           | постоянно.                                                                                                                                                                                                                                                                                                                                                              |
| 17 | Устройство автоматической регулировки освещенности уровня | Фоторезистор<br>Светодиод<br>При полном освещении фоторезистора светодиод выключен; при максимальном затемнении – светодиод светит максимально ярко; при разной освещенности фоторезистора яркость свечения светодиода должна быть разной: чем больше затемнение фоторезистора, тем ярче должен светиться светодиод                                                     |
| 18 | Система дистанционного управления ИК                      | ИК-пульт<br>ИК-приемник<br>Символьный дисплей (в первой строке выводится «Code = xxxx», где xxxx – код нажатой на пульте кнопки; во второй строке «Symbol = x», где x символ, соответствующий нажатой на пульте кнопке                                                                                                                                                  |
| 19 | Устройство распознавания цвета                            | Датчик цвета<br>Графический дисплей (в первой строке выводится «Red = xx», где xx – значение красного цвета; во второй строке «Green = yy», где yy – значение зеленого цвета; «Blue = zz», где zz – значение синего цвета; в четвертой строке выводится название «Red», «Green» или «Blue», когда определяемый цвет максимально приближен к одному из указанных цветов) |
| 20 | Клавиатура управления объектом                            | Клавиатура 3x4<br>Графический дисплей (в первой строке выводится «Code = xxxx», где xxxx – код нажатой кнопки; во второй строке «Symbol = x», где x символ, соответствующий нажатой кнопке)                                                                                                                                                                             |

1.2 Написать программное обеспечение для исследования электронных узлов и блоков, согласно заданию. Во всех вариантах, где используется

механическая коммутация контактов (клавиатура, переключатели, энкодер и т. д. необходимо предусмотреть процедуру подавления «дребезга» контактов).

При этом допустимо использовать встроенные библиотеки среды разработки Arduino IDE и специализированные библиотеки датчиков и/или объектов управления. Установка специализированной библиотеки может быть произведена двумя способами: средствами Arduino IDE или вручную.

Установка библиотеки средствами Arduino IDE:

- если библиотека не скачана, то ее необходимо скачать;
- войти в меню: Скетч > Подключить библиотеку > Добавить .ZIP библиотеку... (рисунок 5.1);

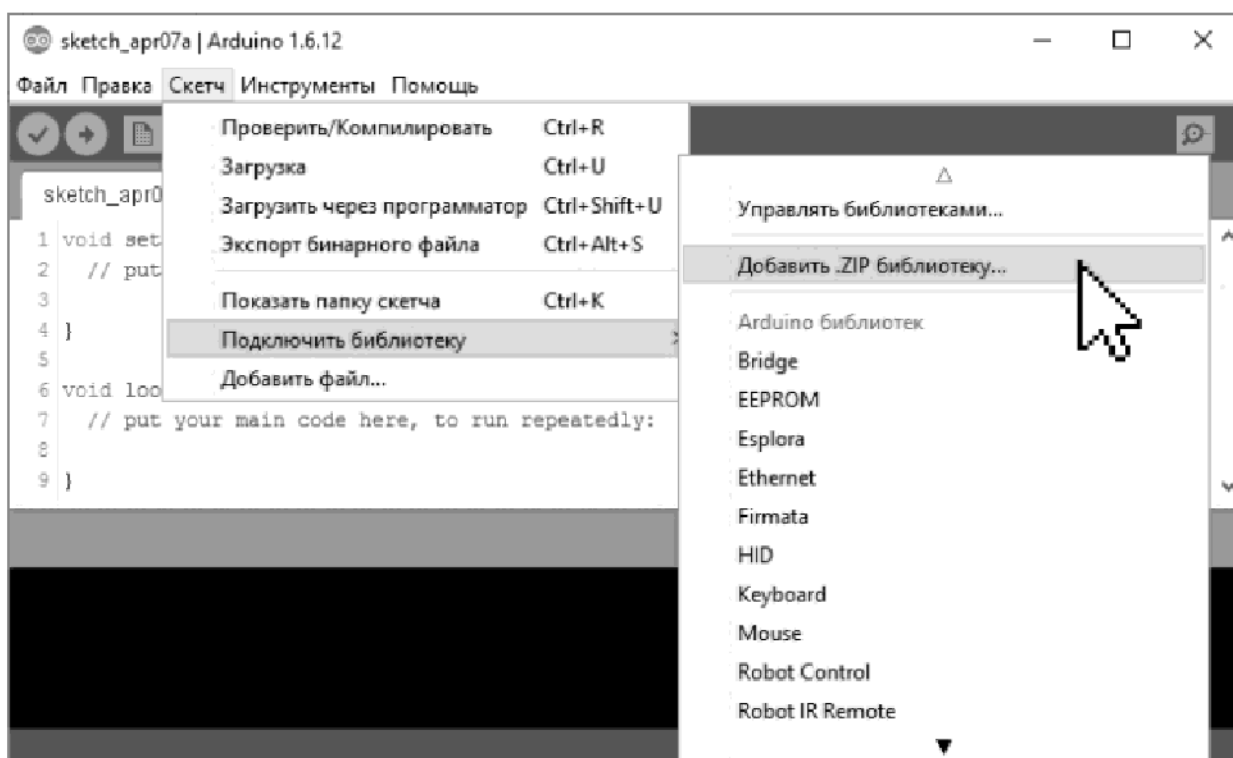


Рисунок 5.1

- в появившемся окне нажать на иконку «Компьютер» и выбрать папку, куда был скачан ZIP архива библиотеки;
- выбрать ZIP файл библиотеки. Название файла может не соответствовать названию библиотеки. После чего нажать на кнопку

«Открыть» (Open) (рисунок 5.2). На этом установка библиотеки завершена, можно приступить к её подключению в скетч.

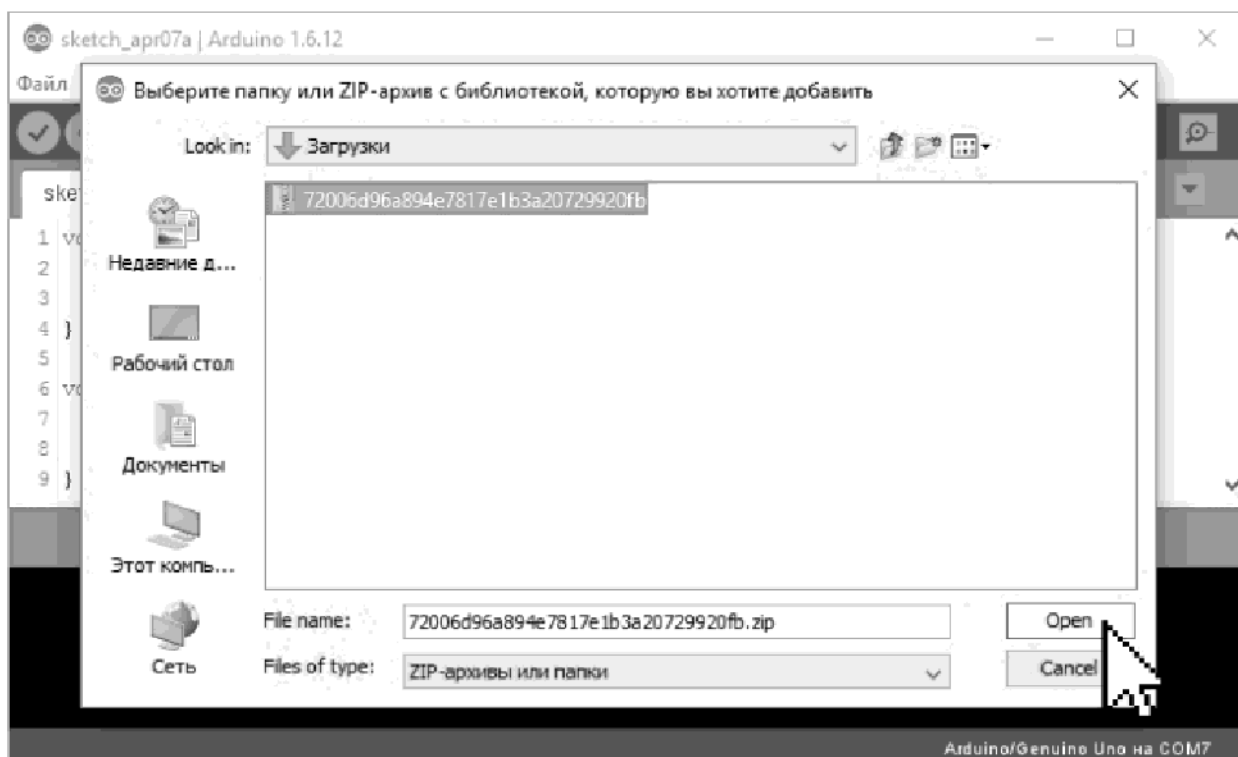


Рисунок 5.2

Установка библиотеки вручную необходимо распаковать скаченный ZIP архив и поместить в папку (имя папки обычно совпадает с названием библиотеки) из данного архива в папку: Компьютер > Документы > Arduino > Libraries. Если во время копирования Arduino IDE была запущена (открыта), то нужно закрыть все окна этой программы, после чего запустить (открыть) Arduino IDE и можно приступать к подключению библиотеки в скетч.

Для того чтобы подключить одну библиотеку, нужно написать одну строку в начале скетча: `"#include <файл.h>"`, например:

```
LiquidCrystal_I2C lcd(0x27,20,4); // lcd это объект библиотеки LiquidCrystal_I2C
```

2 Задание на выполнение работы.

2.1 Подключить плату Arduino к ПК и выполнить необходимые

настройки аппаратного и программного обеспечения (см. п. 2.1 лабораторной работы № 4).

2.2 Выполнить загрузку подготовленного ранее скетча и тестирование.

Если в скетче будут обнаружены ошибки, это будет отображено в нижней части окна. Необходимо устранить ошибки и попытаться записать скетч в плату снова. Рекомендуется перед прошивкой выполнять компиляцию скетча нажатием на кнопку «Проверить» («Verify»).

При успешной загрузке плата Arduino перезагрузится и «L» светодиод начнет мигать.

В качестве подтверждения выполнения задания необходимо в отчет приводить:

- скрин написанного скетча или его непосредственный текст;
- скрины результатов выполнения, если это возможно;
- фото результатов выполнения и собранной схемы (если требуется по заданию);
- текст скетча (проект) должен быть сохранен в программном обеспечении Arduino IDE для демонстрации при защите.

3 Задание на обработку данных.

3.1 Сделать выводы о работе датчиков (элементов регулировки и т. д. в зависимости от варианта задания).

3.2 Сделать выводы о работе объектов управления (устройств индикации и т. д. в зависимости от варианта задания).

3.3 Сделать выводы о полученных результатах работы написанной программы и функционировании электронного устройства в целом.

### **Содержание отчета**

Отчет по лабораторной работе выполняется на листах формат А4 согласно требований ГОСТ и ЕСКД и должен содержать следующие пункты:

- а) титульный лист с названием работы, дисциплины, кода группы, ф.и.о. исполнителя и преподавателя;
- б) цель и содержание работы;
- в) обоснование выбора компонентов для реализации устройства и портов ввода/вывода платы Arduino;
- г) схему электрическую принципиальную и фото электронного устройства;
- д) написанную программу для электронного устройства (в виде скринов или текста). Комментарии обязательны;
- е) результаты исследования работы скетча (фото и/или скрины);
- ж) выводы по результатам работы.

### **Контрольные вопросы**

1.

**Список литературы, рекомендованный к использованию по данной теме**

#### **Основная учебная литература**

#### **Дополнительная литература**

## ПРАКТИЧЕСКАЯ РАБОТА 6

**Тема:** Изучение основ программирования в визуальном графическом редакторе STM32CubeMX и интегрированной среде разработки Keil uVision на базе отладочной платформы STM32.

**Цель работы:** закрепление знаний, полученных на лекциях по разделам 3 (лекции 5 и 6) и 4 (лекция 8). В ходе выполнения лабораторной работы необходимо:

- ознакомиться с возможностями визуального графического редактора STM32CubeMX и интегрированной среды разработки Keil uVision;
- изучить аппаратные средства отладочной платформы STM32;
- выполнить тестирование аппаратной и программной части с использованием контрольного примера;
- создать проект согласно вариантам задания и проверить его работоспособность.

**Формируемые компетенции:** ОПК-9.

### Теоретическая часть

Микроконтроллер STM32 – популярная и очень востребованная платформа, позволяющая создавать профессиональные решения для автоматизации в самых различных областях.

STM32 – это платформа, в основе которой лежат микроконтроллеры STMicroelectronics на базе ARM процессора, различные модули и периферия, а также программные решения (IDE) для работы с железом. Решения на базе STM активно используются благодаря производительности микроконтроллера, его удачной архитектуре, малом энергопотреблении, небольшой цене. В настоящее время STM32 состоит уже из нескольких линеек для самых разных предназначений.

Для ARM архитектуры разработано множество сред разработки. К самым известным относятся инструменты фирм Keil и IAR System.

Программы этих компаний предлагают самые продвинутые инструментарии для оптимизации кода. Также дополнительно существуют различные системы – USB стеки, TCP/IP-стеки и прочие. Применяя системы Keil, пользователь получает хороший уровень технической поддержки.

Для упрощения и автоматизации настройки инициализации, а также для генерации базовых проектов используется визуальный редактор конфигурации и генератора кода STM32CubeMX.

Благодаря своим техническим характеристикам и тому факту, что ПО для работы с микроконтроллерами является доступным и бесплатным, многие современные микроконтроллерные устройства строят на базе семейства STM32.

Именно поэтому важным является изучение и получение навыков программирования в визуальном графическом редакторе STM32CubeMX и интегрированной среде разработки Keil uVision на базе отладочной платформы STM32.

### **Оборудование и материалы**

Лабораторная работа выполняется с использованием визуального графического редактора STM32CubeMX, интегрированной среды разработки Keil uVision и аппаратных средств отладочной платформы STM32.

### **Указания по технике безопасности**

Перед проведением лабораторной работы следует изучить инструкцию по технике безопасности:

- при работе с отладочной платформой STM32;
- при работе на компьютере.

**ВНИМАНИЕ!** Если в варианте задания используются внешние по отношению к отладочной платформе устройства, то все работы по сборке схемы, изменению порядка внешних соединений, подключению измерительных приборов должны производиться **ТОЛЬКО** при отключенном напряжении источника питания и/или отключенном кабеле USB, соединяющем платформу с ПК.

Включение напряжения питания необходимо производить только после проверки преподавателем правильности коммутаций.

При подключении внешних устройств **НЕОБХОДИМО** проверять уровни напряжений питания и сигналов во избежание повреждения внешних устройств и отладочной платформы.

Также следует соблюдать правила и требования, установленные для данного компьютерного класса.

## **Задания**

1 Задание на подготовку к выполнению работы.

1.1 Изучение аппаратных средств отладочной платформы STM32.

Отладочная плата из серии STM32 Nucleo-64 (рисунок 6.1) обеспечивает доступный и гибкий способ для пользователей попробовать новые концепции и создать прототипы с помощью микроконтроллера STM32, выбрав из различных сочетаний производительности, энергопотребления и функций. Поддержка интерфейса Arduino Uno V3 и разъем ST morpho позволяют легко расширять функциональность открытой платформы разработки Nucleo.

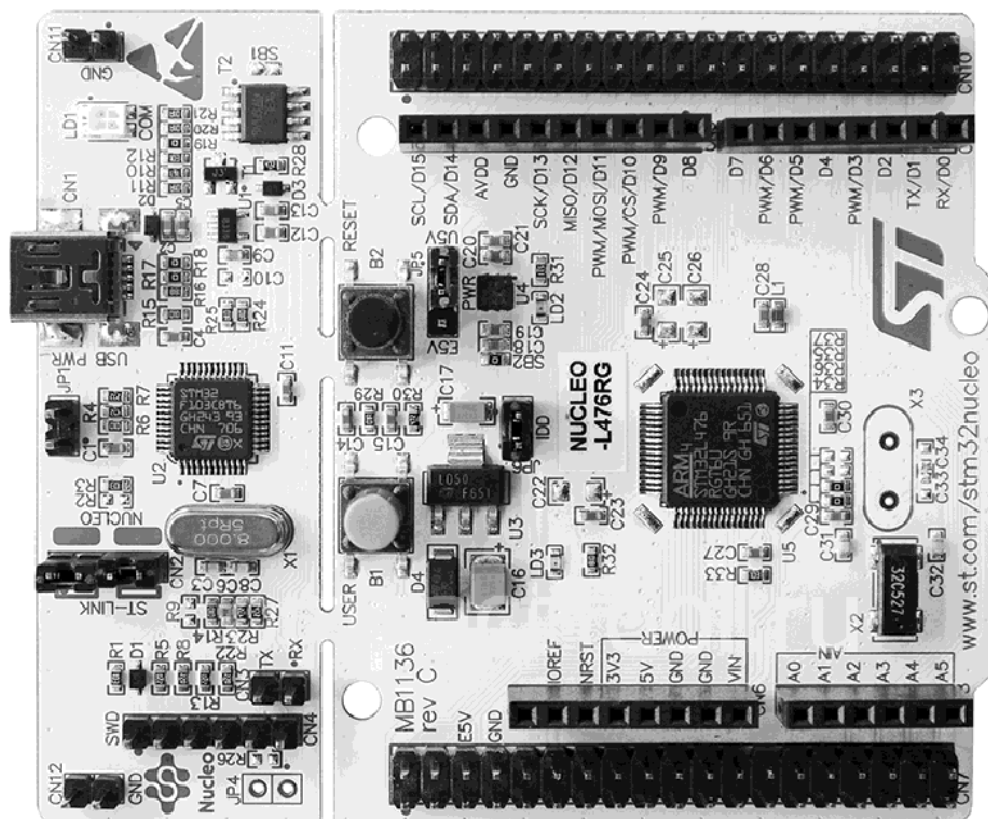


Рисунок 6.1

Особенности:

- микроконтроллер STM32L476RGT6, 32 бит, 80 МГц, ARM Cortex-M4, Flash 1 Мбайт, RAM 128 Кбайт;
- два типа расширения: разъем Arduino Uno V3 и штыревой разъем ST morpho для полного доступа I/O микроконтроллера;
- доступен ресурс ARM mbed - (<http://mbed.org>);
- встроенный программатор/отладчик ST-LINK/V2-1;
- питание платы: через USB шину или от внешнего источника (3,3 В, 5 В, в диапазоне от 7 до 12 В);
- три светодиода: USB коммуникации (LD1), пользовательский (LD2), наличие питания (LD3);
- две кнопки: пользовательская и сброс;
- USB ST-LINK с нумерацией и тремя интерфейсами: виртуальный

COM порт, устройство хранения (Mass storage), порт отладки;

- поддержка широкого выбора интегрированных сред разработки (IDE), включая IAR, ARM Keil, IDE на основе GCC.

1.2 Изучение возможностей визуального графического редактора STM32CubeMX и интегрированной среды разработки Keil uVision. Тестирование аппаратной и программной части с использованием контрольного примера.

1.2.1 Визуальный редактор конфигурации и генератора кода STM32CubeMX, предлагаемый STMicroelectronics, позволяет осуществлять тонкую автоматизированную настройку инициализации и генерировать базовые проекты.

Основными функциями, которые предоставляет визуальный редактор конфигурации, являются:

- настройка использования выводов периферийными устройствами в составе микроконтроллера;
- настройка тактирования;
- настройка периферийных устройств;
- расчет потребления энергии в соответствии с планируемыми режимами работы.

Для начала работы в редакторе необходимо запустить исполняемый файл генератора инициализационного кода CubeMX с помощью ярлыка. Для создания нового проекта необходимо выбрать пункт меню New Project (рисунок 6.2).

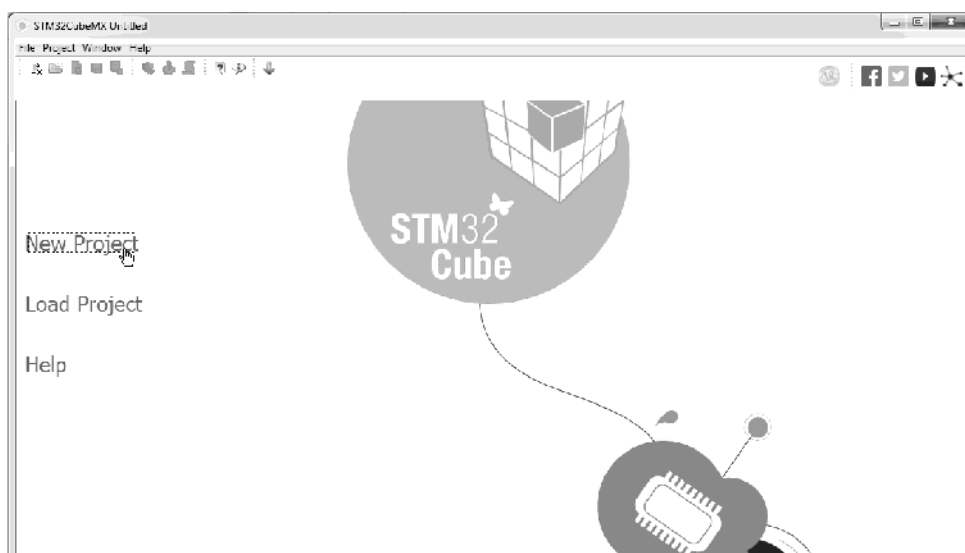


Рисунок 6.2

На вкладке MCU Selector есть возможность найти микроконтроллер, используемый в проекте, либо при помощи параметрического поиска подобрать тот, который оптимально подойдет под конкретные задачи. Для работы с отладочной платой данной работы необходимо перейти на вкладку Board Selector (рисунок 6.3) и выбрать указанные на рисунке пункты.

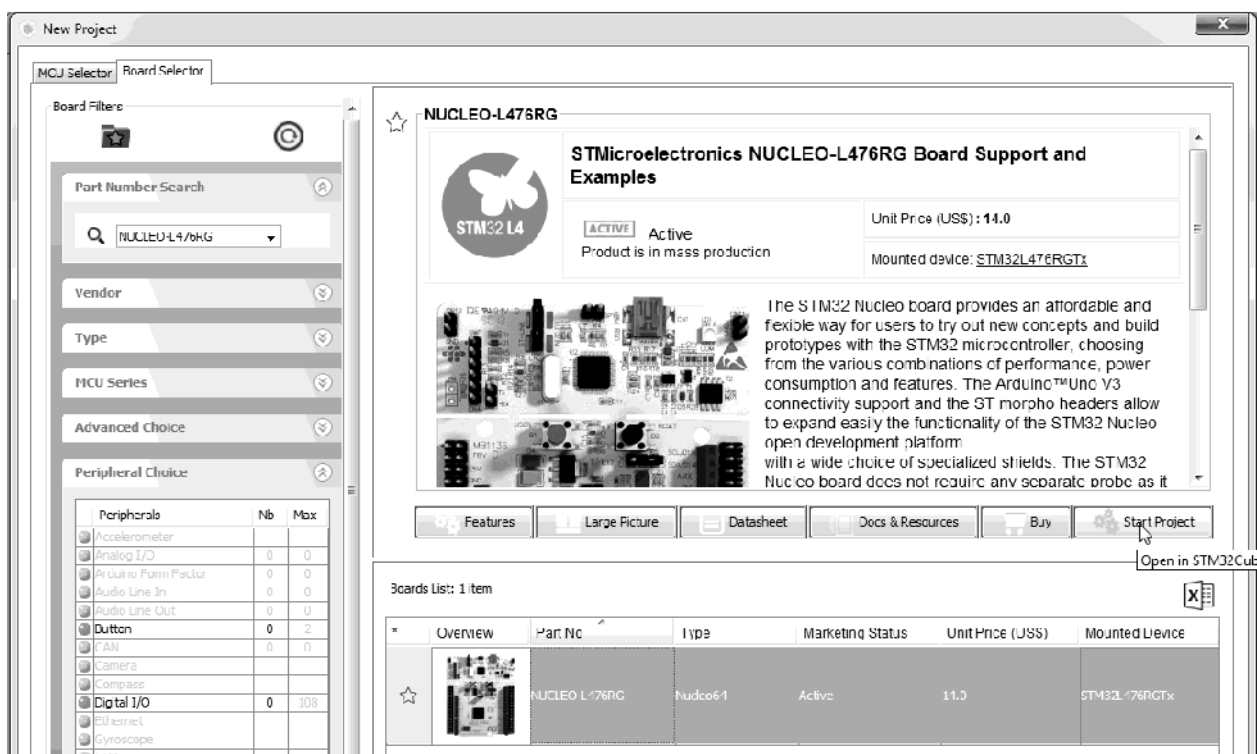


Рисунок 6.3

Настройка функционального назначения выводов производится после выбора микроконтроллера или отладочной платы во вкладке Pinout (рисунок 6.4).

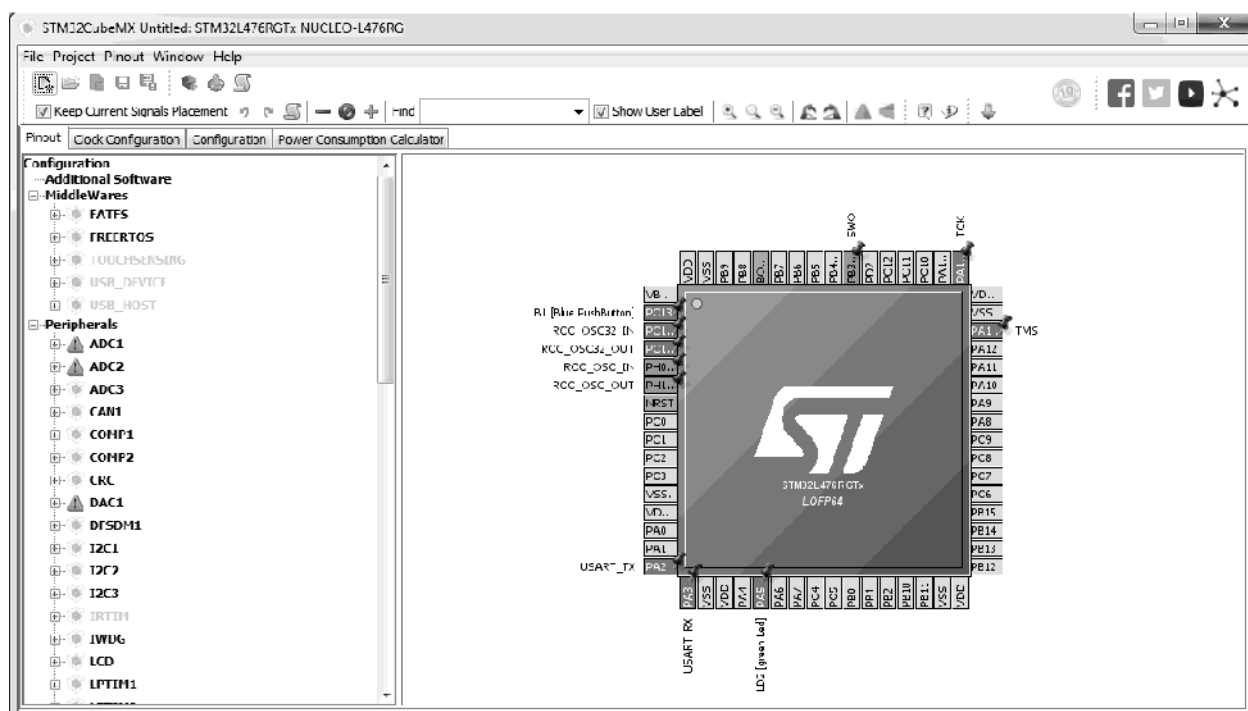


Рисунок 6.4

Рабочая область вкладки разделена на две части: в первой отображается дерево конфигурации периферийных модулей и специализированных программных компонентов, а во второй организовано графическое представление МК и использования его выводов. Дерево конфигурации позволяет не только выбрать, какие модули будут использоваться, но и, например, выбрать режим использования, дополнительные настройки и так далее. По мере выбора используемых компонентов в части отображения будут происходить изменения относительно используемых выводов. Также будет происходить автоматическая проверка того, какие модули не могут использоваться или какие возможности могут быть ограничены в связи с произведенными настройками.

Кроме того, функциональное назначение выводов может быть

произведено и с помощью визуального представления. Для этого необходимо левой клавишей мыши нажать на вывод, после чего появится подсказка о возможных функциях вывода в составе периферийных устройств. Выбор одной из функций приводит к тому, что соответствующее устройство в дереве конфигурации становится активным.

Для контрольного примера ничего не надо менять. То есть мы работаем с заготовкой, которая уже настроена: UART, светодиод, кнопка и некоторые дополнительные функции, которые уже есть на плате (рисунок 6.5).

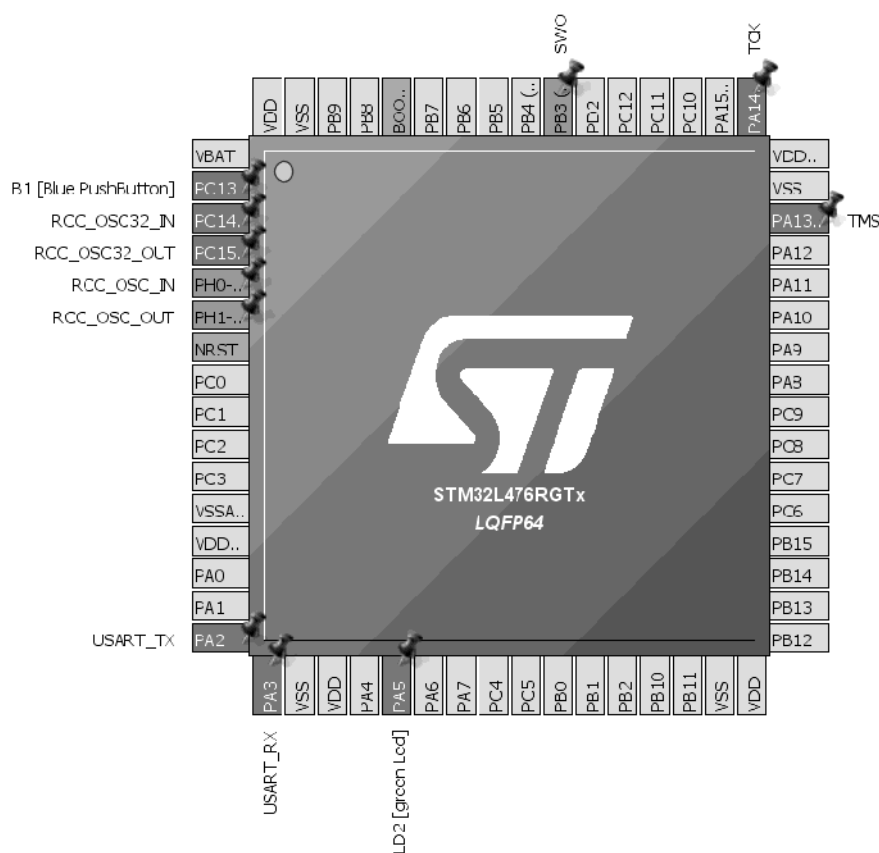


Рисунок 6.5

Вкладка настроек тактирования (Clock Configuration) (рисунок 6.6) дает возможность отслеживать связи между различными составными компонентами системы тактирования и контролировать все этапы данного процесса. При нарушении требований для выбранного микроконтроллера, а также других требований относительно настройки рабочих частот модулей, в

[illegible]

Помимо уже рассмотренной настройки периферии в дереве конфигурации на первой вкладке, где производится базовая настройка и, в основном, используются параметры, влияющие на использование выводов, присутствует также отдельная вкладка Configuration (рисунок 6.7), предназначенная для этих целей. По своей структурной организации она аналогична первой: присутствуют дерево конфигурации используемых

модулей, а также модулей, которые не требуют использования выводов МК, и часть с визуальными компонентами, которые предоставляют доступ к более детальной настройке системы. Визуальные компоненты сгруппированы в соответствии с функциями устройства для настроек (системные, функции управления, обмена данными и другие).

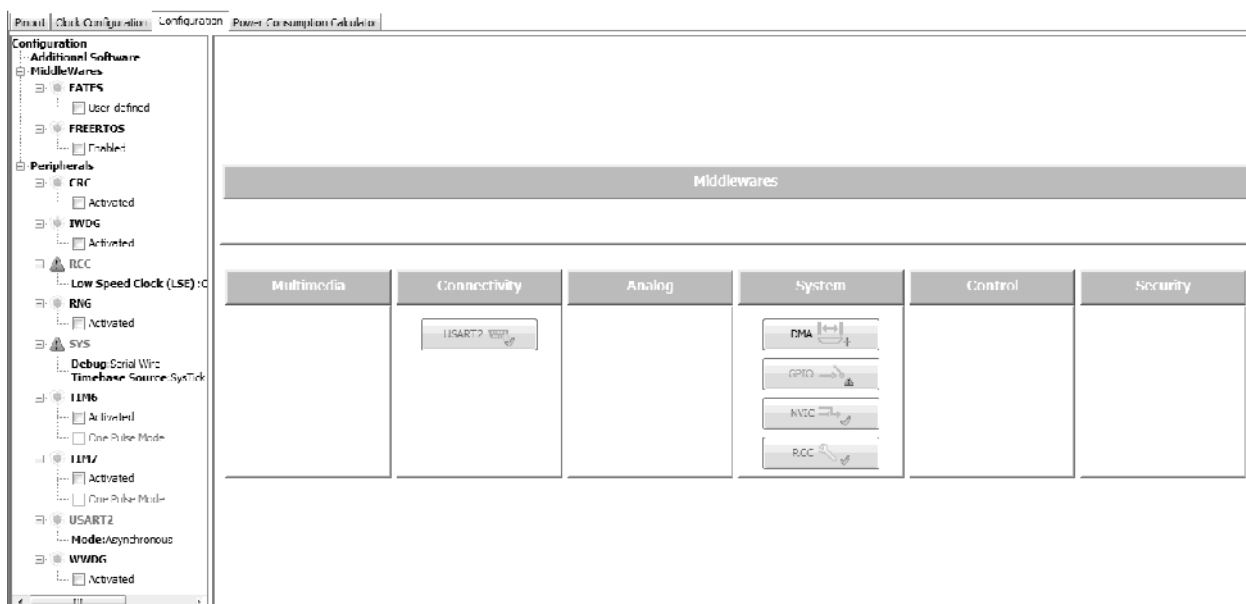


Рисунок 6.7

Нажмите на кнопку GPIO (General Purpose Input/Output, порты ввода/вывода общего назначения). У нас выбрано два вывода (рисунок 6.8). К выводу PA5 мы еще на первой вкладке подключили светодиод, а к выводу PC13 кнопку. Здесь они настроены на выход и вход соответственно, а в этом меню можно произвести их более тонкую настройку. Для контрольного примера ничего не надо менять.

Вкладка Power Consumption предназначена для задания последовательности режимов, в которых будет работать устройство, а также выбрать задействованные периферийные устройства для каждого режима. Для расчета времени жизни устройства пользователь может создать свой тип батарейки или выбрать ее в выпадающем списке слева. Таким образом, можно создать полный профайл устройства, сформировать отчет и сохранить

его в одном из известных форматов для отчета. Инструмент очень удобен и позволяет быстро оценить показатели автономности разрабатываемой системы, что особенно важно в разработках с акцентом на энергосбережении. Для контрольного примера ничего задавать не нужно.

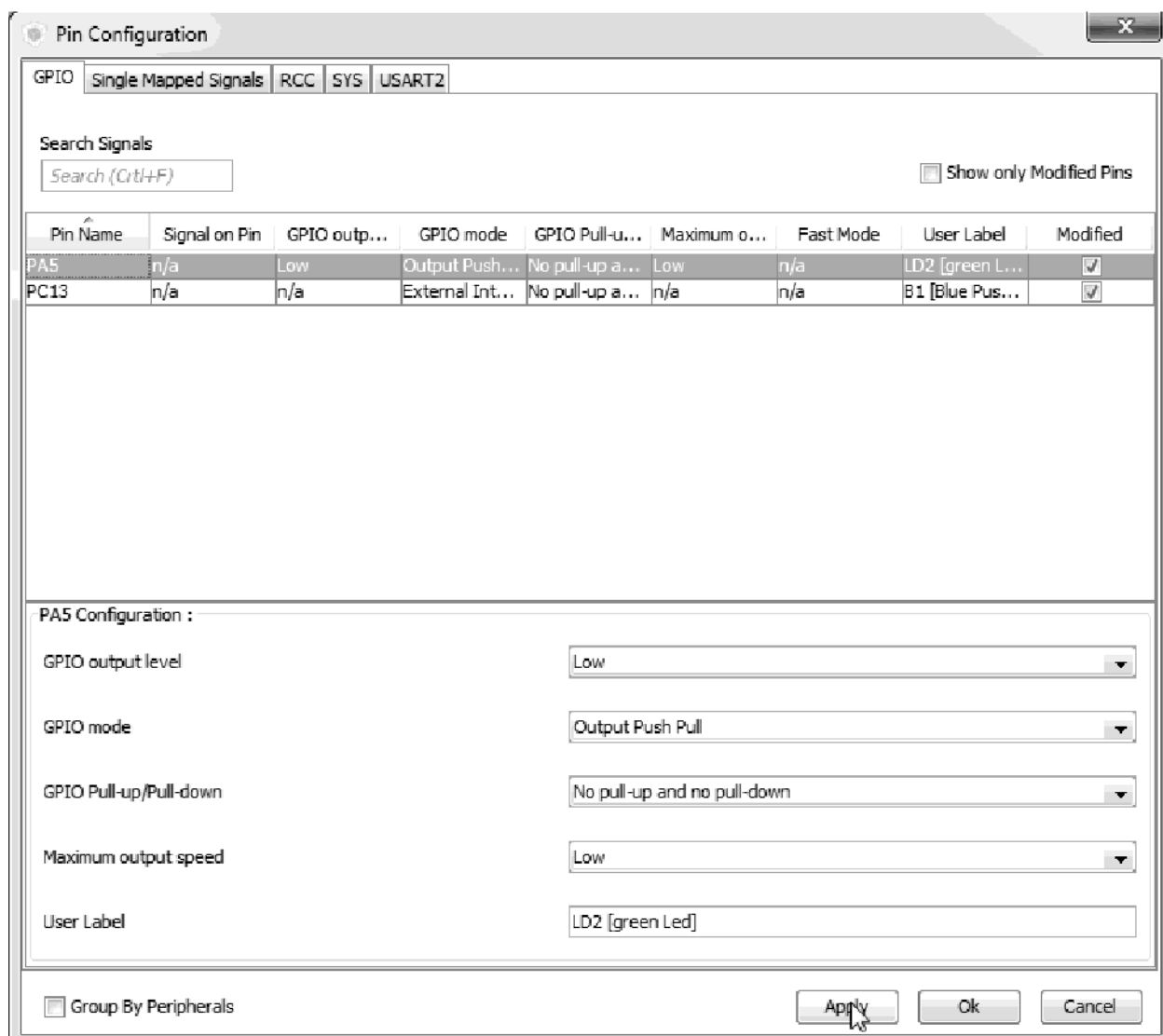


Рисунок 6.8

На этом подготовка проекта завершена и нужно выполнить его генерацию. Во вкладке Project необходимо выбрать пункт Generate Code, задать имя проекта и указать директорию для его хранения (желательно без кириллических символов). Toolchain/IDE (инструментарий, Integrated Development Environment – интегрированная среда разработки) – в этом

пункте необходимо выбрать MDK-ARM V5 с целью генерации проекта для интегрированной среды разработки Keil uVision. CubeMX сразу предложит открыть сгенерированный проект, Keil uVision откроется автоматически. В проводнике слева необходимо выбрать файл main.c и открыть его.

1.2.2 Keil MDK является наиболее комплексной средой разработки для приложений на базе микроконтроллеров ARM. Keil MDK содержит ARM C/C++ компилятор, операционную систему CMSIS-RTOS RTX и среду разработки uVision с отладчиком. Keil uVision позволяет работать с проектами любой степени сложности, начиная с введения и правки исходных текстов и заканчивая внутрисхемной отладкой кода и программированием ПЗУ микроконтроллера.

Интерфейс среды относительно прост. Сверху панель инструментов, слева дерево файлов проекта, справа текстовый редактор, снизу выходная информация о сборке проекта.

После генерации проекта в STM32CubeMX и экспорта в Keil uVision видно, что CubeMX собрал воедино все файлы драйверов и библиотек, написал весь текст файла main.c (рисунок 6.9).

В структуре этого автоматически сгенерированного кода присутствуют комментарии с подсказками и разъяснениями. Рассмотрим те комментарии, в которых есть слова USER CODE BEGIN и USER CODE END. Между этими комментариями необходимо писать свой код. Это обусловлено тем, что после генерации проекта в CubeMX и изменений в Keil uVision, может возникнуть необходимость подключить что-то дополнительно или изменить настройки. Текст программы пользователя, написанный только в этих выделенных частях, при изменениях в CubeMX и генерации проекта в Keil uVision не будет затронут.

int main(void) – точка входа в программу. Затем по тексту происходит инициализация HAL-драйвера (Hardware Abstraction Layer). В работе используется именно этот драйвер.

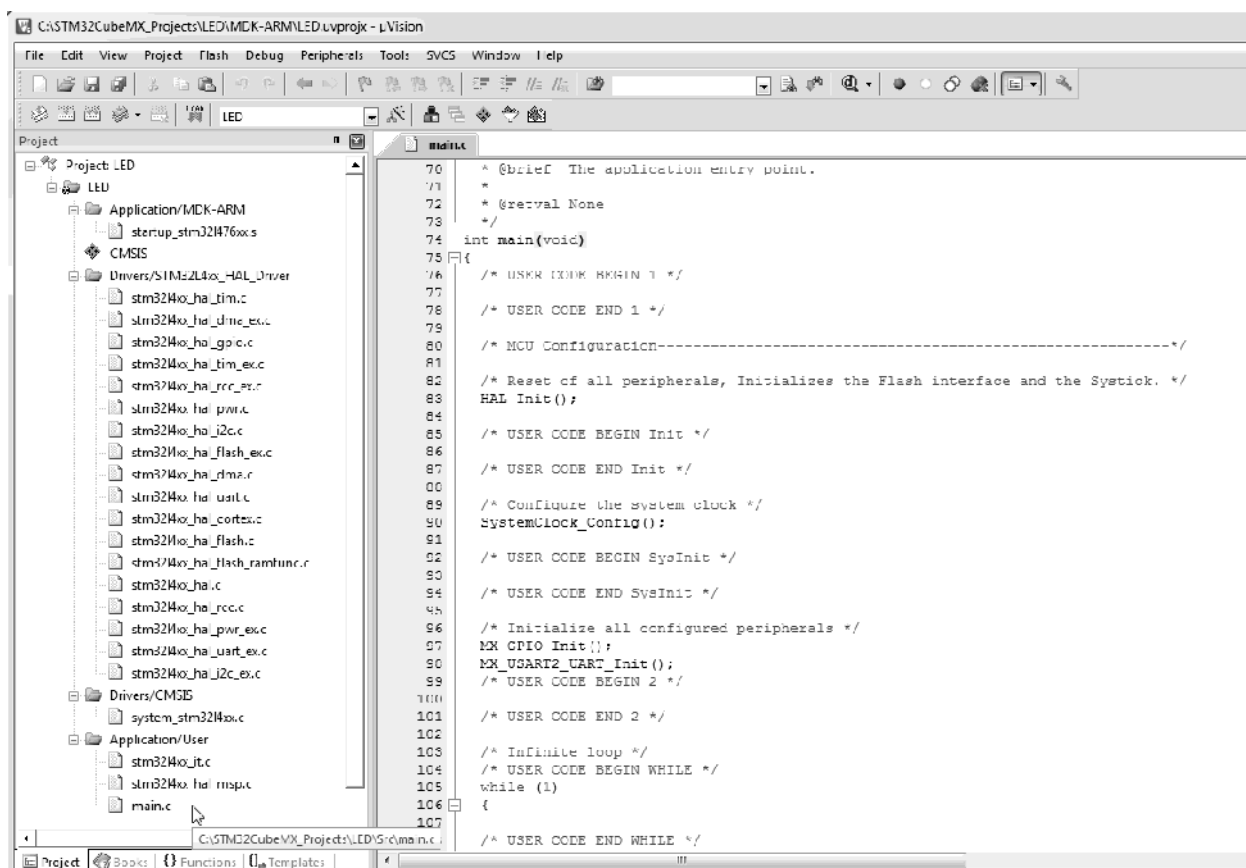


Рисунок 6.9

В файлах этого драйвера содержится описание всех функций работы с аппаратным обеспечением выбранного микроконтроллера. Согласно заданию контрольного примера, необходимо включить светодиод, подключенный к выводу PA5. Соответственно необходимо выбрать в списке файлов и открыть файл `stm32l4xx_hal_gpio.c`.

Для дальнейших работ необходимо изучить все что там написано, но на данном этапе находим в нем нужную информацию. Нажимаем Ctrl+F и ищем в документе слова `HAL_GPIO_Write` (рисунок 6.10). Находим строку 450 и копируем в буфер обмена описание функции, которая будет изменять состояние вывода (рисунок 6.11).

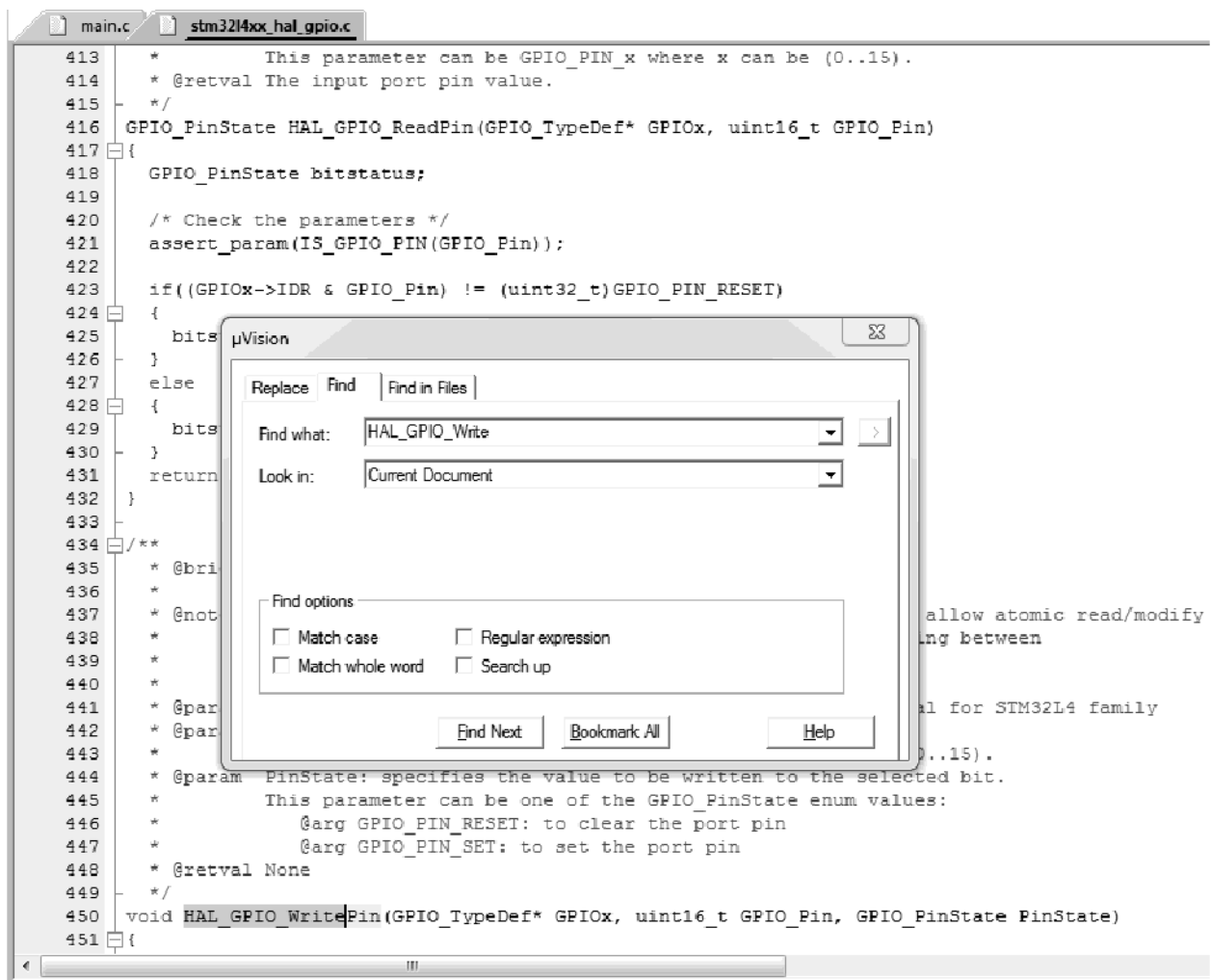


Рисунок 6.10

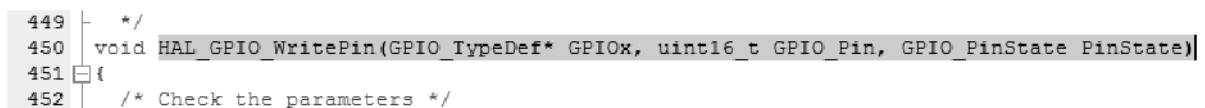
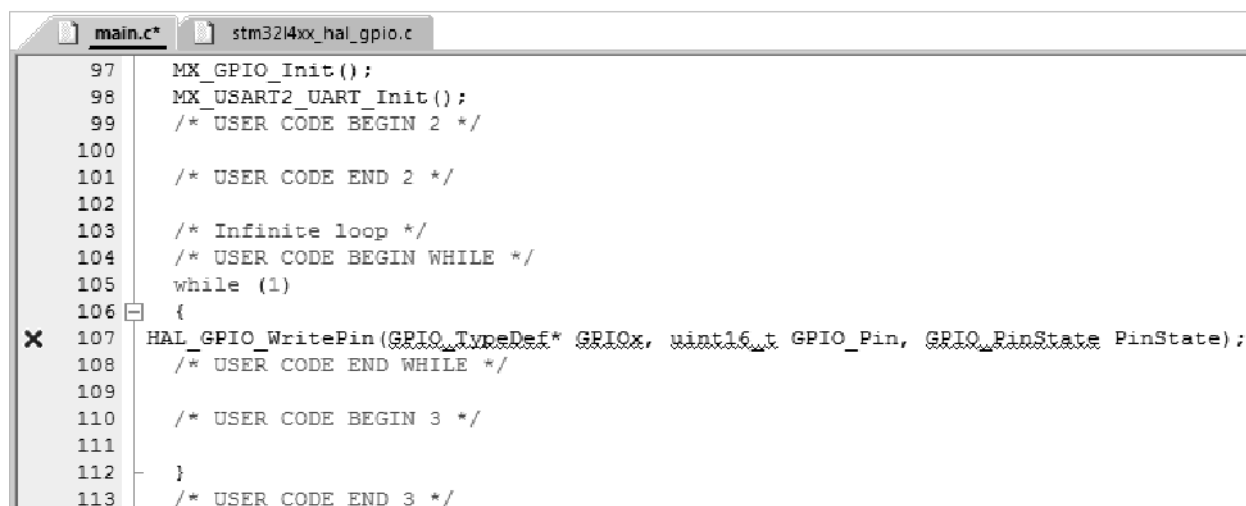


Рисунок 6.11

Затем в файле main.c находим место, где начинается бесконечный цикл while(1). В теле этого цикла и будет происходить включение/выключение светодиода. Скопированный текст необходимо вставить дополнительной 107-ой строкой. Обратите внимание, что слева появится крестик, который обозначает что этот код неправильный (рисунок 6.12).

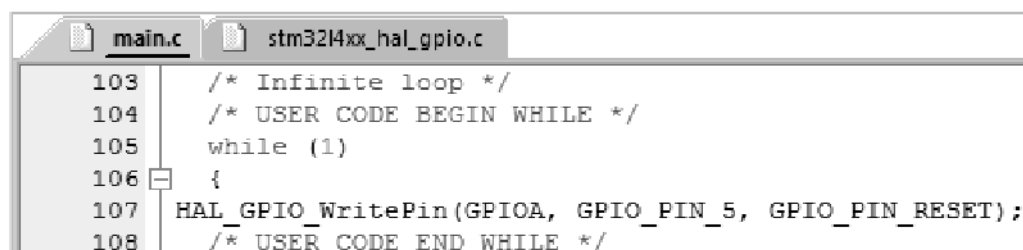
В этой строке необходимо удалить типы переменных, указать конкретный порт и пин, указать его состояние (ноль или единица). Для установки низкого напряжения на выводе необходимо написать

GPIO\_PIN\_RESET (рисунок 6.13).



```
main.c stm324xx_hal_gpio.c
97 MX_GPIO_Init();
98 MX_USART2_UART_Init();
99 /* USER CODE BEGIN 2 */
100
101 /* USER CODE END 2 */
102
103 /* Infinite loop */
104 /* USER CODE BEGIN WHILE */
105 while (1)
106 {
107 HAL_GPIO_WritePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState);
108 /* USER CODE END WHILE */
109
110 /* USER CODE BEGIN 3 */
111
112 }
113 /* USER CODE END 3 */
```

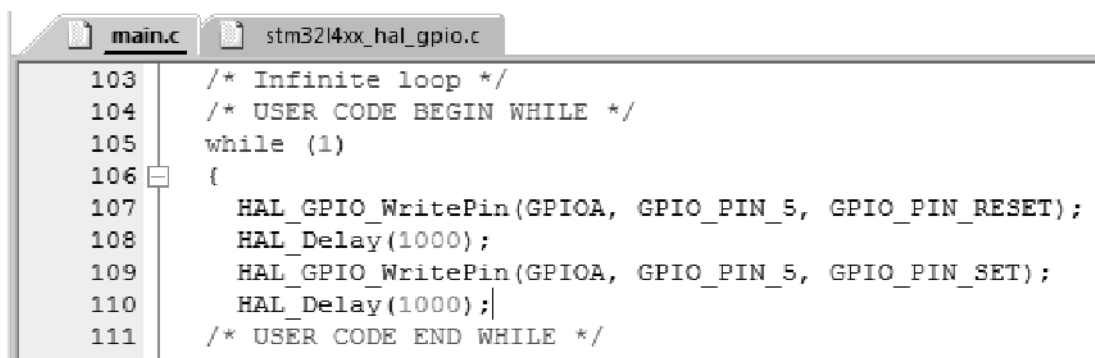
Рисунок 6.12



```
main.c stm324xx_hal_gpio.c
103 /* Infinite loop */
104 /* USER CODE BEGIN WHILE */
105 while (1)
106 {
107 HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);|
108 /* USER CODE END WHILE */
```

Рисунок 6.13

Далее, в соответствии с заданием контрольного примера, необходимо ввести задержку, переключить вывод в высокое логическое состояние и снова сделать задержку. Функция задержки описана в HAL-драйвере как HAL\_Dealy();. Время задержки в ней указывается в миллисекундах. Далее: выставить вывод в единичное состояние – GPIO\_PIN\_SET. На рисунке 6.14 показан код контрольного примера.



```
main.c stm324xx_hal_gpio.c
103 /* Infinite loop */
104 /* USER CODE BEGIN WHILE */
105 while (1)
106 {
107 HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
108 HAL_Delay(1000);
109 HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
110 HAL_Delay(1000);|
111 /* USER CODE END WHILE */
```

Рисунок 6.14

На этом код программы контрольного примера завершен, необходимо выполнить его компиляцию с помощью команды Project -> Build (или нажать F7). Если все сделано верно, то в нижнем окне после компиляции будет выведено сообщение о том, что axf-файл создан без ошибок и предупреждений (рисунок 6.15).

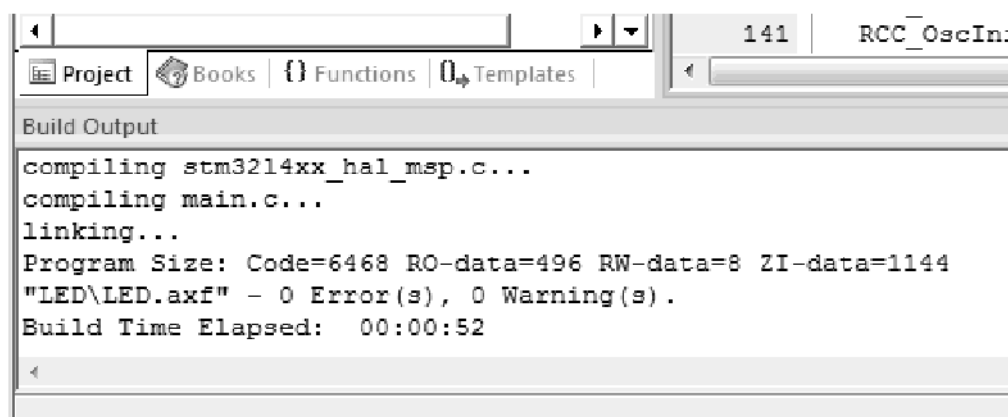


Рисунок 6.15

Теперь необходимо подключить отладочную плату и проверить, что STLink работает. Можно «прошить» плату через STLink Utility, но лучше сразу настроить Keil на работу с ним.

Откройте пункт меню Project -> Options for Target 'LED'... На вкладке Debug -> Options. На вкладке Debug необходимо выбрать Use ST-Link Debugger (рисунок 6.16). Нажмите на кнопку Settings и перед вами откроется новое окно (рисунок 6.17).

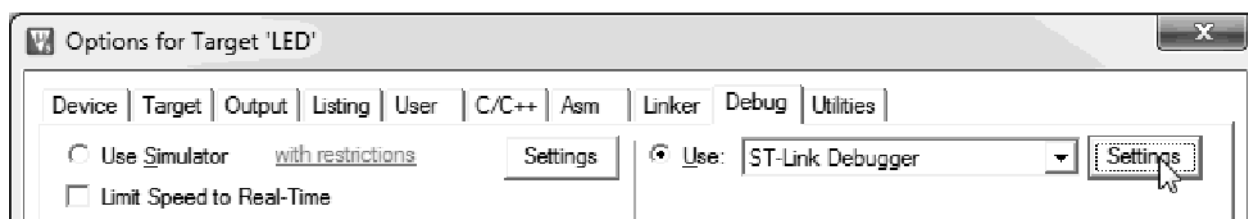


Рисунок 6.16

Рисунок 6.17

На вкладке Flash Download необходимо установить галочку Reset and Run, чтобы программа запускалась на выполнение сразу после перезагрузки (рисунок 6.18).

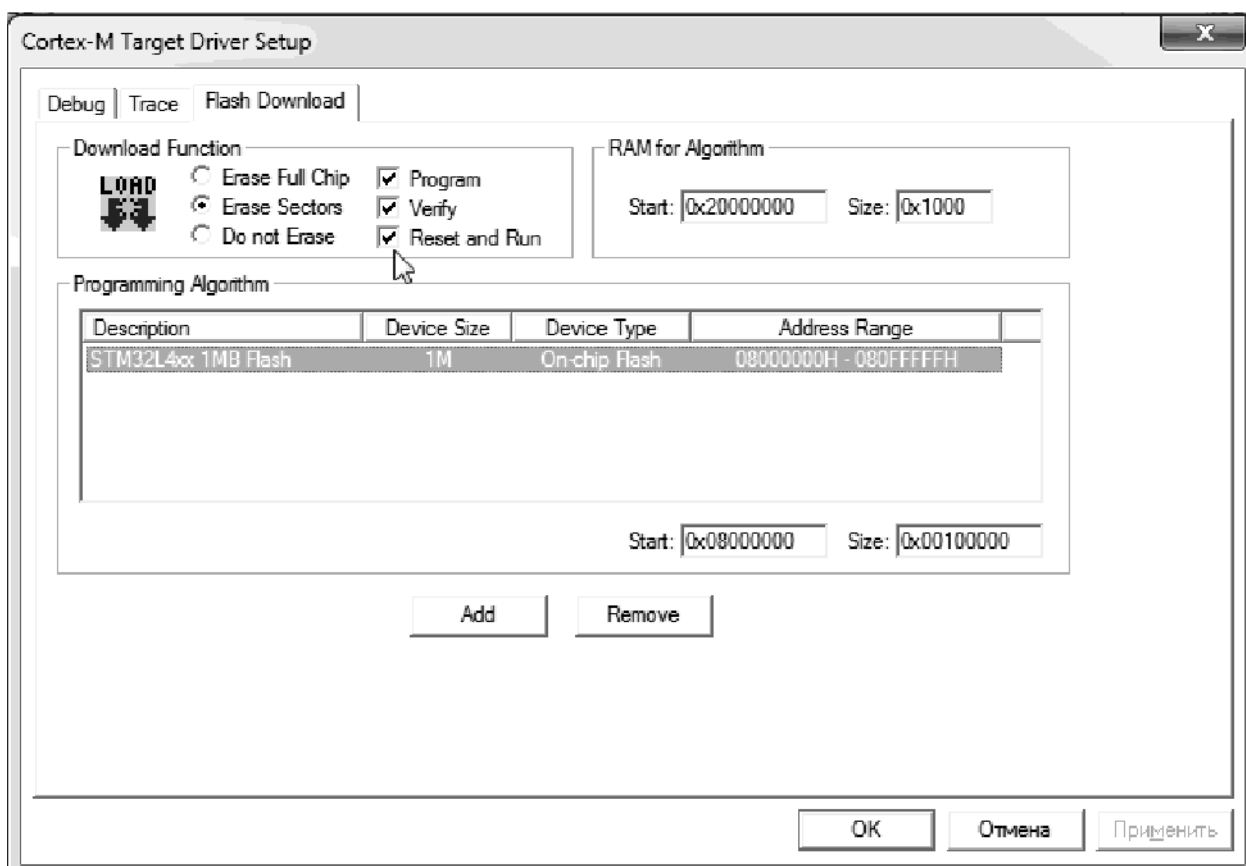


Рисунок 6.18

Далее необходимо нажать на кнопку Load на панели инструментов, программный код контрольного примера запишется в микроконтроллер отладочной платы, и на плате замигает светодиод LD2. Если он не замигал, то, возможно, плата просто не начала исполнять программу после обновления прошивки. Перезагрузите ее при помощи кнопки RESET.

Далее, во всех заданиях, необходимо в самом первом комментарии файла main.c указывать свою фамилию, имя и отчество. В качестве подтверждения выполнения задания необходимо в отчет приводить:

- скрин написанного файла main.c или его непосредственный текст;
- скрины результатов выполнения, если это возможно;

- фото результатов выполнения и собранной схемы (если требуется по заданию);

- тексты всех проектов должны быть сохранены в интегрированной среде разработки Keil uVision для демонстрации при защите.

Изменить время паузы включения и выключения светодиода, чтобы убедиться в том, что это именно написанная программа контролирует его состояние. Выполнить загрузку и тестирование. Сохранить результаты выполнения как описано выше.

## 2 Задание на выполнение работы.

### 2.1 Написать программы для работы с простейшими устройствами.

Во всех заданиях пункта 2 необходимо самостоятельно произвести обоснованный выбор портов ввода/вывода. Допустимо использовать только выводы штыревого разъема ST morpho. Обоснование выбора должно быть описано в отчете.

Схема электрическая принципиальная отладочной платы подключения микроконтроллера приведена на рисунке 6.19, а схема электрическая принципиальная разъемов отладочной платы – на рисунке 6.20.

Необходимо помнить, что напряжение питания микроконтроллера составляет величину 3,3 В, следовательно, для схем, исследуемых в работе, необходимо обеспечить напряжение питания не более указанного значения.

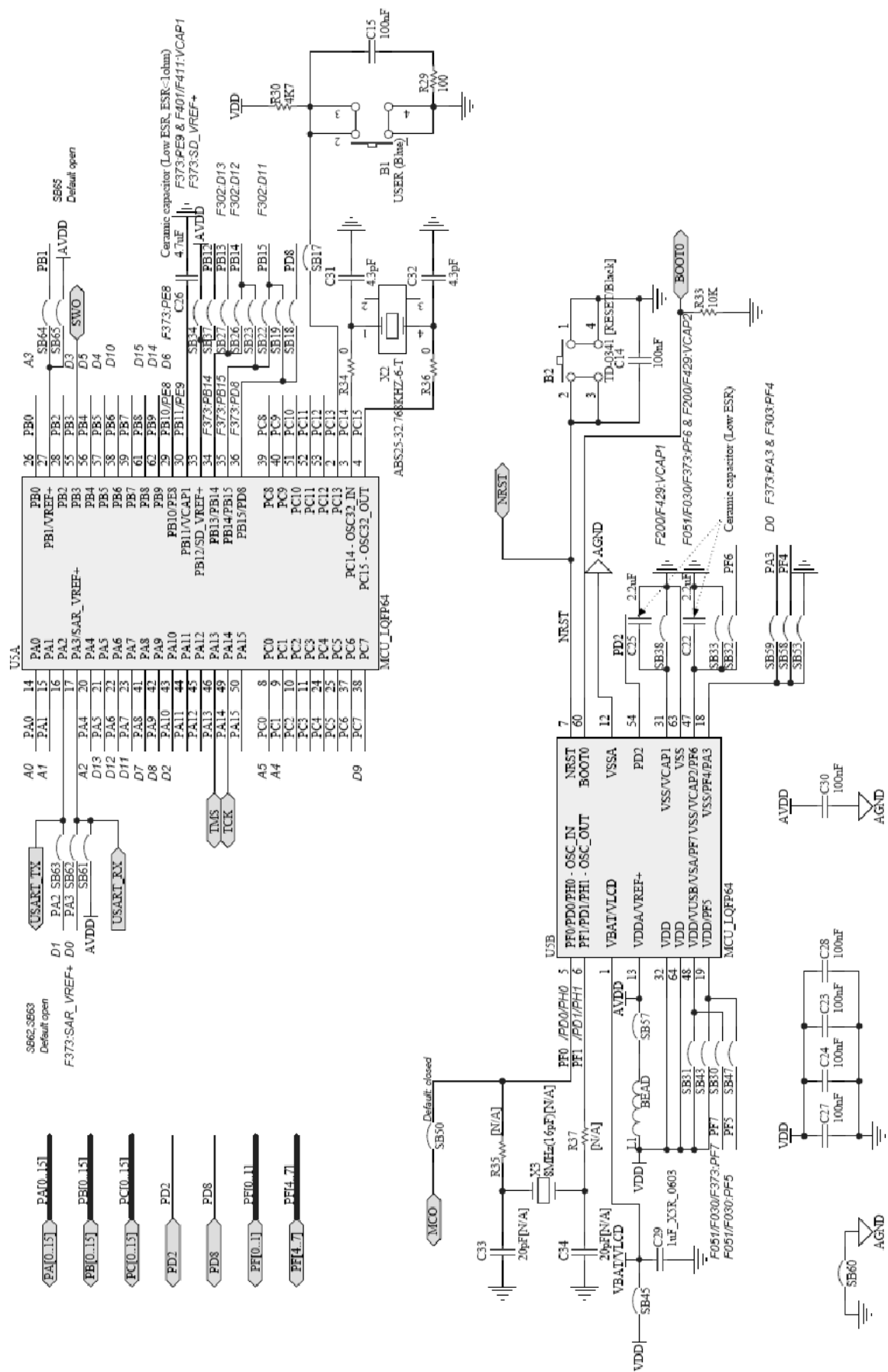


Рисунок 6.19



## 2.1.1 Исследование применения цифровых входов/выходов.

### 2.1.1.1 Мигающий светодиод.

Для реализации примера необходимо на макетной плате собрать схему электрическую принципиальную (рисунок 6.21). Сопротивление резистора – не менее 300 Ом.

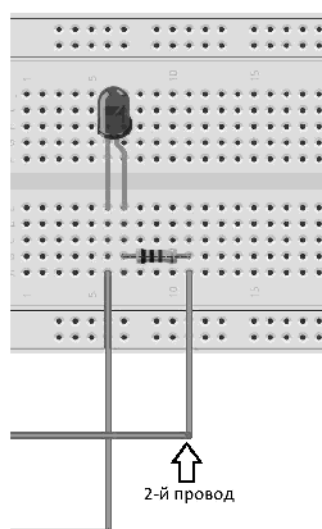


Рисунок 6.21

Написать код программы включения светодиода с миганием по алгоритму «0,5 с (светится); 0,5 с (не светится); n с (светится); n с (не светится); 1 с (светится); 1 с (не светится)» (где n – номер варианта).

Выполнить загрузку кода и тестирование. Сохранить результаты выполнения как описано в п. 1.2.2.

### 2.1.1.2 Кнопка.

Для реализации примера необходимо на макетной плате собрать схему электрическую принципиальную (рисунок 6.22). Сопротивление резистора в цепи кнопки – 10 кОм. Сопротивление резистора в цепи светодиода – не менее 300 Ом.

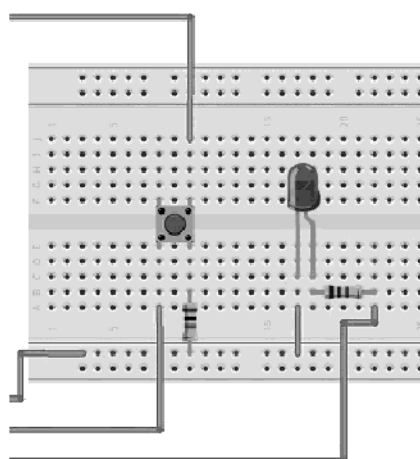


Рисунок 6.22

Написать код программы, чтобы светодиод светился, пока нажата кнопка, и выключался после ее отпускания.

Выполнить загрузку кода и тестирование. Сохранить результаты выполнения как описано в п. 1.2.2.

Модифицировать код таким образом, чтобы светодиод светился при первом нажатии кнопки. После второго нажатия – светодиод должен выключиться. Далее – работа устройства повторяется.

Выполнить загрузку кода и тестирование. Сохранить результаты выполнения как описано в п. 1.2.2.

В рассмотренных примерах очень велик шанс неверного срабатывания устройства, то есть не так, как задумано в алгоритме. Это происходит из-за того, что при нажатии кнопки возникает явление, – «дребезг» контактов. Переходный процесс переключения кнопки показан на рисунке 6.23.

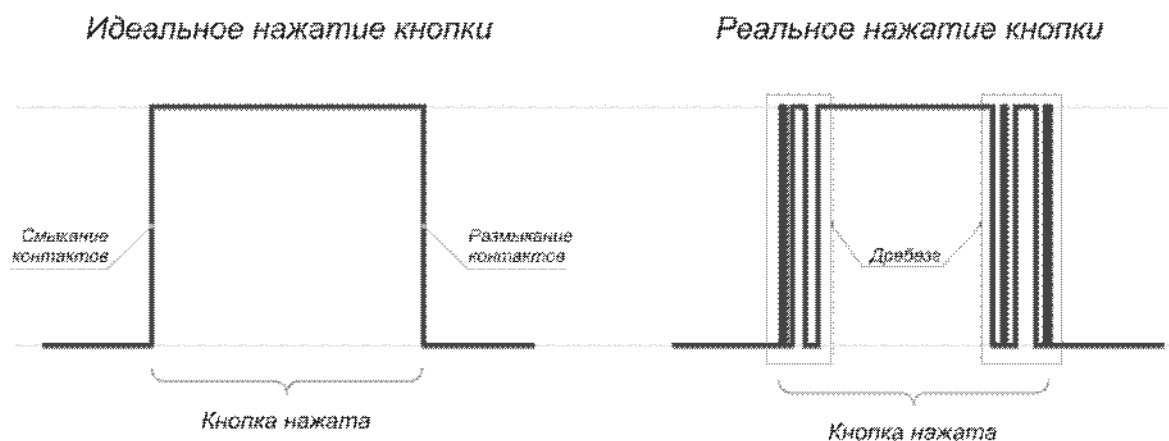


Рисунок 6.23 – Переходный процесс при нажатии кнопки

Переходные процессы протекают очень быстро и исчезают за доли – единицы миллисекунд. Поэтому мы редко их замечаем. Но, обрабатывая сигнал от кнопки в цифровых устройствах, мы вполне можем столкнуться с такими переходными эффектами и должны их учитывать при программировании. Учет сводится к борьбе с последствиями. Возможны два метода решения проблемы: аппаратный и программный.

Необходимо использовать программный способ борьбы с «дребезгом» контактов, принимая, что время переходного процесса не превышает 10 мс.

Модернизировать первые два кода п. 2.1.1.2. Выполнить загрузку кода и тестирование. Сохранить результаты выполнения как описано в п. 1.2.2.

## 2.1.2 Исследование применения аналоговых входов.

### 2.1.2.1 Измерение уровня освещенности.

Аналоговые входы могут только принимать сигнал и измерять напряжение поступающего сигнала. В микроконтроллере изучаемой отладочной платы STM32 Nucleo-64 имеются три 12-битных аналогово-цифровых преобразователя. Это означает, что 0 В считывается как значение «0», а 3,3 В считываются как значение «4095». То есть аналоговые входы измеряют, подаваемое на них напряжение, с точностью до 0,0008 В.

Для измерения уровня освещенности будет использован фоторезистор.

Для реализации примера необходимо на макетной плате собрать схему электрическую принципиальную (рисунок 6.24). Сопротивление резистора в цепи фоторезистора – 10 кОм.

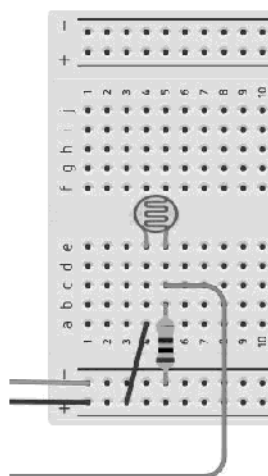


Рисунок 6.24

Написать код программы, которая будет выводить в персональный компьютер через UART микроконтроллера. Программатор ST-Link платы Nucleo уже содержит преобразователь USB-to-COM, подключенный к основному микроконтроллеру.

В CubeMX необходимо включить UART:

- на вкладке Pinout надо включить модуль USART2 в асинхронном режиме;
- на вкладке Configuration изменить настройки подключения, в частности для установки необходимой скорости обмена.

Далее необходимо настроить работу АЦП через DMA с помощью CubeMX.

На следующем шаге нужно сгенерировать проект и открыть его в Keil uVision. Для передачи в асинхронном режиме рекомендуется использовать самую простую функцию – HAL\_UART\_Transmit(). Она расположена в файле драйвера stm32l4xx\_hal\_uart.c. Необходимо вызывать эту функцию в

бесконечном цикле с задержкой и передавать в нее показания с АЦП.

Для приема сообщения на стороне компьютера можно использовать любую терминальную программу, рекомендуется – Terminal 1.9b.

Если все сделано верно, то после компиляции и «прошивки» микроконтроллера, с заданной частотой в COM-порт будут отправляться данные с АЦП, пропорциональные уровню освещенности фоторезистора.

Выполнить загрузку кода и тестирование. Сохранить результаты выполнения как описано в п. 1.2.2.

Модернизировать схему измерителя уровня освещенности, как показано на рисунке 6.24. Сопротивление резистора – не менее 300 Ом.

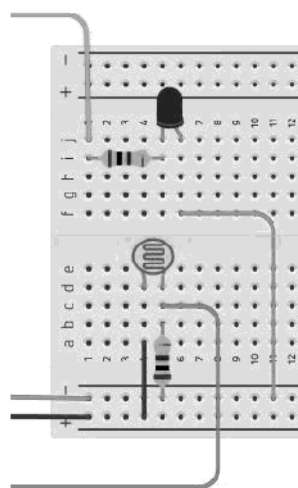


Рисунок 6.24

Написать код программы, которая будет управлять яркостью свечения светодиода по следующему алгоритму:

- для четных номеров вариантов при максимальной освещенности фоторезистора светодиод должен быть выключен. При уменьшении освещенности – яркость свечения светодиода должна возрасти до максимального значения при полном затемнении фоторезистора;

- для нечетных номеров вариантов при максимальной освещенности фоторезистора светодиод должен светиться максимально ярко. При

уменьшении освещенности – яркость свечения светодиода должна уменьшаться до состояния выключения при полном затемнении фоторезистора.

При этом необходимо учитывать, что разрядность ШИМ изучаемой отладочной платы STM32 Nucleo-64 составляет 16 бит (выходной диапазон от 0 до 65535).

Выполнить загрузку кода и тестирование. Сохранить результаты выполнения как описано в п. 1.2.2.

#### 2.1.2.2 Измерение напряжения на аналоговом входе.

Для измерения напряжения на аналоговом входе необходимо на макетной плате собрать схему электрическую принципиальную (рисунок 6.25). Сопротивление резистора – 10 кОм.

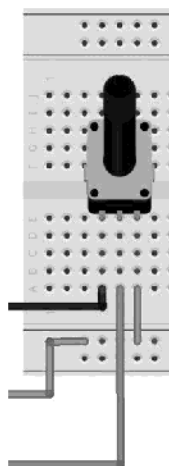


Рисунок 6.25

Написать код программы, используя UART микроконтроллера, передачу данных с АЦП через DMA и терминальную программу, как описано в п. 2.1.2.1. На экране терминальной программы будет выводиться значения напряжения на аналоговом входе, устанавливаемое поворотом ручки переменного резистора. При этом необходимо учесть, что в микроконтроллере изучаемой отладочной платы STM32 Nucleo-64 имеются три 12-битных аналогово-цифровых преобразователя. Это означает, что 0 В

считывается как значение «0», а 3,3 В считаются как значение «4095». То есть аналоговые входы измеряют, подаваемое на них напряжение, с точностью до 0,0008 В. Таким образом, для вывода в персональный компьютер необходимо пересчитать полученное с аналогового входа значение в напряжение. Формула должна быть введена в код программы.

Выполнить загрузку кода и тестирование. Сохранить результаты выполнения как описано в п. 1.2.2.

### 2.1.3 Исследование реализации ШИМ (PWM).

С помощью определенных выходов изучаемой отладочной платы STM32 Nucleo-64 можно реализовать ШИМ-управление объектами. Далее реализуем простейший пример управления яркостью светодиода в зависимости от количества нажатий кнопки. Для реализации примера необходимо на макетной плате собрать схему электрическую принципиальную (рисунок 6.26).

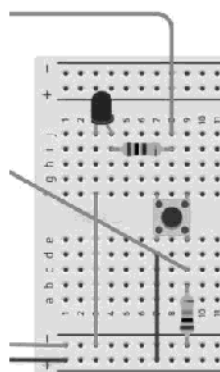


Рисунок 6.26

При каждом нажатии на кнопку яркость свечения светодиода будет изменяться.

Для четных номеров вариантов при включении яркость светодиода должна быть установлена максимальной. При каждом нажатии на кнопку яркость светодиода должна дискретно уменьшаться вплоть до полного выключения.

Для нечетных номеров вариантов при включении яркость светодиода должна быть установлена равной нулю. При каждом нажатии на кнопку яркость светодиода должна дискретно увеличиваться вплоть до максимальной.

При этом необходимо учитывать, что разрядность ШИМ изучаемой отладочной платы STM32 Nucleo-64 составляет 16 бит (выходной диапазон от 0 до 65535). Шаг изменения кода выбрать самостоятельно, чтобы получить возможность полного регулирования за разумное количество нажатий кнопки.

Написать код программы. Выполнить загрузку кода и тестирование. Сохранить результаты выполнения как описано в п. 1.2.2.

#### 2.1.4 Исследование метода управления двигателем постоянного тока.

Цифровые выводы изучаемой отладочной платы STM32 Nucleo-64 обеспечивают ток нагрузки до 20 мА при напряжении до 3,3 В. Для управления нагрузками с более высокими значениями токов и/или напряжений необходимо использование транзистора в ключевом режиме. В данном эксперименте будем управлять скоростью вращения мотора изменением напряжения, подаваемого на транзистор с помощью ШИМ. Для выполнения эксперимента необходимо на макетной плате собрать схему электрическую принципиальную (рисунок 6.27).

База транзистора подключена к отладочной плате через резистор сопротивлением 1 кОм. Параллельно двигателю необходимо подключить защитный диод.

Написать код программы, работающий по следующему алгоритму:

- при включении питания (нажатии кнопки Reset) двигатель не вращается в течении 5 с;

- далее он начинает вращаться со скоростью, пропорциональной максимальной скорости вращения деленной на  $8 \cdot n$  (где  $n$  – номер варианта), в течении 5 с;

- далее он начинает вращаться со скоростью, пропорциональной максимальной скорости вращения деленной на  $3 \cdot n$  (где  $n$  – номер варианта), в течении 5 с;

- далее он начинает вращаться со скоростью, пропорциональной максимальной скорости вращения деленной на  $n$  (где  $n$  – номер варианта), в течении 5 с;

- далее он начинает вращаться с максимальной скоростью в течении 5 с, после чего останавливается на 10 с.

Алгоритм повторяется циклически.

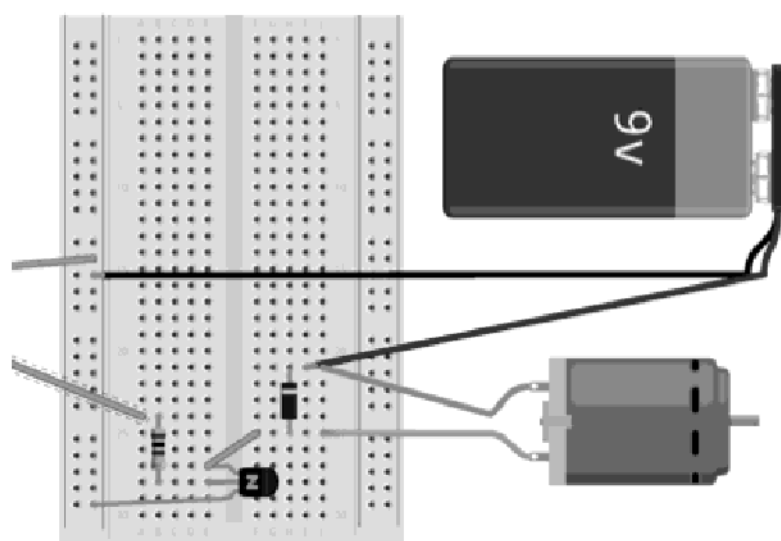


Рисунок 6.27

Выполнить загрузку кода и тестирование. Сохранить результаты выполнения как описано в п. 1.2.2.

#### 2.1.5 Исследование метода управления сервоприводом.

Сервопривод – это мотор, положением вала которого можно управлять. От обычного мотора он отличается тем, что ему можно точно в градусах задать положение, в которое встанет вал. Сервоприводы используются для моделирования различных механических движений роботов.

В отличие от обычных двигателей постоянного тока, сервоприводы обычно помимо выводов питания ( $V_{cc}$  и  $GND$ ) имеют еще один

дополнительный вывод (сигнальный). Сигнальный вывод используется для подачи управляющего сигнала от микроконтроллера, поворачивая тем самым его вал на необходимый угол.

Для выполнения эксперимента необходимо на макетной плате собрать схему электрическую принципиальную (рисунок 6.28).

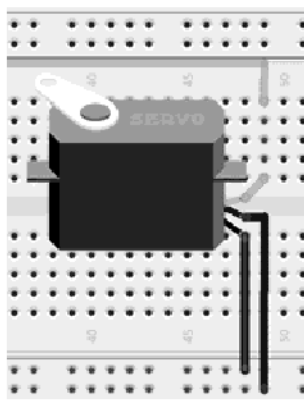


Рисунок 6.28

Написать код программы, работающий по следующему алгоритму:

- при включении питания (нажатии кнопки Reset) вал сервопривода должен установиться в нулевое положение;
- через 3 с вал должен повернуться на угол 170 градусов;
- каждые 3 с вал должен поворачиваться обратно к нулевой позиции с шагом в 10 градусов – для каждого четного номера варианта, и с шагом 15 градусов – для каждого нечетного варианта, с паузой между поворотами в 3 с.

Выполнить загрузку кода и тестирование. Сохранить результаты выполнения как описано в п. 1.2.2.

2.1.6 Исследование метода вывода символов на 7-сегментный индикатор.

Для реализации примера необходимо на макетной плате собрать схему электрическую принципиальную (рисунок 6.29). Сопротивление резисторов –

не менее 300 Ом. Перед подключением необходимо определить тип индикатора: с общим катодом или с общим анодом. На рисунке приведен пример для индикатора с общим катодом. В случае индикатора с общим анодом, его необходимо соединить с гнездом +3,3 V отладочной платы.

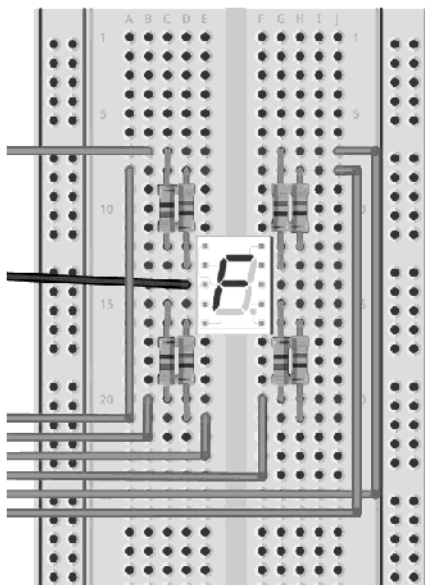


Рисунок 6.29

Написать код программы для вывода на индикатор номера варианта в 16-й системе счисления. Если номер варианта превышает максимальное число для данного индикатора, то необходимо из числа 20 вычесть номер своего варианта и полученную разность в 16-й системе счисления выводить на индикатор.

Выполнить загрузку кода и тестирование. Сохранить результаты выполнения как описано в п. 1.2.2.

Модернизировать код таким образом, чтобы показания индикатора мигали с частотой 1 Гц.

Выполнить загрузку кода и тестирование. Сохранить результаты выполнения как описано в п. 1.2.2.

### 3 Задание на обработку данных.

#### 3.1 Сделать выводы о работе цифровых входов/выходов.

- 3.2 Сделать выводы о работе аналоговых входов.
- 3.3 Сделать выводы о работе PWM-выводов (ШИМ) платы.
- 3.4 Сделать выводы об управлении двигателем постоянного тока.
- 3.5 Сделать выводы об управлении сервоприводом.
- 3.6 Сделать выводы о выводе информации на светодиодный индикатор.
- 3.7 Сделать выводы о полученных результатах работы написанных программ.

### **Содержание отчета**

Отчет по лабораторной работе выполняется на листах формат А4 согласно требований ГОСТ и ЕСКД и должен содержать следующие пункты:

- а) титульный лист с названием работы, дисциплины, кода группы, ф.и.о. исполнителя и преподавателя;
- б) цель и содержание работы;
- в) краткое описание аппаратной части и программного обеспечения, необходимого для выполнения работы;
- г) написанные программы приведенных заданий (в виде скринов или текста). Комментарии обязательны;
- д) результаты исследований работы кодов (фото и/или скрины);
- е) выводы по результатам работы.

### **Контрольные вопросы**

- 1.

**Список литературы, рекомендованный к использованию по данной теме**

**Основная учебная литература**

**Дополнительная литература**

## **ПРАКТИЧЕСКАЯ РАБОТА 7**

**Тема:** Программирование электронных узлов и устройств на базе отладочной платформы STM32.

**Цель работы:** закрепление знаний, полученных на лекциях по разделам 3 (лекции 5 и 6) и 4 (лекция 8). В ходе выполнения лабораторной работы необходимо:

- произвести обоснованный выбор элементной базы для реализации электронного устройства;
- собрать схему электрическую принципиальную посредством макетирования;
- написать программное обеспечение для электронных узлов и блоков согласно вариантам. Выполнить загрузку и, в случае необходимости, отладку.

**Формируемые компетенции:** ОПК-9.

### **Теоретическая часть**

Отладочная платформа STM32 применяется для создания электронных устройств с возможностью приема сигналов от различных цифровых и аналоговых датчиков, которые могут быть подключены к нему, а также, – управления различными исполнительными устройствами. Проекты устройств, основанные исследуемой платформе, могут работать самостоятельно или взаимодействовать с программным обеспечением на компьютере.

Выполнение электронных устройств на платформе STM32 позволяет:

- быстро протестировать схемотехническое решение разработанного устройства и в случае наличия ошибок оперативно их исправить;
- быстро выполнить тестирование и отладку алгоритма работы электронного устройства и в случае наличия ошибок оперативно их исправить.

Далее, разработанный проект может быть выполнен на отдельной печатной плате, включающей микроконтроллер STM32, и необходимую периферию.

Именно поэтому важным является изучение методики программирования электронных узлов и устройств на базе отладочной платформы STM32.

### **Оборудование и материалы**

Лабораторная работа выполняется с использованием визуального графического редактора STM32CubeMX, интегрированной среды разработки Keil uVision и аппаратных средств отладочной платформы STM32.

### **Указания по технике безопасности**

Перед проведением лабораторной работы следует изучить инструкцию по технике безопасности:

- при работе с отладочной платформой STM32;
- при работе на компьютере.

**ВНИМАНИЕ!** Если в варианте задания используются внешние по отношению к отладочной платформе устройства, то все работы по сборке схемы, изменению порядка внешних соединений, подключению измерительных приборов должны производиться **ТОЛЬКО** при отключенном напряжении источника питания и/или отключенном кабеле USB, соединяющем платформу с ПК.

Включение напряжения питания необходимо производить только после проверки преподавателем правильности коммутаций.

При подключении внешних устройств **НЕОБХОДИМО** проверять уровни напряжений питания и сигналов во избежание повреждения внешних

устройств и отладочной платформы.

Также следует соблюдать правила и требования, установленные для данного компьютерного класса.

### **Задания**

1 Задание на подготовку к выполнению работы.

1.1 Выполнить сборку схемы электронного устройства.

Произвести обоснованный выбор элементной базы для реализации электронного устройства.

Во всех заданиях пункта 1 необходимо самостоятельно произвести обоснованный выбор портов ввода/вывода. Допустимо использовать только выводы штыревого разъема ST morpho. Обоснование выбора должно быть описано в отчете.

Схема электрическая принципиальная отладочной платы подключения микроконтроллера и схема электрическая принципиальная разъемов отладочной платы приведены на соответствующих рисунках (см. рисунок 6.19 и 6.20) лабораторной работы № 6.

Необходимо помнить, что напряжение питания микроконтроллера составляет величину 3,3 В, следовательно, для схем, исследуемых в работе, необходимо обеспечить напряжение питания не более указанного значения или сигнал с внешнего по отношению к отладочной плате устройства не должен превышать указанного значения.

Тип устройства выбирается в соответствии с номером варианта, заданным преподавателем. Исходные данные приведены в таблице 7.1.

Подключить необходимые датчики (элементы регулировки и т. д.) и/или объекты управления (устройства индикации и т. д.) непосредственно к отладочной плате STM32 Nucleo-64 или с использованием дополнительной макетной платы (Shield).

Включение напряжения питания и подключение к ПК необходимо производить только после проверки преподавателем правильности реализации схемного решения и произведенных коммутаций.

Таблица 7.1

| Вариант | Устройство                        | Рекомендации                                                                                                                                                                                                                                                                                                                      |
|---------|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | Измеритель температуры            | Цифровой датчик температуры<br>Символьный дисплей 16x2 (в первой строке выводится надпись «Temperature = xx», где xx – числовое значение измеренной температуры. Во второй строке надпись «Normal», когда значение температуры от 0 до 35 градусов Цельсия включительно; «Attention», когда температура выше 35 градусов Цельсия) |
| 2       | Электронный измеритель расстояния | Ультразвуковой датчик<br>Символьный дисплей 16x2 (в первой строке выводится надпись «Distance = xx cm», где xx – числовое значение измеренного расстояния в сантиметрах. Во второй строке надпись «Normal» для расстояния до объекта от 10 до 20 см, «Close» – для расстояния менее 10 см и «Away» – для расстояния более 20 см)  |
| 3       | Клавиатура управления объектом    | Клавиатура 4x4<br>Символьный дисплей (в первой строке выводится «Code = xxxx», где xxxx – код нажатой кнопки; во второй строке «Symbol = x», где x символ, соответствующий нажатой кнопке)                                                                                                                                        |
| 4       | Система дистанционного управления | ИК-пульт<br>ИК-приемник<br>Графический дисплей (в первой строке выводится «Code = xxxx», где xxxx – код нажатой на пульте кнопки; во второй строке «Symbol = x», где x символ, соответствующий нажатой на пульте кнопке)                                                                                                          |

|    |                               |                                                                                                                                                                                                                                                                                                                               |
|----|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5  | Таймер обратного отсчета      | Сенсорный датчик<br>4-разрядный семисегментный индикатор<br>Счет от 80 до 0. Каждый новый отсчет начинается при касании сенсорного датчика                                                                                                                                                                                    |
| 6  | Счетчик посещений             | Инфракрасный датчик препятствий<br>4-разрядный семисегментный индикатор                                                                                                                                                                                                                                                       |
| 7  | Электронная метеостанция      | Цифровой датчик температуры<br>Цифровой датчик влажности<br>Графический дисплей (в первой строке выводится надпись «Temperature = xx °C», где xx – числовое значение измеренной температуры. Во второй строке выводится надпись «Humidity = yy %», где yy – числовое значение измеренной влажности)                           |
| 8  | Охранное устройство           | Модуль датчика вибрации<br>Сенсорный датчик<br>Графический дисплей (в первой строке выводится надпись «OFF». При установлении факта вибрации в эту строку выводится надпись «Vibration». При касании сенсорного датчика снова выводится надпись «OFF» до момента установления факта следующей вибрации)                       |
| 9  | Цифровой индикатор напряжения | Подстроечный резистор<br>Шкала линейная светодиодная<br>При минимальном измеряемом напряжении светится только 1 разряд шкалы, при максимальном – все                                                                                                                                                                          |
| 10 | Измеритель температуры        | Цифровой датчик температуры<br>Графический дисплей (в первой строке выводится надпись «Temperature = xx», где xx – числовое значение измеренной температуры. Во второй строке надпись «Normal», когда значение температуры от 0 до 32 градусов Цельсия включительно; «Attention», когда температура выше 32 градусов Цельсия) |

|    |                                                                     |                                                                                                                                                                                                                                                                                                                                                                          |
|----|---------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11 | Устройство управления скоростью вращения двигателя постоянного тока | Двигатель постоянного тока<br>Подстроечный резистор<br>При минимальном напряжении регулировки – двигатель не вращается, при максимальном – вращается с максимальной скоростью. Напряжение регулировки считывается со среднего вывода резистора (диапазон от 0 до 5 В)                                                                                                    |
| 12 | Устройство управления сервоприводом                                 | Сервопривод с фиксированным углом<br>Энкодер инкрементный<br>При перемещении ручки энкодера на по часовой стрелке сервопривод поворачивается также по часовой стрелке на такое же количество фиксированных шагов, на сколько шагов повернули ручку энкодера. при вращении против часовой стрелки алгоритм работы такой же (сервопривод вращается против часовой стрелки) |
| 13 | Устройство ограничения доступа                                      | Метка RFID<br>Приемник RFID<br>Графический дисплей (в первой строке выводится «UID = x...x», где x...x – идентификационный номер метки; во второй строке «Access is allowed», если UID совпадает с установленным в скетче и «Access is denied», если UID не совпадает с установленным в скетче)                                                                          |
| 14 | Электронный измеритель расстояния                                   | Ультразвуковой датчик<br>Символьный дисплей 16x2 (в первой строке выводится надпись «Distance = xx см», где xx – числовое значение измеренного расстояния в сантиметрах. Во второй строке надпись «Normal» для расстояния до объекта от 10 до 20 см, «Close» – для расстояния менее 10 см и «Away» – для расстояния более 20 см)                                         |
| 15 | Электронный замок                                                   | Метка RFID<br>Приемник RFID<br>Сервопривод с фиксированным углом                                                                                                                                                                                                                                                                                                         |

|    |                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----|----------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|    |                                                    | Если UID совпадает с установленным в скетче, то сервопривод поворачивается на угол 90 градусов, Через 5 секунд вал сервопривода возвращается в первоначальное (нулевое) положение. Если UID не совпадает, то вращения вала нет                                                                                                                                                                                                                                                                                                                                                |
| 16 | Ультразвуковой датчик парковки                     | Ультразвуковой датчик<br>Звукоизлучатель со встроенным генератором частоты<br>При расстоянии до датчика более 60 см, звуковой сигнал не формируется. При расстоянии от 40 до 60 см, звуковой сигнал формируется с частотой 0,25 Гц (время звучания равно времени паузы). При расстоянии от 20 до 40 см, звуковой сигнал формируется с частотой 1 Гц (время звучания равно времени паузы). При расстоянии от 10 до 20 см, звуковой сигнал формируется с частотой 2 Гц (время звучания равно времени паузы). При расстоянии менее 10 см, звуковой сигнал формируется постоянно. |
| 17 | Устройство автоматической регулировки освещенности | уровня<br>Фоторезистор<br>Светодиод<br>При полном освещении фоторезистора светодиод выключен; при максимальном затемнении – светодиод светит максимально ярко; при разной освещенности фоторезистора яркость свечения светодиода должна быть разной: чем больше затемнение фоторезистора, тем ярче должен светиться светодиод                                                                                                                                                                                                                                                 |
| 18 | Система дистанционного управления                  | ИК<br>ИК-пульт<br>ИК-приемник<br>Символьный дисплей (в первой строке выводится «Code = xxxx», где xxxx – код нажатой на пульте кнопки; во второй строке «Symbol = x», где x символ, соответствующий нажатой на                                                                                                                                                                                                                                                                                                                                                                |

|    |                                |                                                                                                                                                                                                                                                                                                                                                                         |
|----|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|    |                                | пульте кнопке                                                                                                                                                                                                                                                                                                                                                           |
| 19 | Устройство распознавания цвета | Датчик цвета<br>Графический дисплей (в первой строке выводится «Red = xx», где xx – значение красного цвета; во второй строке «Green = yy», где yy – значение зеленого цвета; «Blue = zz», где zz – значение синего цвета; в четвертой строке выводится название «Red», «Green» или «Blue», когда определяемый цвет максимально приближен к одному из указанных цветов) |
| 20 | Клавиатура управления объектом | Клавиатура 3x4<br>Графический дисплей (в первой строке выводится «Code = xxxx», где xxxx – код нажатой кнопки; во второй строке «Symbol = x», где x символ, соответствующий нажатой кнопке)                                                                                                                                                                             |

1.2 Написать программное обеспечение для исследования электронных узлов и блоков, согласно заданию. Во всех вариантах, где используется механическая коммутация контактов (клавиатура, переключатели, энкодер и т. д. необходимо предусмотреть процедуру подавления «дребезга» контактов).

## 2 Задание на выполнение работы.

2.1 Подключить отладочную плату STM32 Nucleo-64 к ПК. Выполнить необходимые настройки микроконтроллера и его периферии в визуальном графическом редакторе STM32CubeMX. Сгенерировать проект и открыть его в интегрированной среды разработки Keil uVision.

2.2 Написать программный код, выполнить его загрузку и тестирование.

Если в коде будут обнаружены ошибки, это будет отображено в нижней части окна. Необходимо устранить ошибки и записать код в плату снова.

При успешной загрузке и правильном коде устройство будет функционировать согласно заданию.

В качестве подтверждения выполнения задания необходимо в отчет приводить:

- скрин написанного файла main.c или его непосредственный текст;
- скрины результатов выполнения, если это возможно;
- фото результатов выполнения и собранной схемы (если требуется по заданию);
- тексты всех проектов должны быть сохранены в интегрированной среде разработки Keil uVision для демонстрации при защите.

### 3 Задание на обработку данных.

3.1 Сделать выводы о работе датчиков (элементов регулировки и т. д. в зависимости от варианта задания).

3.2 Сделать выводы о работе объектов управления (устройств индикации и т. д. в зависимости от варианта задания).

3.3 Сделать выводы о полученных результатах работы написанной программы и функционировании электронного устройства в целом.

## **Содержание отчета**

Отчет по лабораторной работе выполняется на листах формат А4 согласно требований ГОСТ и ЕСКД и должен содержать следующие пункты:

- а) титульный лист с названием работы, дисциплины, кода группы, ф.и.о. исполнителя и преподавателя;
- б) цель и содержание работы;
- в) обоснование выбора компонентов для реализации устройства и портов ввода/вывода отладочной плате STM32 Nucleo-64;
- г) схему электрическую принципиальную и фото электронного устройства;

д) написанную программу для электронного устройства (в виде скринов или текста). Комментарии обязательны;

е) результаты исследования работы программы (фото и/или скрины);

ж) выводы по результатам работы.

### **Контрольные вопросы**

1.

**Список литературы, рекомендованный к использованию по данной теме**

#### **Основная учебная литература**

#### **Дополнительная литература**

## ПРАКТИЧЕСКАЯ РАБОТА 8

**Тема:** Изучение основ моделирования в многоязычной среде моделирования HDL – ModelSim.

**Цель работы:** закрепление знаний, полученных на лекциях по разделу 4 (лекция 9). В ходе выполнения лабораторной работы необходимо:

- ознакомиться с возможностями системы моделирования ModelSim и ее графическим интерфейсом;
- создать проект примера в Modelsim;
- для своего варианта создать поведенческий модуль на основе структуры case и тестовый стенд;
- для своего варианта создать структурный и поведенческий модули в соответствии со схемой и тестовый стенд.

**Формируемые компетенции:** ОПК-9.

### Теоретическая часть

Одна из характерных тенденций современного этапа развития технологии проектирования цифровых систем – применение языков описания аппаратуры HDL (Hardware Description Language) высокого уровня, среди которых наибольшее распространение получили VHDL и Verilog.

При этом не только возрастает роль средств моделирования в процессе разработки устройств, но и меняются методы и требования, предъявляемые к процедурам верификации. Для своевременного обнаружения возможных ошибок средства моделирования должны обеспечивать возможность контроля результатов каждого этапа процесса проектирования: создания исходных HDL-описаний, синтеза, размещения и трассировки в кристалл. Такой подход обеспечивает минимальное время разработки устройства и сокращает стоимость этого процесса, так как цена ошибки возрастает с каждым последующим шагом проектирования.

Пакет программных средств ModelSim™ корпорации Model Technology (одного из подразделений компании Mentor Graphics) в настоящее время является самой распространенной системой HDL-моделирования.

Именно поэтому важным является изучение и получение навыков моделирования в многоязычной среде моделирования HDL – ModelSim.

### **Оборудование и материалы**

Лабораторная работа выполняется с использованием Пакета программных средств ModelSim™.

### **Указания по технике безопасности**

Перед проведением лабораторной работы следует изучить инструкцию по технике безопасности при работе на компьютере. Также следует соблюдать правила и требования, установленные для данного компьютерного класса.

### **Задания**

1 Задание на подготовку к выполнению работы.

1.1 Изучение пользовательского интерфейса пакета ModelSim.

Графический интерфейс системы моделирования ModelSim представлен девятью окнами:

- основное окно (Main window);
- окно исходного кода (Source);
- окно структуры проекта (Structure);
- окно переменных (Variables);
- окно сигналов (Signals);

- окно таблиц (List);
- окно процессов (Process);
- окно временных диаграмм (Wave);
- окно трассировки сигналов (Dataflow).

Основное окно ModelSim является главным элементом пользовательского интерфейса пакета. Оно содержит все необходимые инструменты управления процессами компиляции и моделирования. Открытие вспомогательных окон, используемых для отображения информации о проекте и результатах моделирования в различной форме, и их конфигурирование осуществляется из главного окна. Структура основного окна ModelSim (рисунок 8.1) включает следующие элементы:

- заголовок окна (1);
- главное меню (2);
- оперативная панель управления (3);
- рабочая область (4);
- консольная область (5);
- строка состояния (6).

В заголовке главного окна отображается название программы моделирования с указанием ее версии и редакции.

Главное меню основного окна открывает доступ ко всем командам управления проектом, процессами компиляции и моделирования, отображением результатов моделирования, установки необходимых значений параметров и конфигурирования системы.

Каждый пункт главного меню используется для активизации одноименного всплывающего меню, в котором представлена соответствующая группа команд. Всплывающее меню File включает команды создания, открытия и закрытия проектов, файлов и наборов данных, а также очистки и сохранения содержимого консольной области в виде файла на диске. Кроме того, в эту группу включены команды установки параметров

консольной области и завершения работы. Выпадающее меню Edit объединяет команды редактирования исходного HDL-кода и установки точек прерывания. В выпадающем меню View сгруппированы команды, определяющие вид основного окна ModelSim и открывающие вспомогательные окна.

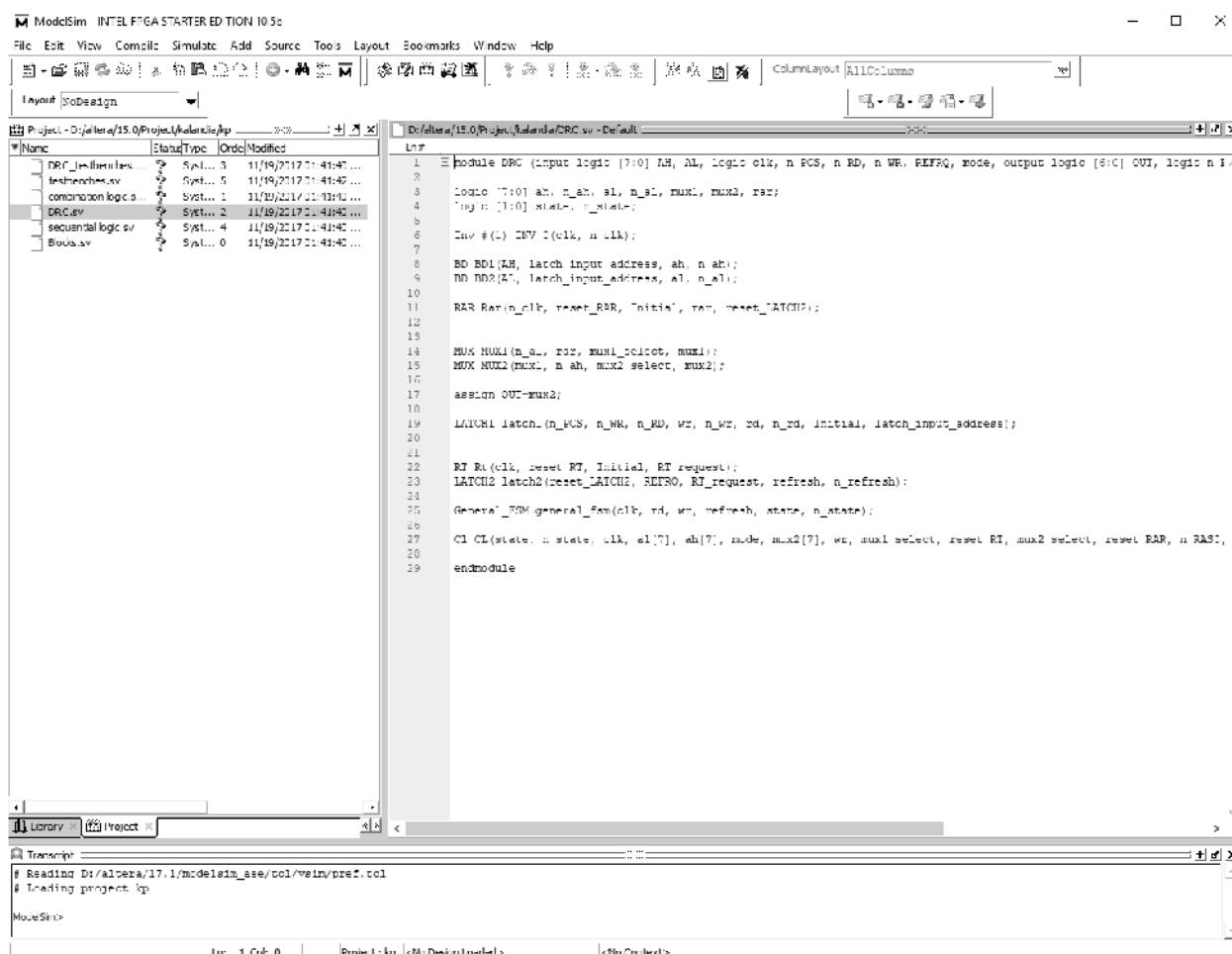


Рисунок 8.1 – Основное окно системы моделирования

Всплывающее меню Compile включает настройки компиляции и позволяет компилировать файлы проекта. Меню Simulate содержит команды для запуска, остановки и конфигурации процесса моделирования.

Выпадающее меню Add, позволяет пользователю добавить элементы в соответствующих окнах рабочей области (Wave, Dataflow, List и т. д.)

Всплывающее меню Project содержит команды работы с проектом, которые позволяют включать в проект файлы исходных описаний и

выполнять компиляцию всех файлов проекта.

Всплывающее меню Window содержит команды установки взаимной конфигурации открытых окон пакета ModelSim. Пункт Help основного меню предоставляет справочную информацию о программах пакета.

Оперативная панель управления содержит кнопки быстрого доступа, которые дублируют наиболее часто используемые команды меню пакета ModelSim.

Рабочая область основного окна ModelSim представляет собой эффективный визуальный инструмент управления проектом. Здесь отображается детальная информация о структуре текущего проекта и используемых библиотек. Выполнение необходимых операций осуществляется с помощью команд всплывающих контекстно-зависимых меню, которые выводятся на экран при щелчке правой кнопки мыши на свободном поле или на выделенной строке в рабочей области. Эта область содержит, как правило, несколько страниц, снабженных закладками с их названиями (Project, Library, Sim), каждая из которых представляет соответствующий вид информации о проекте. Страница Project предназначена для отображения списка файлов, которые входят в состав текущего проекта. С помощью команд всплывающего контекстно-зависимого меню пользователь может быстро открыть выбранный файл для редактирования, выполнить компиляцию отдельного или всех файлов, добавить или удалить файлы из проекта. Страница Library позволяет просмотреть состав библиотек, используемых в проекте, и выполнить необходимые операции с ними.

В поле выбора, расположенном в верхней части страницы, указывается название библиотеки, структуру которой необходимо просмотреть. Для этого используется выпадающий список доступных библиотек, который выводится при нажатии кнопки, расположенной у правой границы поля выбора. Команды всплывающего контекстно-зависимого меню позволяют

редактировать и удалять модули, входящие в состав выбранной библиотеки, перекомпилировать ее, а также создавать новые библиотеки. Страница Sim содержит подробную информацию об иерархической структуре проекта, для которого активизирован процесс моделирования. Эта информация может отображаться в сжатом или развернутом виде. Выбор формата отображения осуществляется с помощью команд всплывающего контекстно-зависимого меню. Кроме рассмотренных, в рабочей области могут присутствовать страницы, отражающие структуру открытых наборов данных.

В консольной области основного окна фиксируются выполняемые команды и все сообщения программы, которые появлялись в текущем сеансе работы пакета ModelSim. Кроме того, в нижней части этой области отображается командная строка, в начале которой указывается заголовок, соответствующий текущему режиму работы. Командная строка позволяет издавать команды управления, используя клавиатуру. Содержимое консольной области может быть сохранено в виде файла на диске. Сохраненные файлы далее могут использоваться в качестве макросов (командных файлов), вызываемых в последующих сеансах работы с программой.

В строке состояния, расположенной в нижней части основного окна, отображается информация о загруженном проекте, текущем времени и итерации моделирования, название выбранного элемента проекта.

## 1.2 Изучение этапов создания проекта в Modelsim.

### 1.2.1 Создание нового проекта.

Для создания проекта необходимо открыть программу и выбрать File -> New -> Project (рисунок 8.2).

Откроется окно диалога создания проекта. Заполняем необходимые поля и нажимаем ОК (рисунок 8.3).

Библиотека будет располагаться в той же директории, что и проект. Далее появится окно, позволяющее добавить элементы в проект

(рисунок 8.4). Выбираем необходимое действие и закрываем окно.

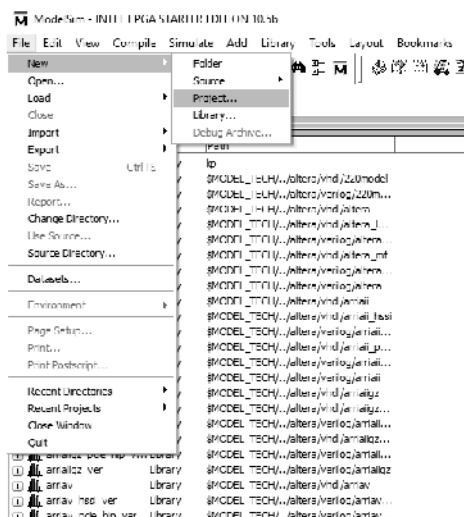


Рисунок 8.2 – Новый проект

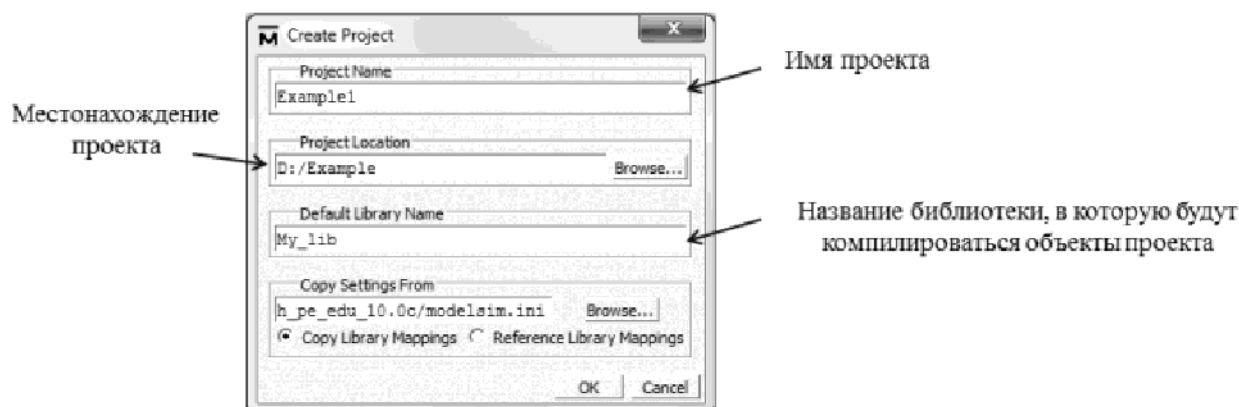


Рисунок 8.3 – Диалоговое окно создания проекта

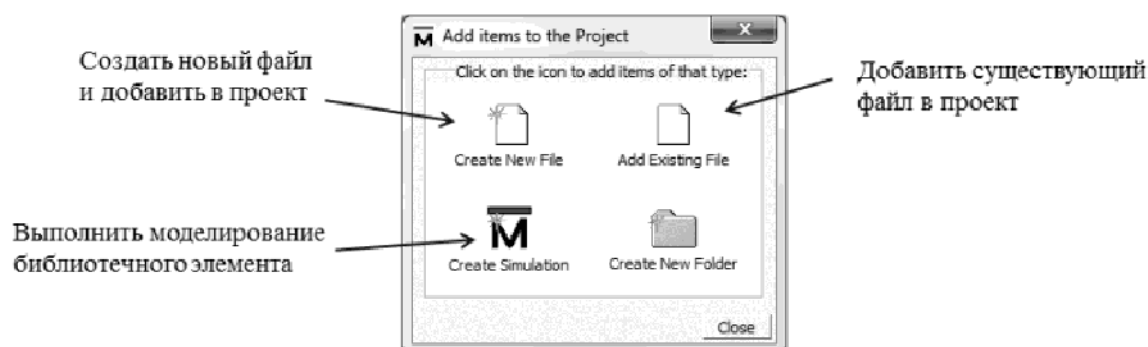


Рисунок 8.4 – Окно предварительной конфигурации создаваемого проекта

В случае выбора создание нового файла (Create New File) появится

окно создания файла (рисунок 8.5). Заполняем форму и нажимаем ОК.

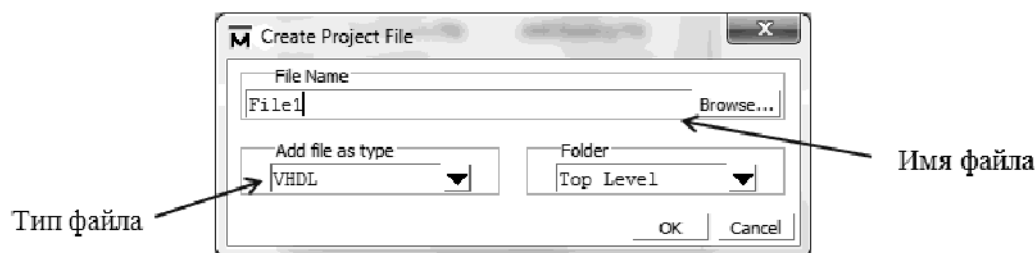


Рисунок 8.5 – Окно создания нового файла

В итоге на экране будет показано окно Project с только что созданным файлом (рисунок 8.6) – это окно, позволяет управлять файлами проекта и их компиляцией. В нём отображается тип файла и его статус, который указывает на наличие или отсутствие ошибок компиляции, а также на то, что процесс компиляции не запускался для данного файла с момента его последнего изменения.



Рисунок 8.6 – Окно проекта

## 1.2.2 Написание и компиляция программы.

Окно редактирования открывается двойным щелчком левой кнопкой мыши по редактируемому файлу, либо через контекстное меню Edit, вызываемое нажатием правой кнопкой мыши на файле проекта.

В качестве примера введите в окно Source код (дешифратор), показанный на рисунке 8.7.

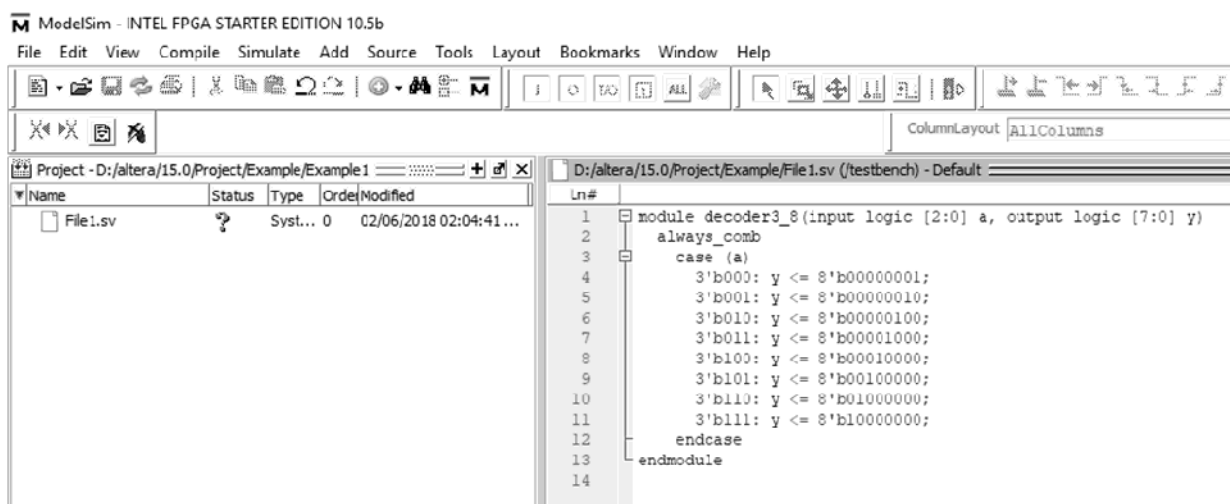


Рисунок 8.7 – Код дешифратора

Для того чтобы проверить код на ошибки его необходимо скомпилировать. Предварительно необходимо сохранить файл (окно Source должно быть активным), через пункт основного меню File -> Save или кнопку с изображением дискеты на панели инструментов. Затем через пункт основного меню Compile -> Compile selected или нажатием правой кнопкой мыши на файле в окне Project выбираем из контекстного меню Compile -> Compile selected запускаем процесс компиляции.

В случае обнаружения ошибки изменится статус нашего файла и в окне Transcript появится сообщение об ошибке. Для того, чтобы найти и исправить ошибку, щелкаем два раза на сообщение об ошибке. Увидим окно с описанием и место ошибки (рисунок 8.8).

Исправьте все ошибки, сохраните файл и скомпилируйте его ещё раз.

При успешной компиляции в окне Transcript будет получено соответствующее сообщение и изменится статус файла.



Рисунок 8.8 – Окно с описанием обнаруженной ошибки

### 1.2.3 Создание Testbench.

Проверка функционирования алгоритма проектируемого устройства осуществляется с помощью испытательного стенда (Testbench), представляющего собой еще один объект проекта. В его архитектуре декларируется тестируемое устройство как компонент, подключаются к нему воздействующие сигналы (вектора) и осуществляется моделирование.

В ModelSim содержатся встроенные шаблоны, которые можно использовать для создания испытательных стендов. Вызов окна шаблонов осуществляется при активном окне Source. Выбираем из меню Source -> Show Language Templates. Слева появится дополнение к окну Source (рисунок 8.9). ДОСТУПНО НЕ ВО ВСЕХ ВЕРСИЯХ!

Двойной щелчок левой кнопкой мыши по пункту Create Testbench вызовет окно мастера создания испытательного стенда (рисунок 8.10). Выполните необходимые шаги (рисунок 8.11).

Как только настройки будут завершены, в проект будет добавлен файл с испытательным стендом. Если оставили все галочки в настройках, то файл откроется автоматически.

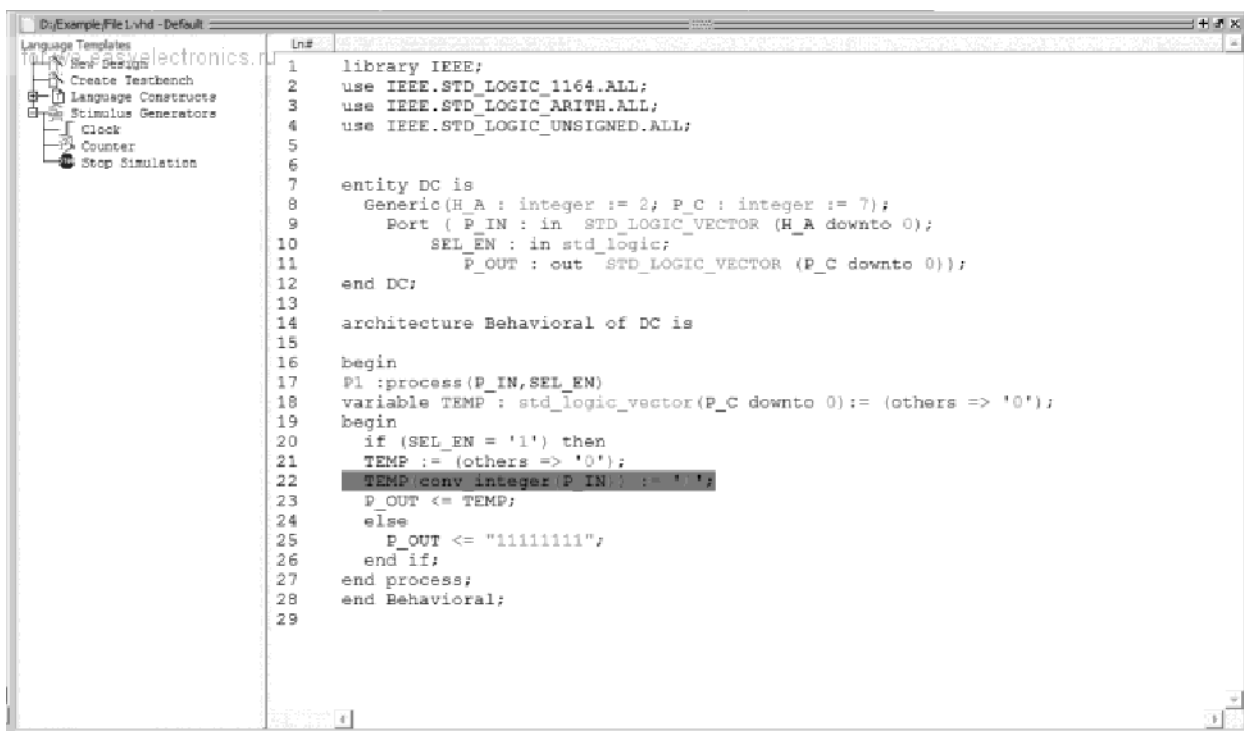


Рисунок 8.9 – Встроенные шаблоны

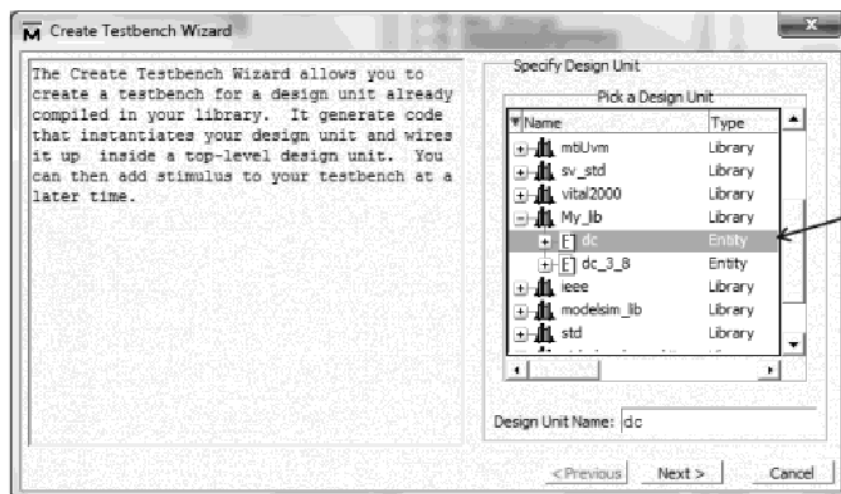


Рисунок 8.10 – Окно мастера создания испытательного стенда

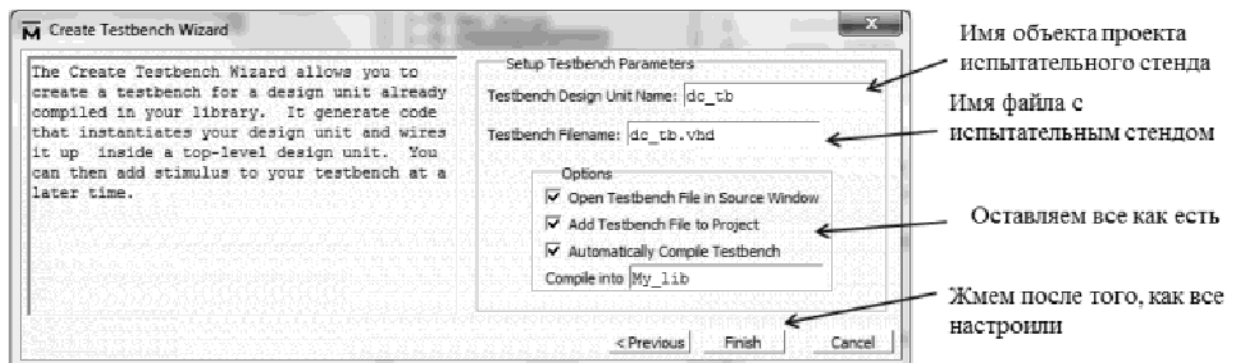


Рисунок 8.11 – Настройки испытательного стенда

Будет создан новый объект проекта, в декларативной части архитектуры объявлено тестируемое устройство в виде компонента, а так же все необходимые сигналы. В теле архитектуры объявлено тестируемое устройство и к нему подключены все необходимые сигналы.

Следующим этапом необходимо задать воздействия, что также можно сделать при помощи шаблонов. Поместить курсор в конце тела архитектуры (окно Source), в окне шаблонов выбрать Stimulus Generators -> Counter. Затем произвести необходимые настройки в соответствующих окнах (рисунок 8.12).

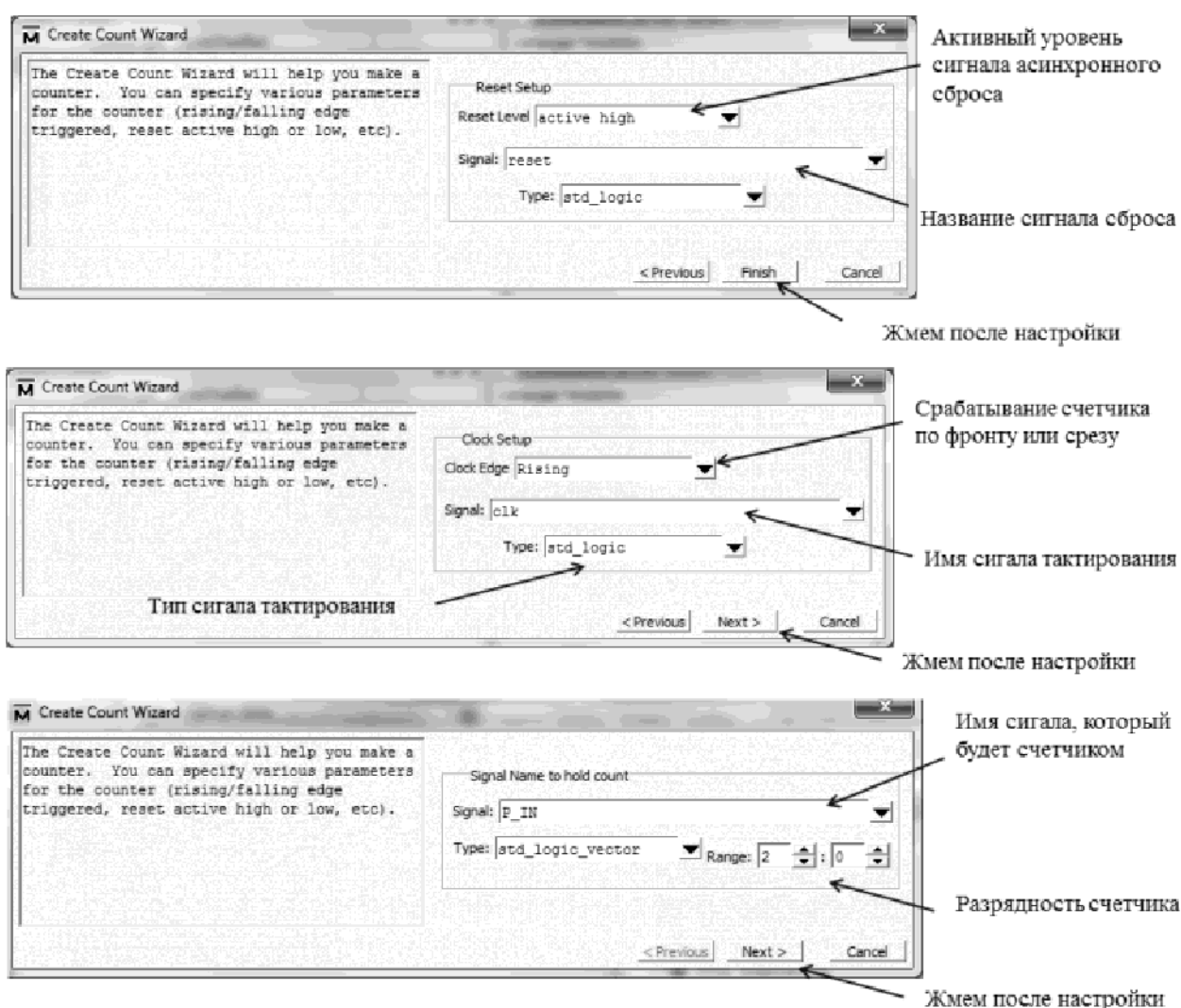


Рисунок 8.12 – Окна мастера настройки воздействующих сигналов (векторов)

Возможно добавление тактовых сигналов с использованием шаблона из меню Stimulus Generators -> Clock (рисунок 8.13).

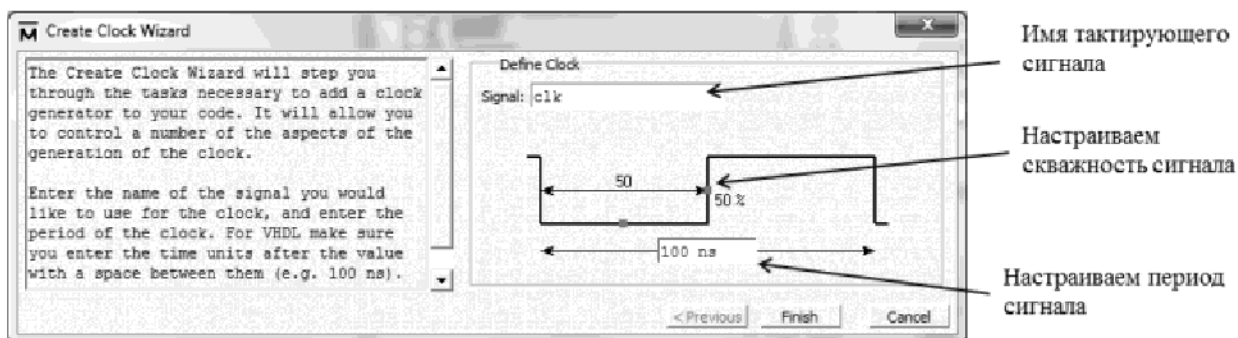


Рисунок 8.13 – Окна мастера настройки тактового сигнала

В результате код будет изменён. При необходимости можно подкорректировать код (для задания или изменения начальных значений векторов). Затем файл необходимо сохранить и скомпилировать.

Испытательный стенд также можно создать в виде исходного кода не прибегая к шаблонам. Для чего необходимо либо добавить новый файл к проекту или внести изменения в исходном файле проекта.

Примеры такого кода приведен на рисунке 8.14.

Для кода, приведённого на рисунке 8.14, также понадобится текстовый файл с векторами воздействия. Добавить файл в проект можно через контекстное меню Add to Project -> NewFile окна Project либо через основное меню Project -> Add to Project -> NewFile. В появившемся окне (рисунок 8.5) указать тип файла текстовый, название «example.txt» и закрыть окно. Отредактируйте вновь созданный файл в соответствии с рисунком 8.15.

```

1 module testbench();
2 logic clk, reset;
3 logic [2:0] a;
4 logic [7:0] y, yexpected;
5 logic [4:0] vectornum, errors;
6 logic [10:0] testvectors[8:0];
7
8 // instantiate device under test
9 decoder3_8 dut(a, y);
10
11 // generate clock
12 always
13 begin
14 clk = 1; #5; clk = 0; #5;
15 end
16
17 initial
18 begin
19 $readmemb("example.txt", testvectors);
20 vectornum = 0; errors = 0;
21 reset = 1; #27; reset = 0;
22 end
23
24 always @(posedge clk)
25 begin
26 #1; {a, yexpected} = testvectors[vectornum];
27 end
28
29 always @(negedge clk)
30 if (~reset) begin
31 if (y != yexpected) begin
32 $display("Error: inputs = %b", a);
33 $display(" outputs = %b (%b expected)", y, yexpected);
34 errors = errors + 1;
35 end
36 vectornum = vectornum + 1;
37 if (testvectors[vectornum] === 11'bx) begin
38 $display("%d test completed with %d errors", vectornum, errors);
39 $stop;
40 end
41 end
42 endmodule
43
44 module decoder3_8(input logic [2:0] a, output logic [7:0] y);
45 always_comb
46 case (a)
47 3'b000: y <= 8'b00000001;
48 3'b001: y <= 8'b00000010;
49 3'b010: y <= 8'b00000100;
50 3'b011: y <= 8'b00001000;
51 3'b100: y <= 8'b00010000;
52 3'b101: y <= 8'b00100000;
53 3'b110: y <= 8'b01000000;
54 3'b111: y <= 8'b10000000;
55 end
56 endmodule

```

Рисунок 8.14 – Пример кода испытательного стенда

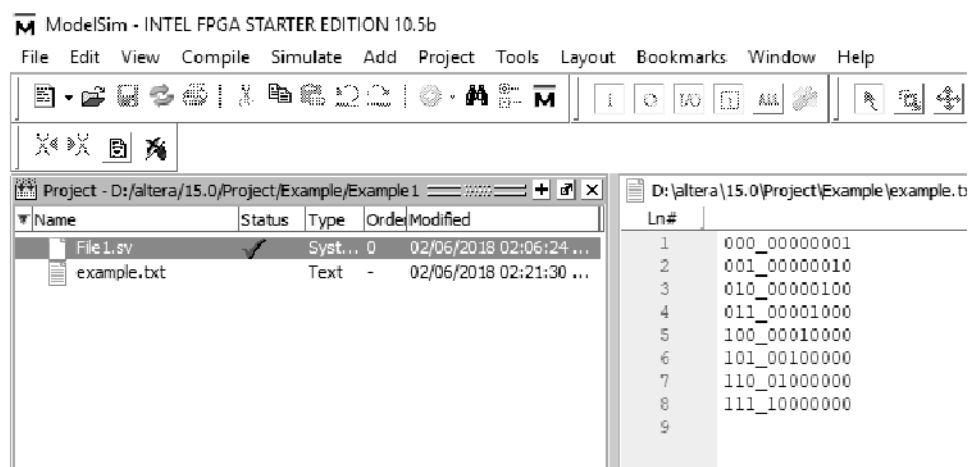


Рисунок 8.15 – Содержимое файла векторов воздействия

#### 1.2.4 Моделирование работы.

Запустить моделирование можно несколькими способами. Выбрать в окне Library библиотеку проекта и объект проекта, работу которого необходимо смоделировать, нажать правую кнопку мыши и выбирать Simulate из контекстного меню (рисунок 8.16). Через основное меню Simulate->Start Simulation..., что приведёт к открытию окна выбора проекта для моделирования (рисунок 8.17), в котором нужно будет выбрать объект моделирования и нажать кнопку ОК.

Запустится моделирование. В зависимости от настроек могут отображаться различные окна. Их расположение можно сохранять и сбрасывать при помощи меню Layout. Отображение и скрывание окон можно осуществлять через меню View.

Добейтесь приблизительного вида окон как показано на рисунке 8.18.

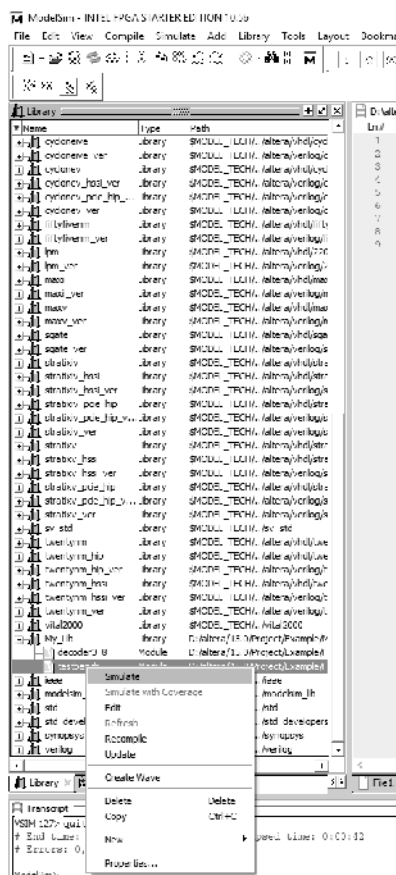


Рисунок 8.16 – Запуск моделирования через контекстное меню

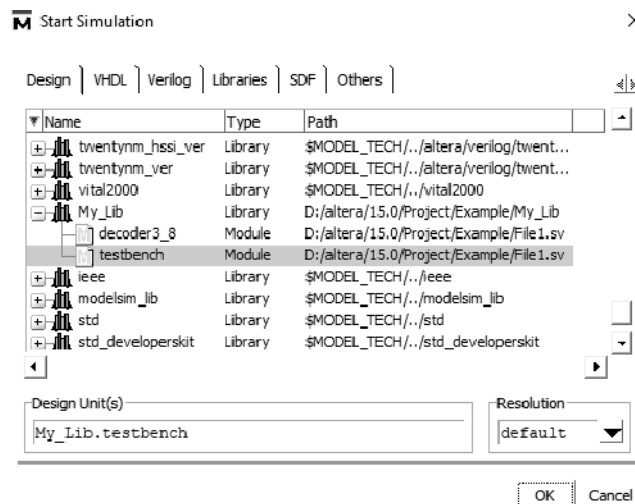


Рисунок 8.17 – Окно выбора объекта моделирования

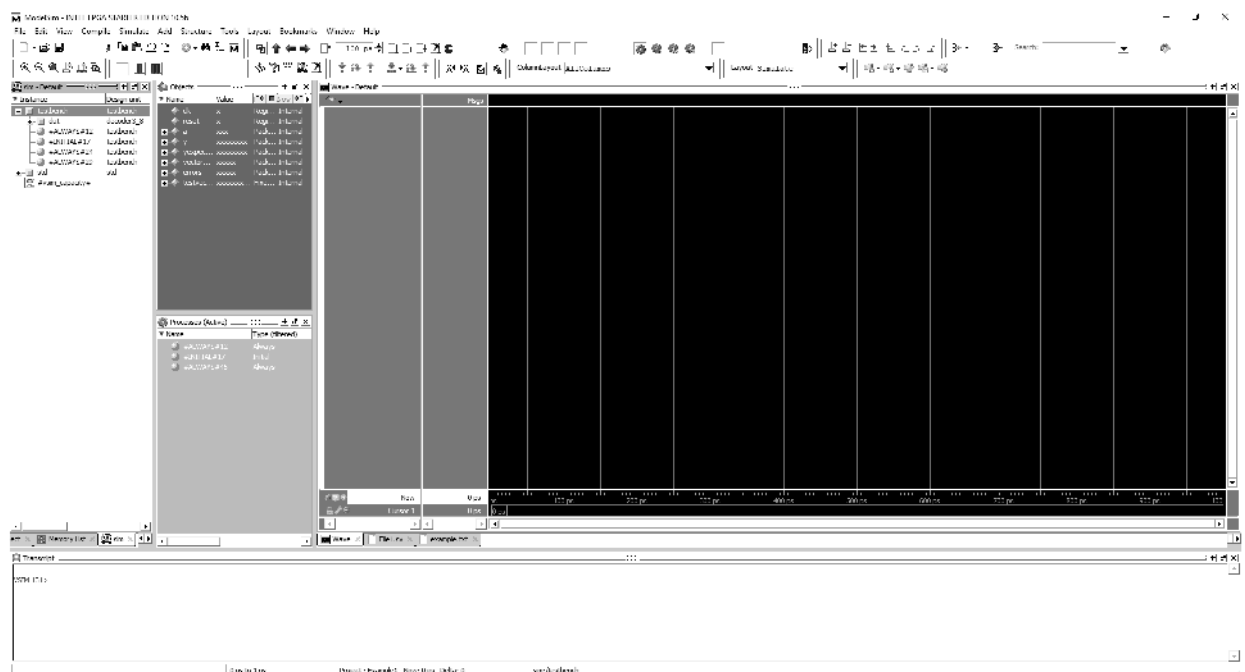


Рисунок 8.18 – Взаимное расположение окон при запуске моделирования

В ModelSim довольно много среди них:

- Wave – окно построения временных диаграмм;
- Structure – окно структуры проекта, показывает иерархию проекта, при выборе любой структуры в остальных окнах будут отображаться ее данные;

- Objects – окно сигналов, здесь отображаются сигналы выбранного объекта проекта;
- Processes – показывает существующие процессы и их статус;
- Dataflow – структура процессов, отображает схему протекания данных.

В ModelSim поддерживается Drag and Drop, поэтому можно перетаскивать элементы из одного окна в другое. Переместите из окна сигналов необходимые сигналы в окно временной диаграммы. Для выделения непрерывной группы объектов можно использовать клавишу Shift, в случае выделения группы объектов с пропусками можно использовать клавишу Ctrl, либо перетаскивать объекты по одному. Перенесите все сигналы в окно временной диаграммы. Измените, отображение сигналов errors и vectornum на беззнаковые десятичные, для этого из контекстного меню выбрать Radix -> Unsigned (рисунок 8.19).

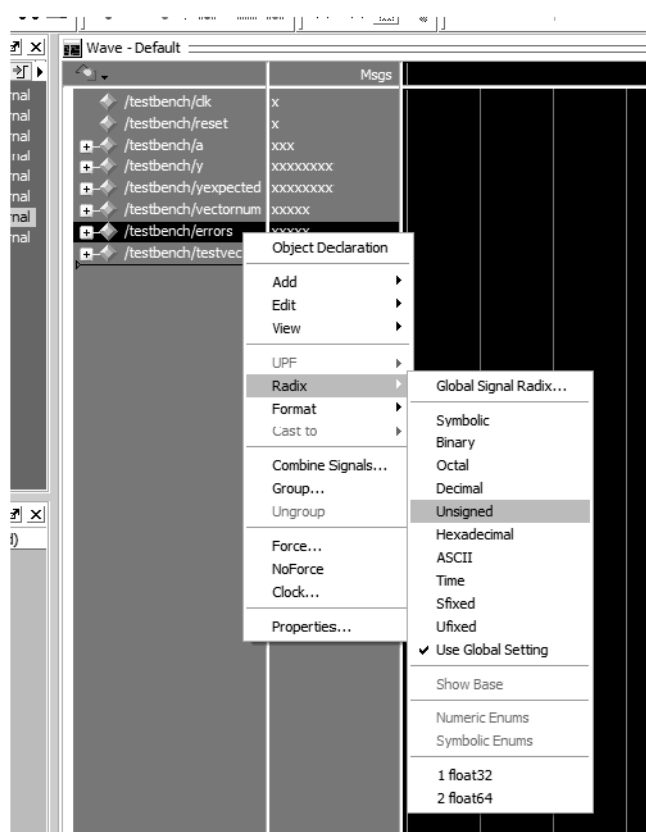


Рисунок 8.19 – Смена формата отображения сигнала

При необходимости, найдите на панели инструментов и настройте время моделирования (рисунок 8.20).

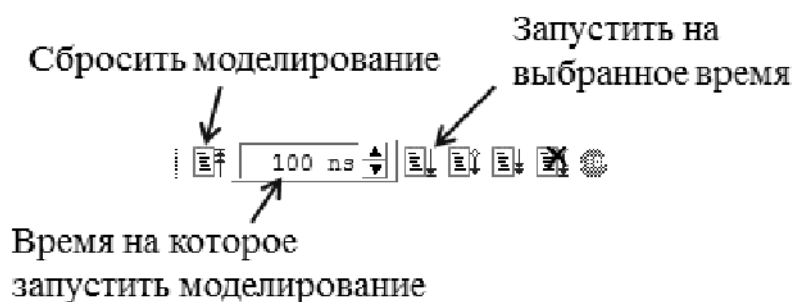


Рисунок 8.20 – Настройка времени моделирования

Запустите моделирования тестового стенда, созданного не по шаблонам, либо нажатием кнопки Run –All на панели инструментов, либо через основное меню Simulate -> Run -> Run –All, либо вводом команды run -all в консольном окне. Результат моделирования приведён на рисунке 8.21.

Масштаб отображения сигналов можно менять через контекстное меню или горячие клавиши.

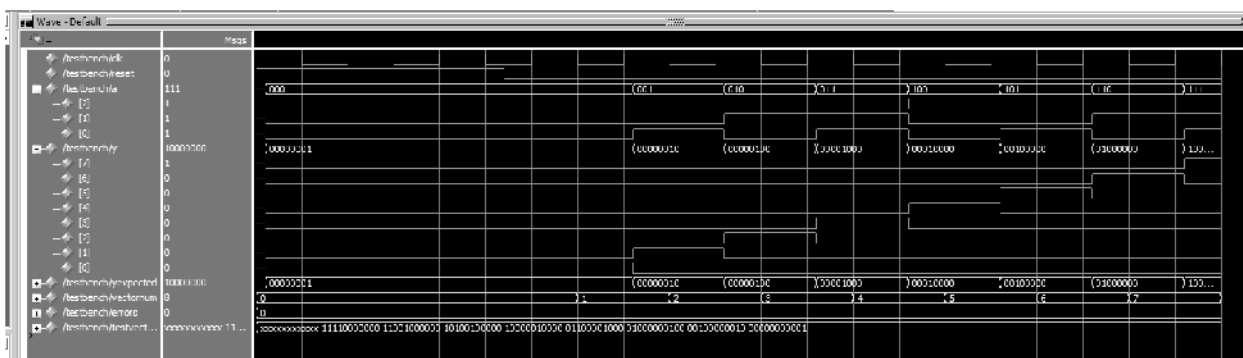


Рисунок 8.21 – Результат моделирования

В окно временных диаграмм можно добавлять маркеры с помощью панели инструментов (рисунок 8.22).



Рисунок 8.22 – Инструменты для работы с маркерами

Для просмотра структуры процессов проекта необходимо перейти на

верхний уровень иерархии в окне Structure и выбрать Add -> To Dataflow -> All items in design в контекстном меню. В окне DataFlow будет отображаться схема протекания данных, при установленном маркере Wave, его смещение будет влиять на отображение в окне DataFlow.

Окно можно сделать плавающим нажатием кнопки  в правом верхнем углу соответствующего окна.

Завершается моделирование из основного меню Simulate -> End Simulation или командой quit -sim в консольном окне.

## 2 Задание на выполнение работы.

2.1 В соответствии с заданным преподавателем вариантом создать поведенческий модуль на основе структуры case и тестовый стенд. Запустить моделирование. Минимизировать функцию (любым методом: карты Карно, Куайна-Мак-Класки, аналитически). Создать поведенческий модуль на основе минимизированной функции. Смоделировать его работу с помощью уже созданного тестового стенда. Исходные данные приведены в таблице 8.1.

Результаты выполнения задания должны быть сохранены в электронном виде (файл проекта) и в отчет (скрины): кода устройства, кода испытательного стенда, окон мастера настройки, результатов моделирования.

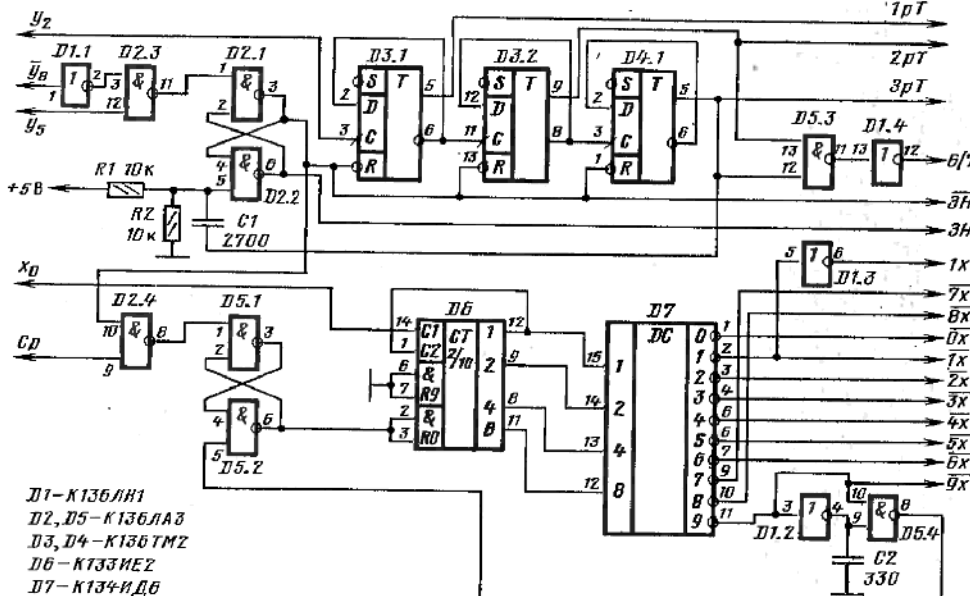
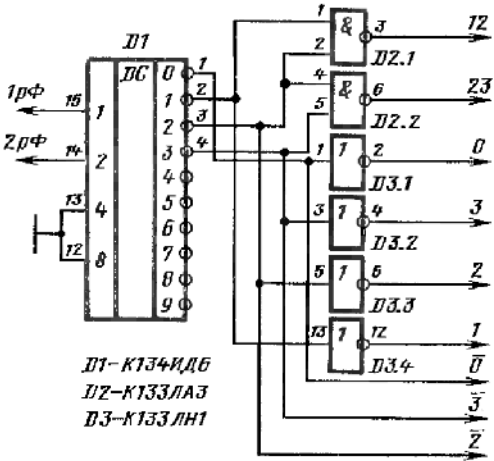
Таблица 8.1

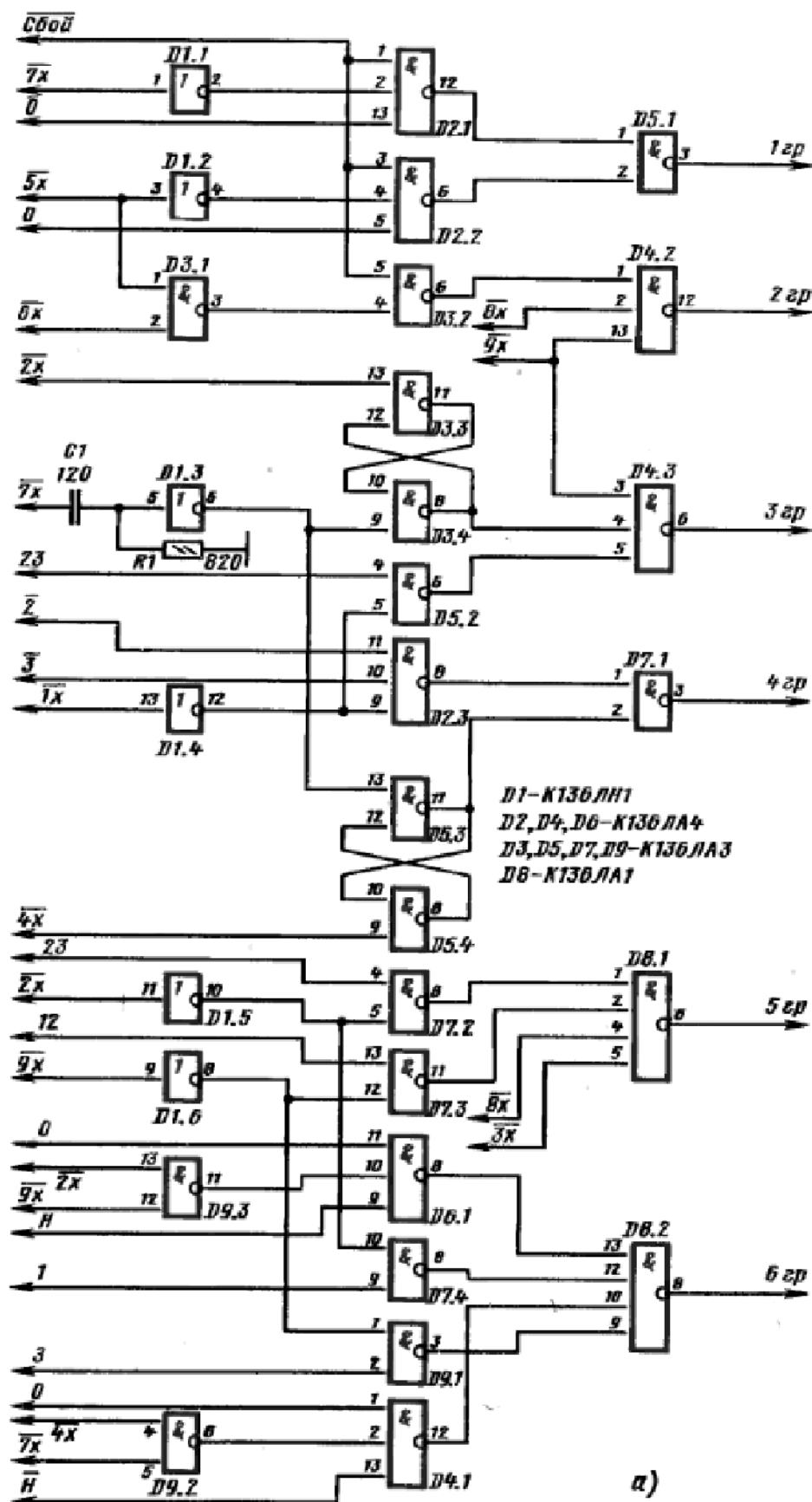
| №<br>варианта | Логическая функция                      | №<br>варианта | Логическая функция                        |
|---------------|-----------------------------------------|---------------|-------------------------------------------|
| 1             | $f = \vee(0, 4, 6, 9, 11, 13, 14, 15)$  | 11            | $f = \wedge(0, 4, 6, 9, 11, 13, 14, 15)$  |
| 2             | $f = \vee(0, 2, 5, 7, 8, 10, 13, 15)$   | 12            | $f = \wedge(0, 2, 5, 7, 8, 10, 13, 15)$   |
| 3             | $f = \vee(0, 1, 8, 9, 11, 13, 14, 15)$  | 13            | $f = \wedge(0, 1, 8, 9, 11, 13, 14, 15)$  |
| 4             | $f = \vee(2, 6, 7, 8, 9, 13, 15)$       | 14            | $f = \wedge(2, 6, 7, 8, 9, 13, 15)$       |
| 5             | $f = \vee(0, 4, 8, 10, 12, 13, 14, 15)$ | 15            | $f = \wedge(0, 4, 8, 10, 12, 13, 14, 15)$ |
| 6             | $f = \vee(1, 5, 9, 11, 13, 14, 15)$     | 16            | $f = \wedge(1, 5, 9, 11, 13, 14, 15)$     |
| 7             | $f = \vee(3, 7, 11, 13, 14, 15)$        | 17            | $f = \wedge(3, 7, 11, 13, 14, 15)$        |
| 8             | $f = \vee(2, 6, 10, 11, 13, 14, 15)$    | 18            | $f = \wedge(2, 6, 10, 11, 13, 14, 15)$    |
| 9             | $f = \vee(0, 1, 4, 5, 10, 11, 14, 15)$  | 19            | $f = \wedge(0, 1, 4, 5, 10, 11, 14, 15)$  |
| 10            | $f = \wedge(2, 3, 6, 7, 8, 9, 12, 13)$  | 20            | $f = \wedge(2, 3, 6, 7, 8, 9, 12, 13)$    |

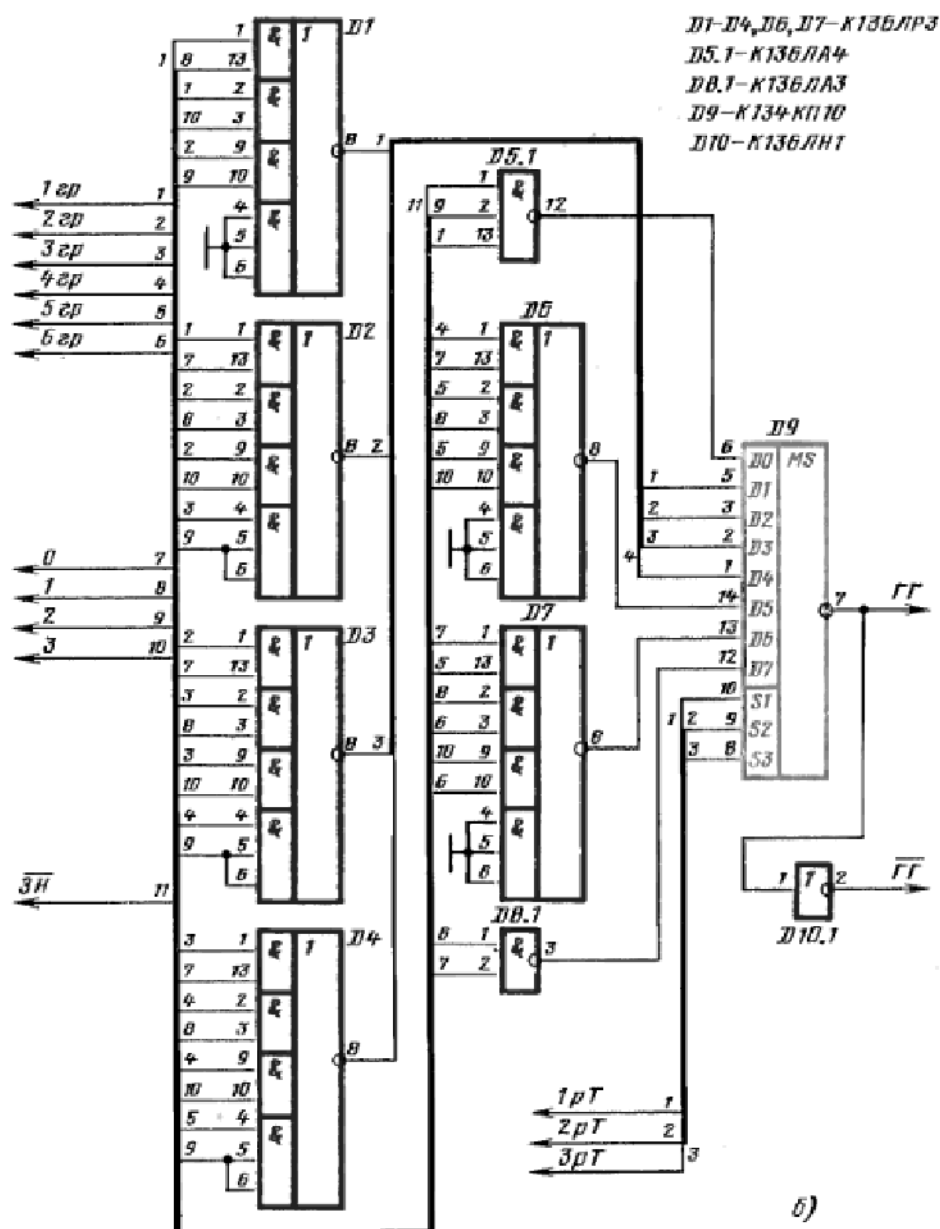
2.2 В соответствии с заданным преподавателем вариантом создать структурный и поведенческий модули в соответствии со схемой и тестовый стенд. Запустить моделирование. Убедиться, что результаты моделирования структурного и поведенческого модулей одинаковы, в противном случае найти и исправить ошибки. Исходные данные приведены в таблице 8.2.

Результаты выполнения задания должны быть сохранены в электронном виде (файл проекта) и в отчет (скрины): испытательного стенда, структурного и поведенческого модулей, результатов моделирования.

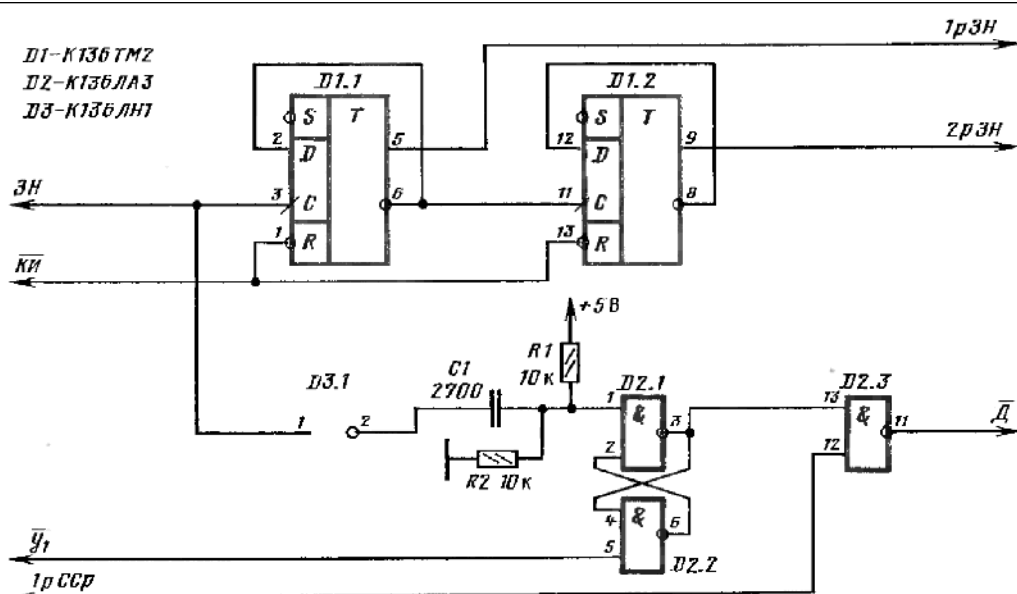
Таблица 8.2

| №<br>варианта | Схема                                                                                                                                                                                                                        |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1             |  <p> <i>D1 – К136ЛН1</i><br/> <i>D2, D5 – К136ЛА3</i><br/> <i>D3, D4 – К136ТМ2</i><br/> <i>D6 – К133НЕ2</i><br/> <i>D7 – К134НД6</i> </p> |
| 2             |  <p> <i>D1 – К134НД6</i><br/> <i>D2 – К133ЛА3</i><br/> <i>D3 – К133ЛН1</i> </p>                                                          |

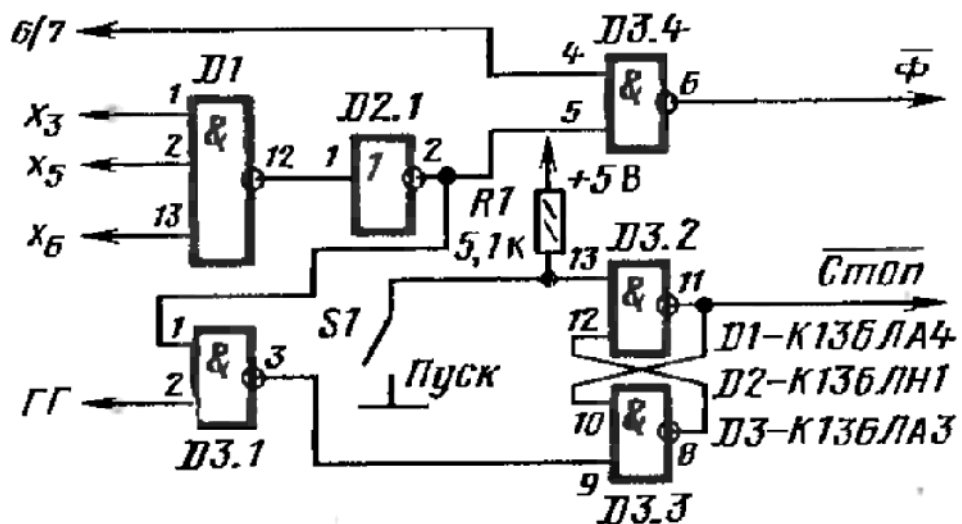




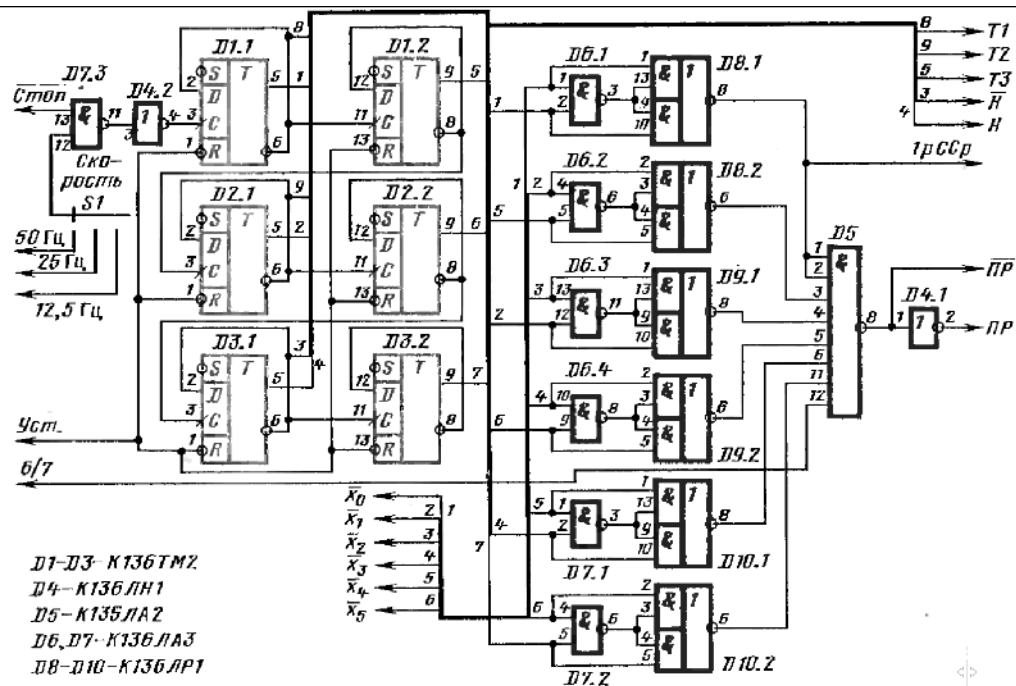
5



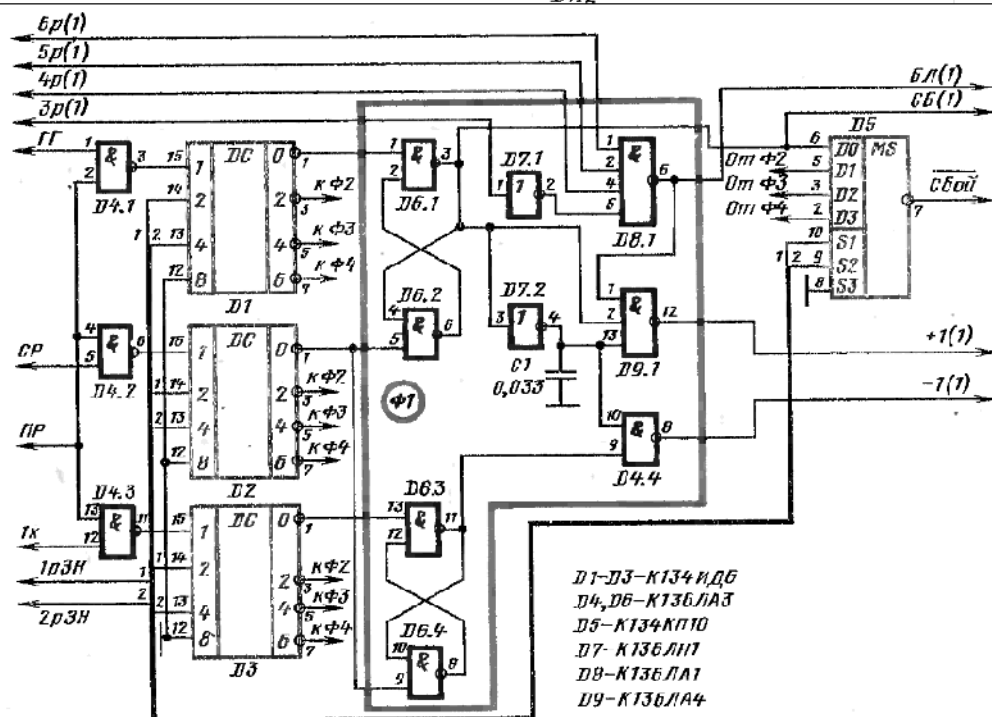
6

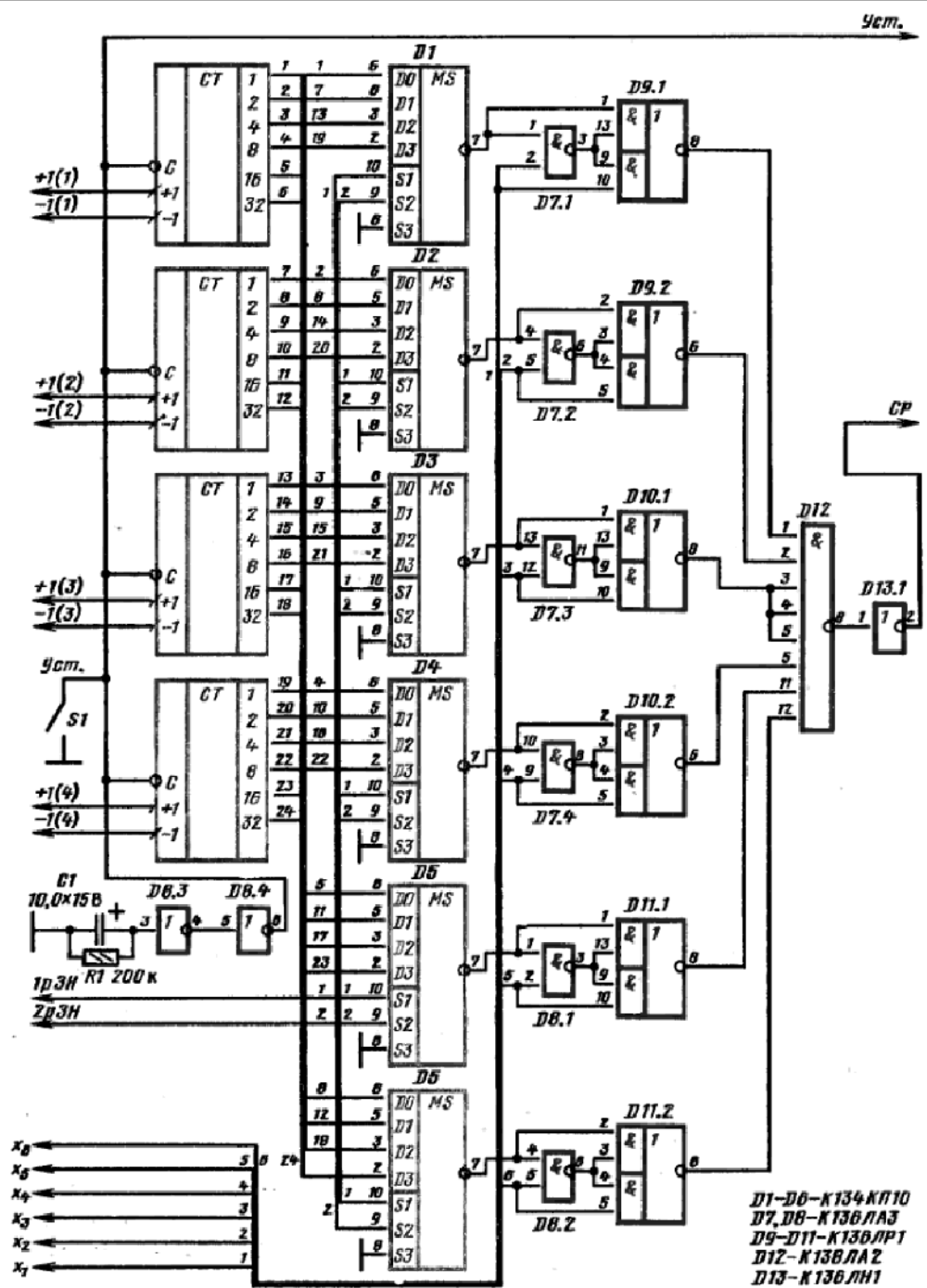


7

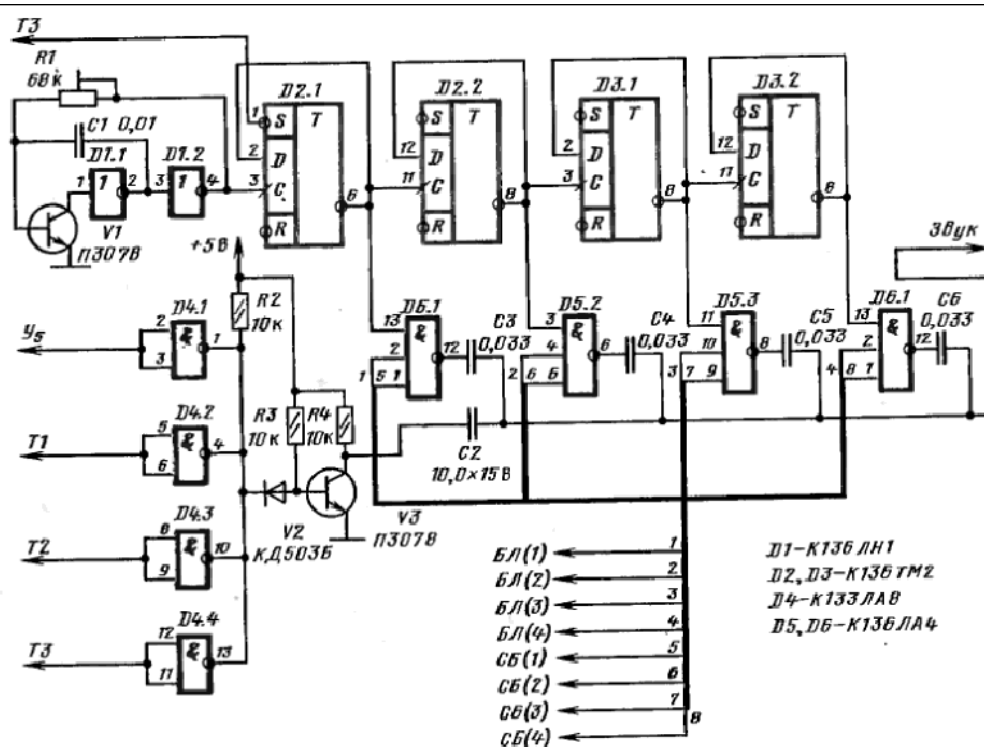


8

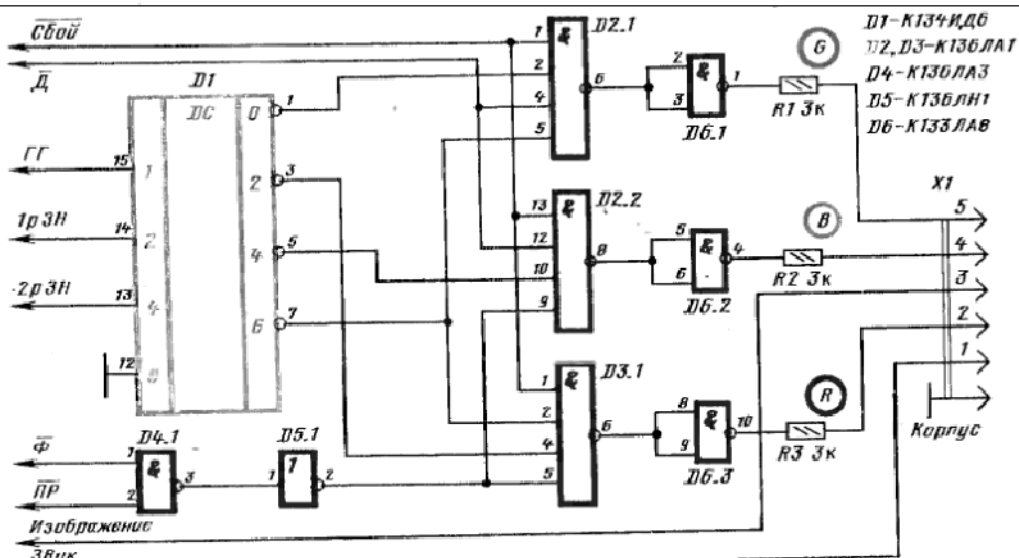


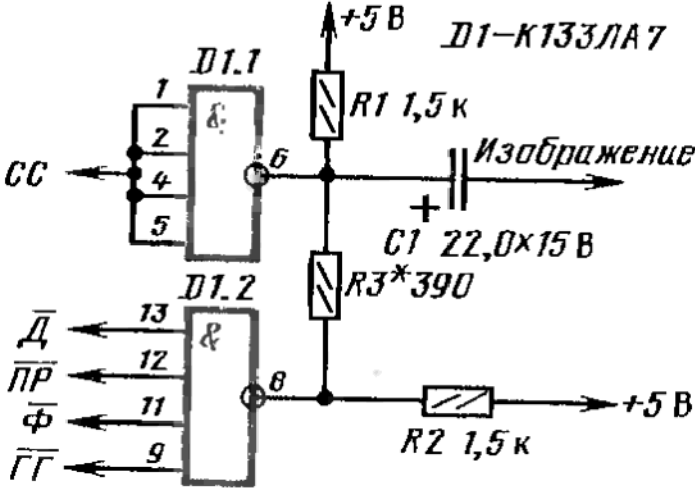
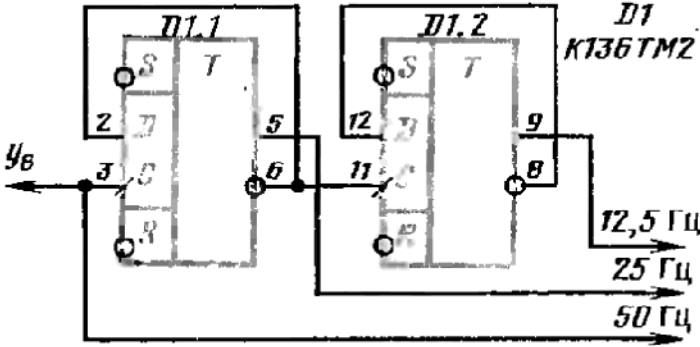


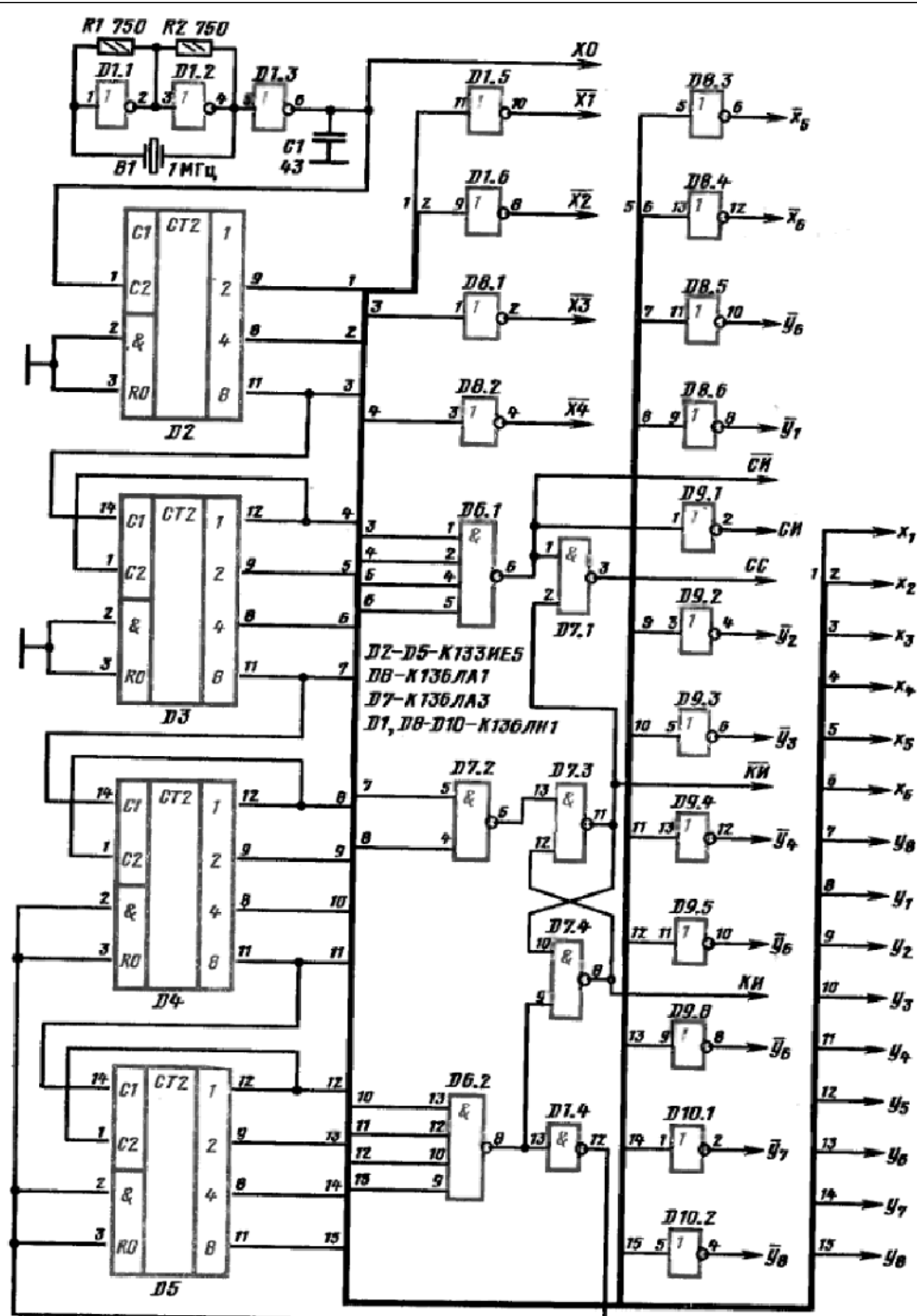
10



11



|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 12 |  <p> <math>+5\text{ В}</math> <math>D1-K133\Lambda A7</math><br/> <math>R1\ 1,5\text{ к}</math><br/> <math>CC</math><br/> <math>6</math><br/> <math>Изображение</math><br/> <math>C1\ 22,0\times 15\text{ В}</math><br/> <math>R3^*390</math><br/> <math>8</math><br/> <math>+5\text{ В}</math><br/> <math>R2\ 1,5\text{ к}</math><br/> <math>D1.1</math><br/> <math>D1.2</math><br/> <math>\bar{D}</math><br/> <math>\bar{PR}</math><br/> <math>\bar{F}</math><br/> <math>\bar{G}</math> </p> |
| 13 |  <p> <math>y_B</math><br/> <math>2</math><br/> <math>3</math><br/> <math>4</math><br/> <math>5</math><br/> <math>6</math><br/> <math>8</math><br/> <math>12,5\text{ Гц}</math><br/> <math>25\text{ Гц}</math><br/> <math>50\text{ Гц}</math><br/> <math>D1.1</math><br/> <math>D1.2</math><br/> <math>D1\ K136TM2</math><br/> <math>\bar{D}</math><br/> <math>\bar{PR}</math><br/> <math>\bar{F}</math><br/> <math>\bar{G}</math> </p>                                                        |



### 3 Задание на обработку данных.

3.1 Сделать выводы о методике создания проекта в Modelsim.

3.2 Сделать выводы о создании поведенческого модуля на основе структуры case и тестового стенда. Выполнить анализ результатов моделирования.

3.3 Сделать выводы о создании структурного и поведенческого

модулей в соответствии со схемой и тестового стенда. Выполнить анализ результатов моделирования.

### **Содержание отчета**

Отчет по лабораторной работе выполняется на листах формат А4 согласно требований ГОСТ и ЕСКД и должен содержать следующие пункты:

- а) титульный лист с названием работы, дисциплины, кода группы, ф.и.о. исполнителя и преподавателя;
- б) цель и содержание работы;
- в) краткое описание программного обеспечения, необходимого для выполнения работы;
- г) скрины: кода устройства, кода испытательного стенда, окон мастера настройки, результатов моделирования по п. 2.1;
- д) скрины: испытательного стенда, структурного и поведенческого модулей, результатов моделирования по п. 2.2;
- е) результаты выполнения п. 3;
- ж) выводы по результатам работы.

### **Контрольные вопросы**

1.

**Список литературы, рекомендованный к использованию по данной теме**

#### **Основная учебная литература**

## **Дополнительная литература**

## ПРАКТИЧЕСКАЯ РАБОТА 9

**Тема:** Изучение основ моделирования в среде программирования Altera Quartus II.

**Цель работы:** закрепление знаний, полученных на лекциях по разделу 4 (лекция 9). В ходе выполнения лабораторной работы необходимо:

- ознакомиться с возможностями среды программирования Altera Quartus II;
- изучить аппаратные средства отладочной платы ПЛИС Cyclone IV;
- выполнить тестирование аппаратной и программной части с использованием контрольного примера;
- создать проект согласно вариантам задания и проверить его работоспособность.

**Формируемые компетенции:** ОПК-9.

### Теоретическая часть

Программируемая логическая интегральная схема – электронный компонент, используемый для создания цифровых интегральных схем. В отличие от обычных цифровых микросхем, логика работы ПЛИС не определяется при изготовлении, а задаётся посредством программирования (проектирования). Для программирования используются программатор и IDE (отладочная среда), позволяющие задать желаемую структуру цифрового устройства в виде принципиальной электрической схемы или программы на специальных языках описания аппаратуры: Verilog, VHDL, AHDL и др.

Именно поэтому важным является изучение аппаратной платформы отладочной платы ПЛИС, получение навыков работы с ней, а также навыков работы в среде программирования.

## **Оборудование и материалы**

Лабораторная работа выполняется с использованием среды программирования Altera Quartus II и аппаратных средств отладочной платы ПЛИС Cyclone IV.

## **Указания по технике безопасности**

Перед проведением лабораторной работы следует изучить инструкцию по технике безопасности:

- при работе с отладочной платой ПЛИС Cyclone IV;
- при работе на компьютере.

**ВНИМАНИЕ!** Если в варианте задания используются внешние по отношению к отладочной плате устройства, то все работы по сборке схемы, изменению порядка внешних соединений, подключению измерительных приборов должны производиться **ТОЛЬКО** при отключенном напряжении источника питания и/или отключенном кабеле USB, соединяющем плату с ПК.

Включение напряжения питания необходимо производить только после проверки преподавателем правильности коммутаций.

При подключении внешних устройств **НЕОБХОДИМО** проверять уровни напряжений питания и сигналов во избежание повреждения внешних устройств и отладочной платы.

Также следует соблюдать правила и требования, установленные для данного компьютерного класса.

## **Задания**

- 1 Задание на подготовку к выполнению работы.

### 1.1 Изучение аппаратных средств отладочной платы ПЛИС Cyclone IV.

Внешний вид используемой в работе отладочной платы приведен на рисунке 9.1, схема электрическая принципиальная на рисунке 9.2.

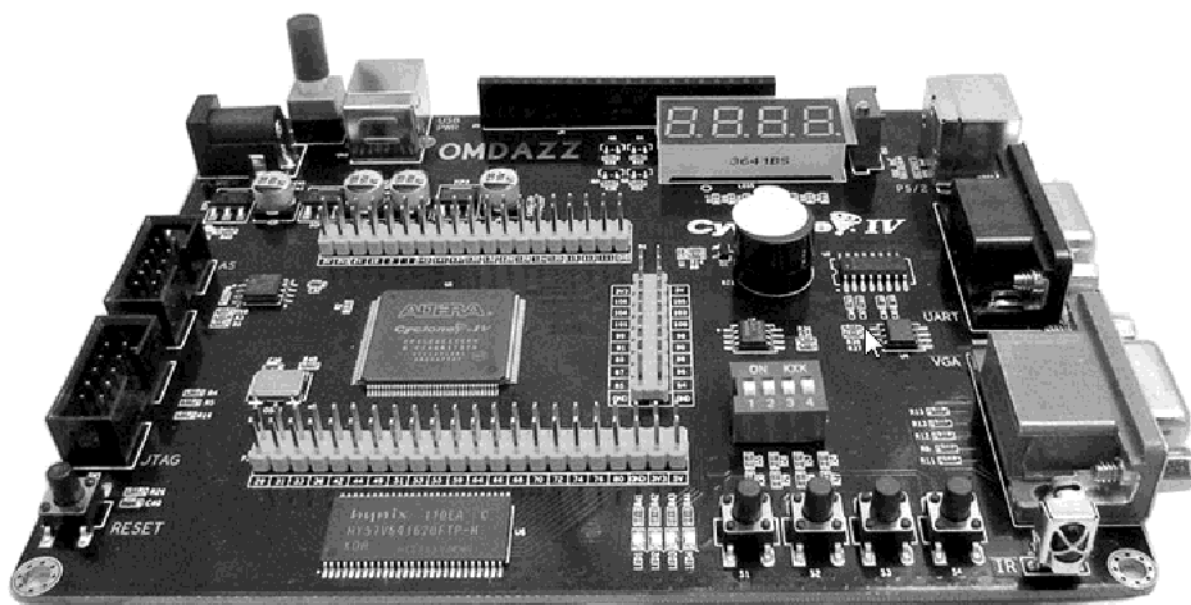


Рисунок 9.1 – Внешний вид отладочной платы ПЛИС Cyclone IV

Рисунок 9.2 – Схема электрическая принципиальная отладочной платы ПЛИС Cyclone IV

Отладочная плата ПЛИС Cyclone IV содержит:

- ПЛИС Altera Cyclone IV EP4CE10E22C8;
- 4-разрядный семисегментный индикатор (Digital Tube) с возможностью изменения яркости свечения подстроечным резистором (LCD contrast adjustment);
- разъём PS/2 для подключения клавиатуры/мыши;
- COM-порт (RS-232 interface) сопряжение с которым выполнено на микросхеме SP3232;
- температурный датчик LM754A;
- спикер (Buzzer);
- 4 DIP переключателя (DIP switch);

- VGA интерфейс;
- ИК-приёмник (Infrared receiver);
- 4 кнопки;
- микросхему последовательной энергонезависимой памяти AT24C08 (8 кБ), подключенную по интерфейсу I2C;
- 4 светодиода (LEDs);
- микросхему статического ОЗУ на 64 Мбит (SDRAM);
- кнопку сброса (RESET key);
- разъём JTAG интерфейса программирования;
- кварцевый резонатор на 50 МГц (50 M active crystal);
- микросхема flash-памяти на 16 Мбит (M25P16 Flash);
- разъём AS интерфейса (AS download interface);
- интегральные стабилизаторы напряжений на 3,3; 2,5 и 1,2 В;
- разъём для 5 В источника питания;
- кнопка включения питания (Power switch);
- разъём USB с поддержкой питания (USB power supply);
- разъём для дисплейного модуля (LCD 12864/1602 Serial AD DA);
- штырьковые коннекторы для подключения дополнительных периферийных устройств (I/O pins for Expansion board).

## 1.2 Изучение основ создания проектов в среды программирования Altera Quartus II.

Выполнить проект контрольного примера, в котором необходимо создать 4-разрядный двоичный счётчик с выводом текущего состояния на четыре светодиода отладочной платы.

Запустите приложение Altera Quartus II и создайте новый проект: File-> New -> New Quartus Prime Project (рисунок 9.3).

Откроется окно мастера создания нового проекта (рисунок 9.4). При первом запуске начальной вкладкой будет «Introduction», её можно отключить для последующих запусков мастера проекта установкой флага

«Don't show me this introduction again».

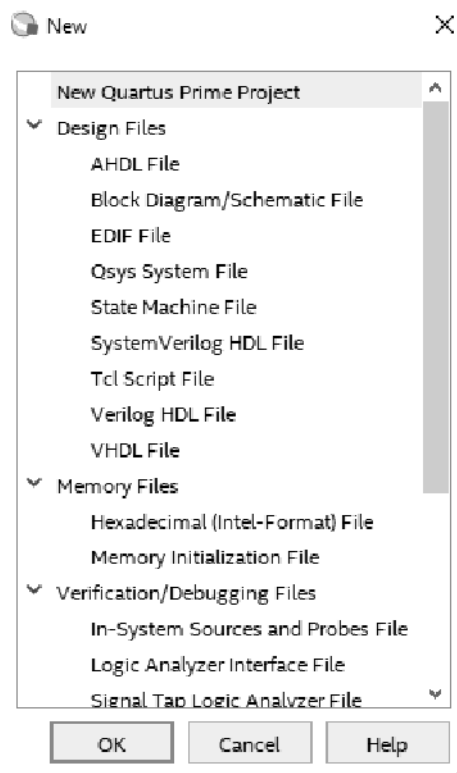


Рисунок 9.3 – Окно выбора типа файла

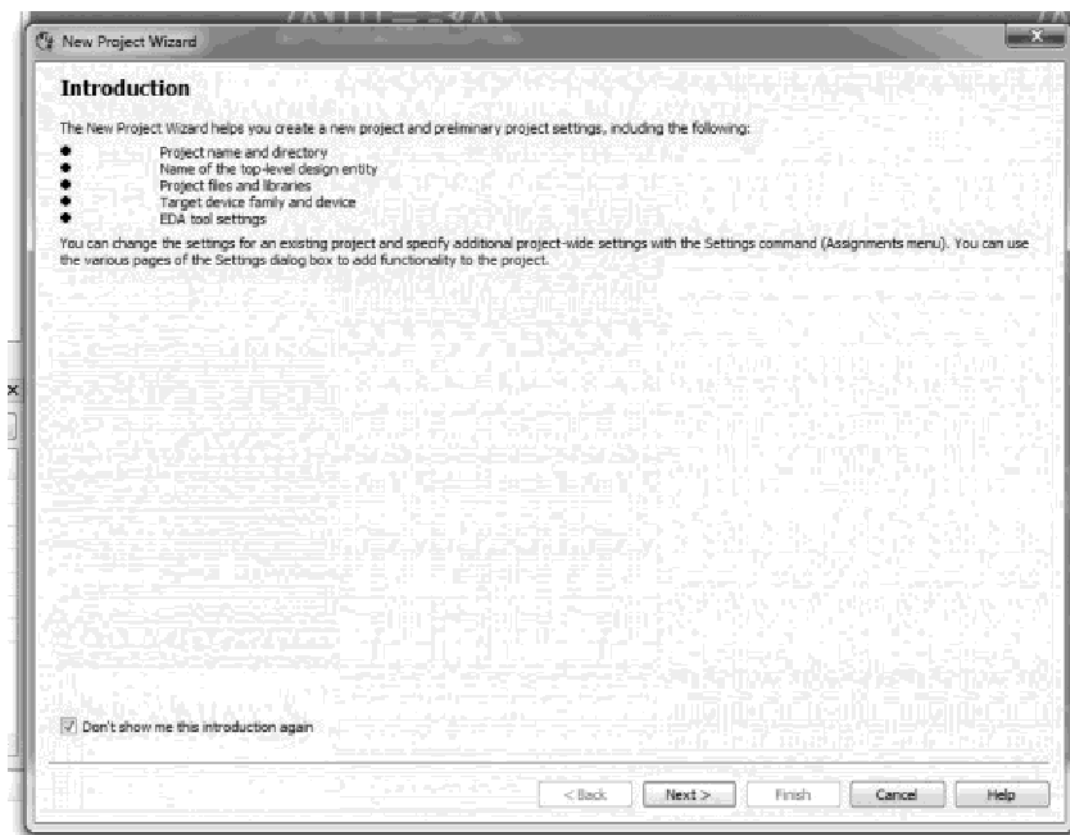


Рисунок 9.4 – Начальная вкладка мастера создания проектов

В следующей вкладке (рисунок 9.5) укажите путь к папке проекта и его название (Counter – счётчик). В окне выбора типа проекта (рисунок 9.6) укажите поле «Empty project» (Пустой проект).

The screenshot shows the 'New Project Wizard' window with the title bar 'New Project Wizard' and a close button 'X'. The main heading is 'Directory, Name, Top-Level Entity'. Below it, there are three text input fields, each with a browse button '...':

- First field: 'What is the working directory for this project?' with the text 'D:/altera/17.1/Project/Counter'.
- Second field: 'What is the name of this project?' with the text 'Counter'.
- Third field: 'What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.' with the text 'Counter'.

Below the third field is a button labeled 'Use Existing Project Settings...'. At the bottom of the window are five buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

Рисунок 9.5 – Окно выбора пути к создаваемому проекту и его наименования

The screenshot shows the 'New Project Wizard' window with the title bar 'New Project Wizard' and a close button 'X'. The main heading is 'Project Type'. Below it, there is a label 'Select the type of project to create.' followed by two radio button options:

- ☒ Empty project  
Create new project by specifying project files and locations, target device family and device, and EDA tool settings.
- ☐ Project template  
Create a project from an existing design template. You can choose from design templates installed with the Quartus Prime software, or download design templates from the Design Store.

At the bottom of the window are five buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

Рисунок 9.6 – Окно выбора типа проекта

Так как проект создаётся с «нуля», то в следующей вкладке (рисунок 9.7) оставляем всё без изменений (файлы не добавляем).

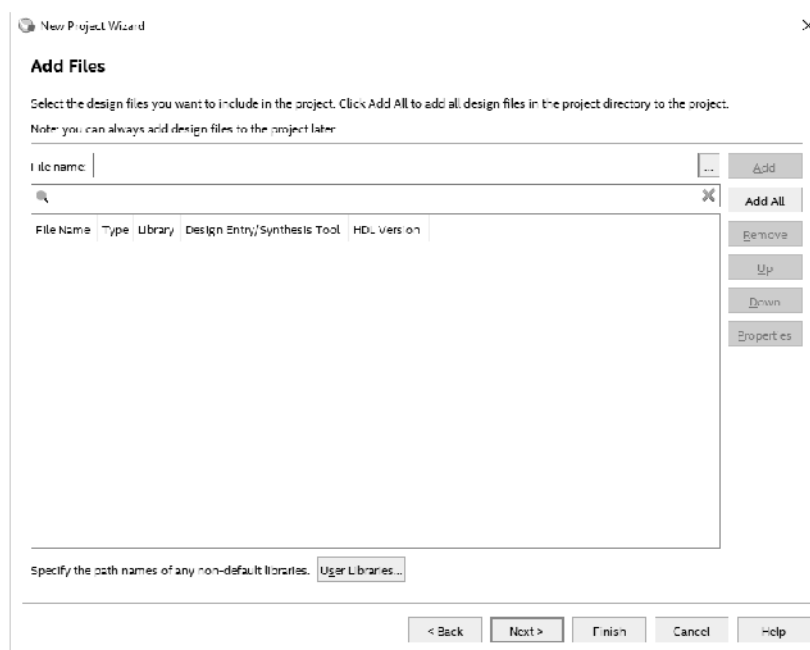


Рисунок 9.7 – Окно добавления файлов в проект

На вкладке выбора устройства (рисунок 9.8) необходимо выбрать семейство и наименование ПЛИС для отладочной платы. В нашем случае это семейство Cyclone IV E и ПЛИС EP4CE10E22C8. Снимите флаг «Show advanced devices».

Следующая вкладка (рисунок 9.9) позволяет настроить моделирование проекта, если это необходимо (можете оставить без изменений).

Последняя вкладка мастера создания проекта (рисунок 9.10) содержит итоговые сведения по проекту. Завершите выполнение мастера (Finish).

Создайте файл верхнего уровня проекта. Для этого выберите File -> New -> Block Diagram / Schematic File в окне выбора типа файла. Сохраните файл под именем Counter.bdf.

Для счётчика понадобятся следующие сигналы: CLK – тактовый сигнал, RSTn – сброс и четыре сигнала управления светодиодами LED\_Out [3:0], объединенные в 4-проводную шину. Добавить порты можно с

ПОМОЩЬЮ ПАНЕЛИ ИНСТРУМЕНТОВ, КАК ПОКАЗАНО НА РИСУНКЕ 9.11.

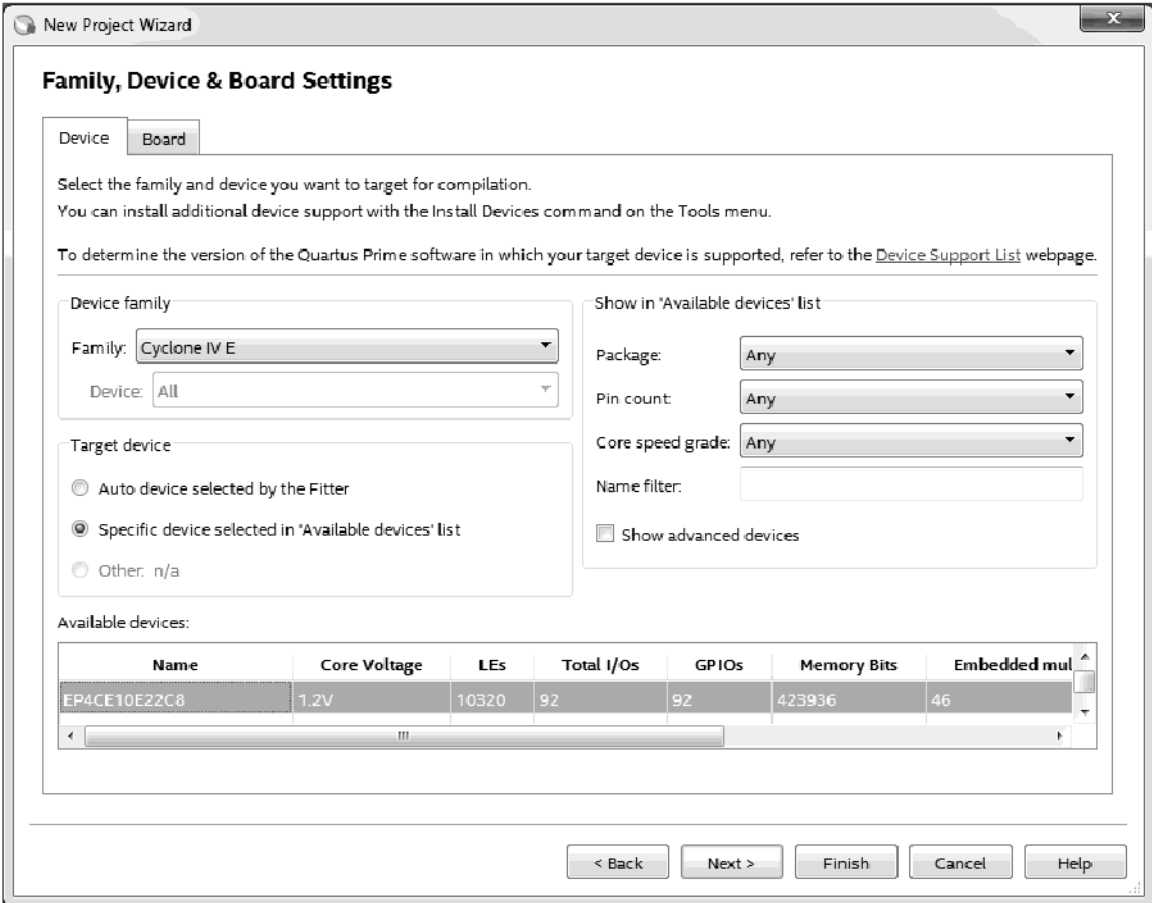


Рисунок 9.8 – Вкладка выбора устройства

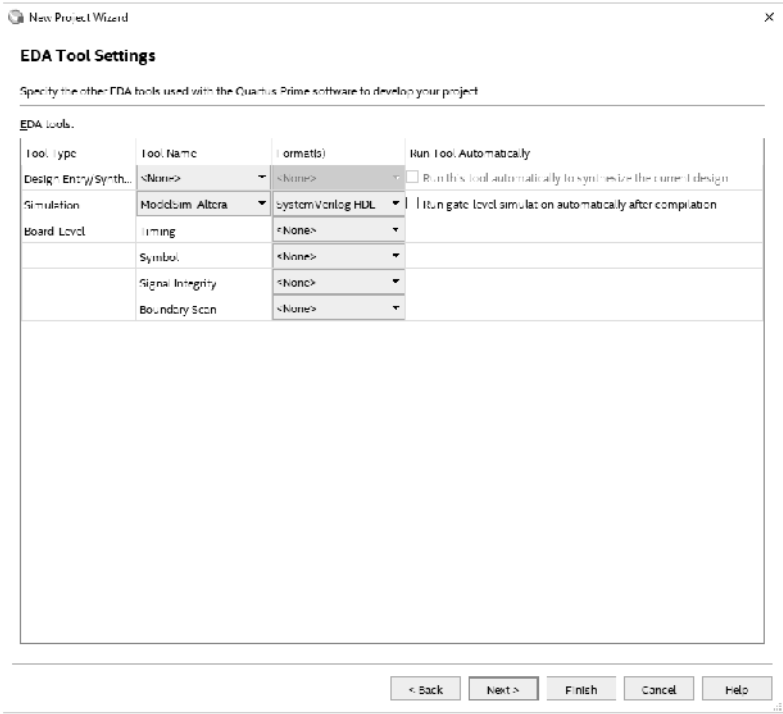


Рисунок 9.9 – Окно настройки для моделирования проекта

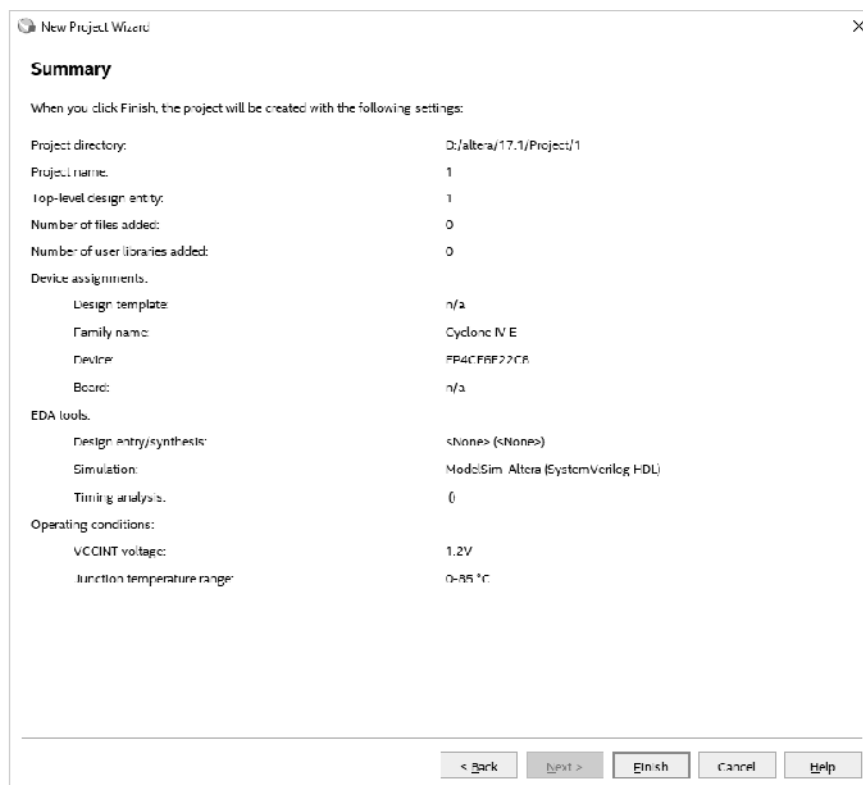


Рисунок 9.10 – Итоговая вкладка мастера создания нового проекта

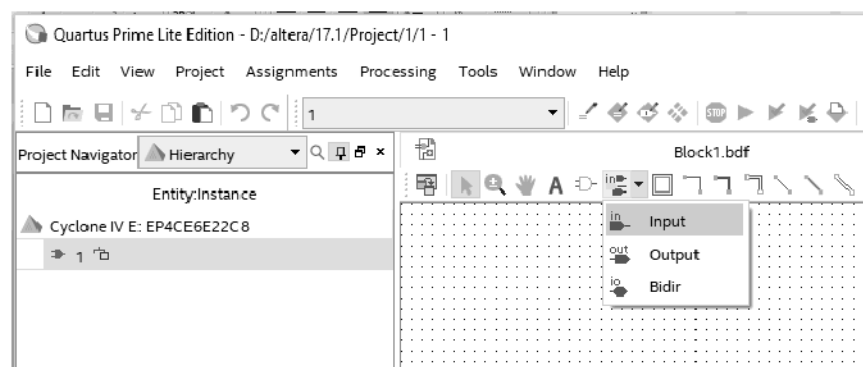


Рисунок 9.11 – Добавление портов ввода/вывода на схему

Произвольно расположите порты и переименуйте их как показано на рисунке 9.12.

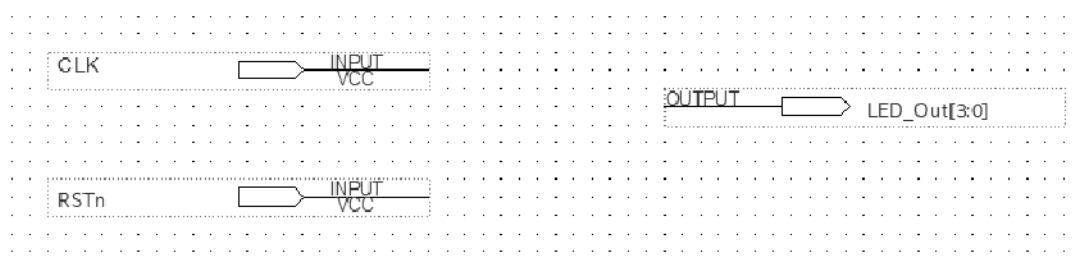


Рисунок 9.12 – Порты ввода/вывода

Следующим этапом необходимо создать счётчик. Для этого можно использовать стандартные компоненты из библиотеки Quartus или из ранее созданных проектов, а также возможно создать свой компонент. Создайте собственный компонент. Для чего выберите новый Verilog-файл: File -> New -> Verilog HDL File и сохраните его как Counter.v. Введите текст программы (рисунок 9.13).

```

1 module count(input wire CLK,
2 input wire RST,
3 output wire [3:0] LED_Out);
4
5 reg [27:0] count; // Counter for clock divider
6
7 always@(posedge CLK or negedge RST)
8 begin
9 if (!RST)
10 count <= 0;
11 else
12 count <= count + 1;
13 end
14
15 assign LED_Out = count [27:24]; // Connecting 4 last digits of counter to LEDs
16
17 endmodule
18

```

Рисунок 9.13 – Файл Verilog для счётчика

Частота кварцевого резонатора установленного на плате 50 МГц, чтобы индикация была заметна, необходимо добиться низкой частоты на входе проектируемого счётчика (примерно от 1 до 3 Гц), предварительно разделив частоту кварца. Поэтому в модуле определён 28-разрядный счётчик-делитель, у которого только четыре старших разряда будут отображаться на светодиодах.

$$f = \frac{f_{\text{кварц}}}{2^n} = \frac{50 \cdot 10^6}{2^{24}} \approx 2,98 \text{ Гц},$$

где  $f$  – частота поступающая на вход счётчика, Гц;

$f_{\text{кварц}}$  – частота кварцевого резонатора, Гц;

$n$  – число разрядов счётчика-делителя ( $28 - 4 = 24$ ).

Полученный модуль необходимо преобразовать в схемный компонент. Для этого в «Project Navigator», во вкладке «Files» щелчком правой кнопкой

на файле Counter.v выбираем из контекстного меню «Create Symbol Files for Current File» (рисунок 9.14).

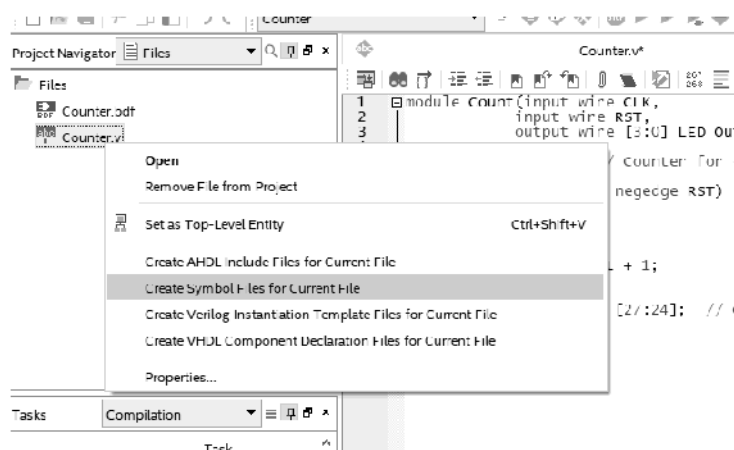


Рисунок 9.14 – Создание схемного компонента

Затем добавьте синтезированный символ в файл Counter.bdf (рисунок 9.15) и соедините выводы (рисунок 9.16).

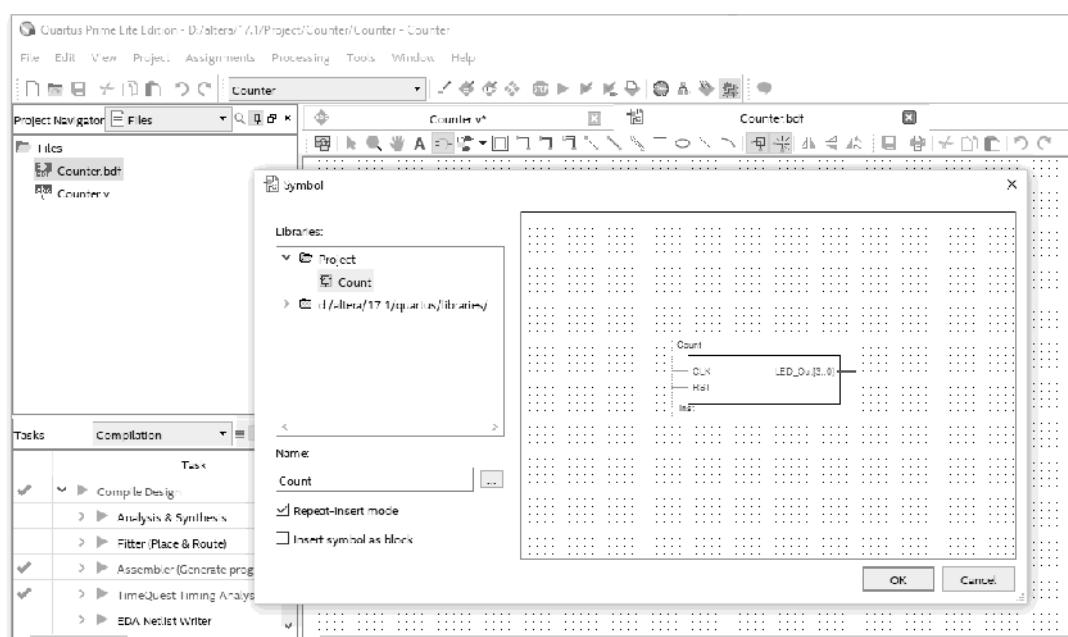


Рисунок 9.15 – Добавление символа на схему

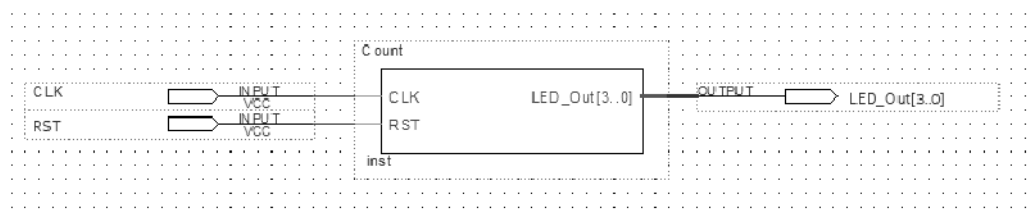


Рисунок 9.16 – Схема проекта

Сохраните проект и запустите компиляцию проекта (рисунок 9.17).

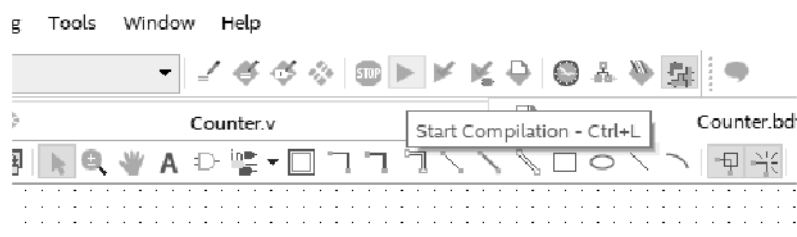


Рисунок 9.17 – Запуск компиляции проекта

Процесс компиляции проекта займёт некоторое время. О прогрессе выполнения компиляции можно судить по индикаторам (рисунок 9.18).

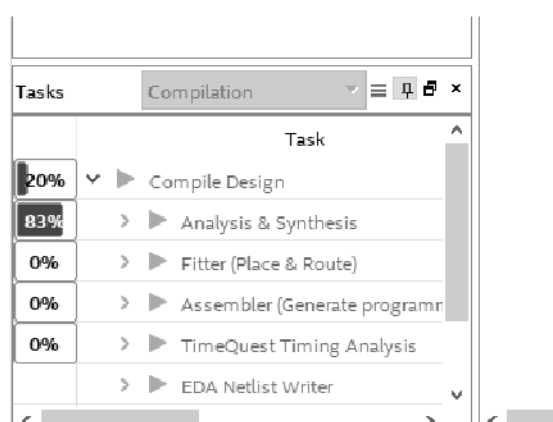


Рисунок 9.18 – Прогресс компиляции

При успешном прохождении компиляции окно может выглядеть так, как показано на рисунке 9.19.

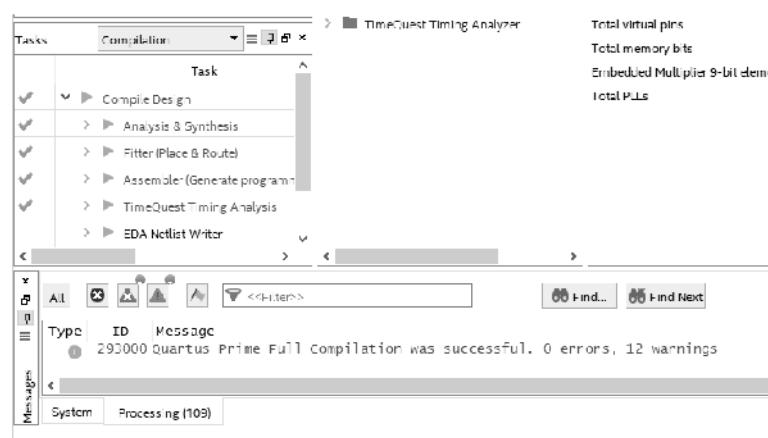


Рисунок 9.19 – Фрагмент окна программы Quartus при успешном завершении компиляции

Прежде чем производить запись проекта в ПЛИС, необходимо указать программе назначение выводов. Для этого запустите Pin Planner (рисунок 9.20).

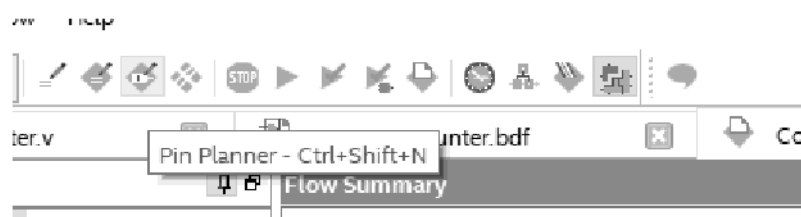


Рисунок 9.20 – Инструмент Pin Planner

В открывшемся окне (рисунок 9.21) необходимо назначить сигналам проекта соответствующие выводы.

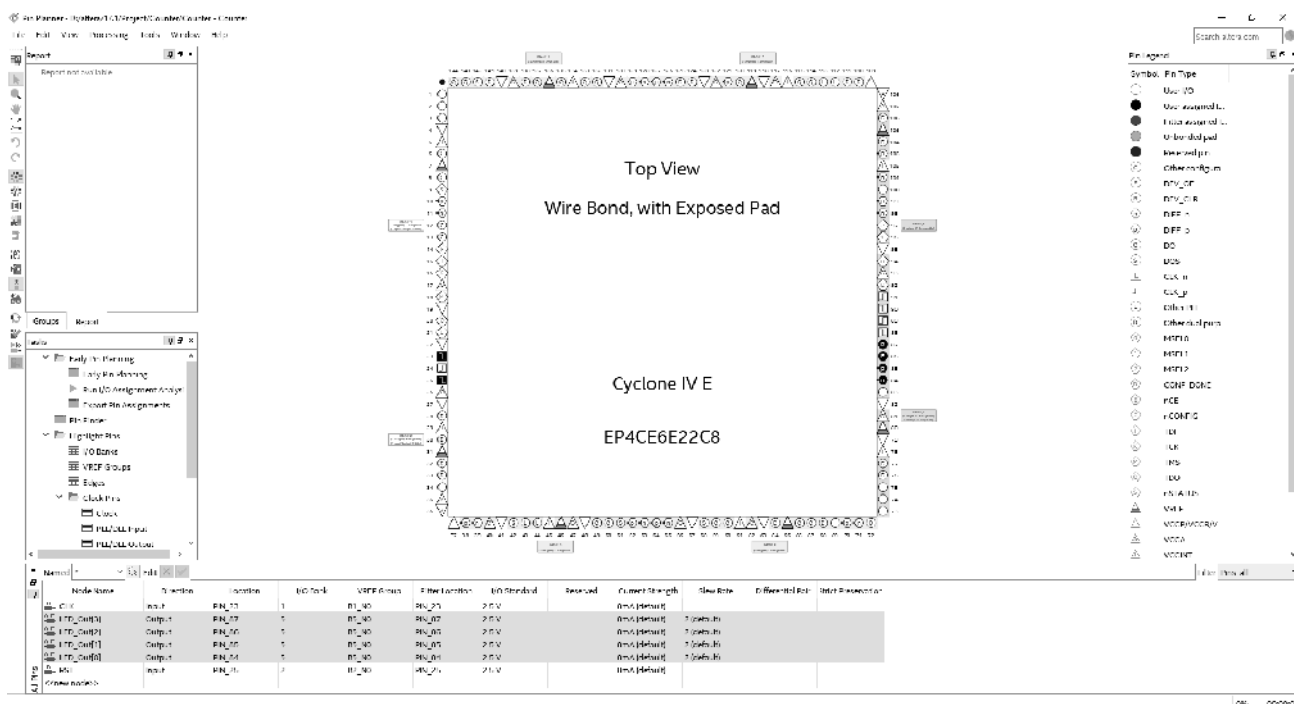


Рисунок 9.21 – Окно назначения выводов

Выводы должны быть присвоены в соответствии с технической документацией на отладочную плату. Откройте файл «Development board pin information.xls». В нём указаны подключения выводов на отладочной плате. Установите для портов ввода/вывода проекта соответствующие номера выводов. Закройте окно назначения выводов. Сохраните проект и повторно

запустите компиляцию.

Подключите плату и программатор. Запустите программатор (рисунок 9.22).



Рисунок 9.22 – Инструмент работы с программатором

Откроется окно программатора (рисунок 9.23). Если настройки верны, нажмите Start.

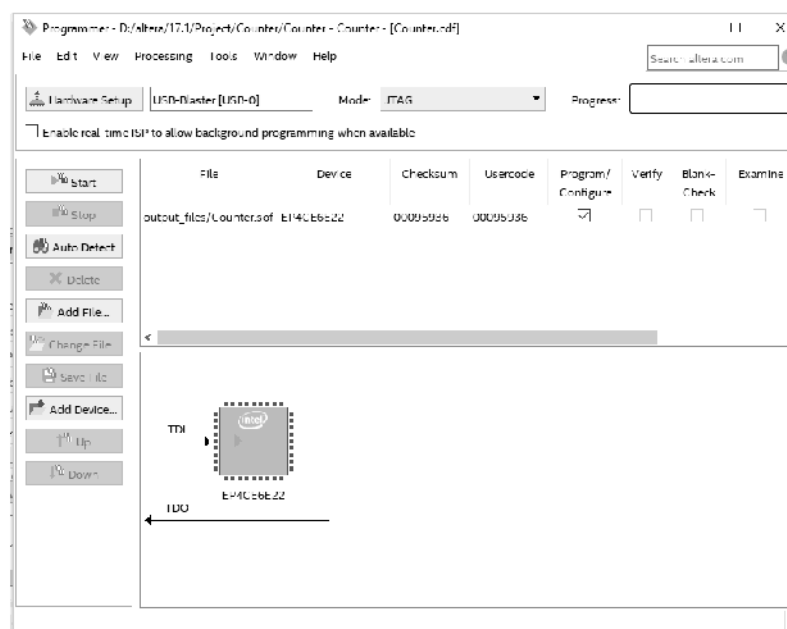


Рисунок 9.23 – Окно программатора

После прошивки убедитесь в том, что работа счётчика верна.

## 2 Задание на выполнение работы.

Выполнить проект заданного устройства с использованием среды программирования Altera Quartus II. В случае удачной компиляции, подключить отладочную плату ПЛИС Cyclone IV и программатор. Произвести прошивку и проверить работоспособность проекта. В случае

необходимости, выполнить исправление ошибок и отладку.

Тип устройства выбирается в соответствии с номером варианта, заданным преподавателем. Исходные данные приведены в таблице 9.1.

Таблица 9.1

| Вариант | Задание                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | Спроектировать двоично-десятичный счётчик с выводом текущего состояния на 4-х разрядный семисегментный индикатор с использованием одного разряда.                                                                                                                                                                                                                                                                                     |
| 2       | Спроектировать двоично-десятичный счётчик с выводом текущего состояния на 4-х разрядный семисегментный индикатор с использованием 4-х разрядов.                                                                                                                                                                                                                                                                                       |
| 3       | Спроектировать схему работы со светодиодами, реализующую 3 режима работы: попарное включение/выключение соседних светодиодов (1-2, 3-4), попарное включение/выключение светодиодов 1-3, 2-4, попарное включение/выключение светодиодов 1-4, 2-3. Режим изменяется по нажатию кнопки.                                                                                                                                                  |
| 4       | Спроектировать схему работы со светодиодами, реализующую 3 режима работы: бегущие огни (при включении следующего светодиода гаснет, предыдущий), последовательное включение светодиодов в цикле двух видов: включается следующий светодиод без отключения предыдущего, цикл повторяется с полностью погашенных светодиодов и во втором варианте светодиоды гасятся в обратной последовательности. Режим изменяется по нажатию кнопки. |
| 5       | Спроектировать схему не стандартного использования семисегментных индикаторов – бегущие огни. Поочередно включаются сегменты индикатора по часовой или против часовой стрелки. Направление выбирается нажатием клавиши.                                                                                                                                                                                                               |
| 6       | Спроектировать схему реверсивного счётчика нажатия клавиш с выводом показаний на семисегментный индикатор. Нажатие на S1 увеличивает значение на 1, на S2 – уменьшает значение на 1.                                                                                                                                                                                                                                                  |
| 7       | Спроектировать схему включения/выключения светодиодов нажатием кнопок. Каждой кнопке соответствует свой светодиод. Каждое нажатие меняет состояние светодиода на противоположное.                                                                                                                                                                                                                                                     |
| 8       | Спроектировать схему преобразователя двоичного кода (отображается на светодиодах) в десятичный (отображается на семисегментном индикаторе). Двоичная комбинация задаётся кнопками.                                                                                                                                                                                                                                                    |
| 9       | Спроектировать схему не стандартного использования семисегментных индикаторов – «заряд батареи». Поочередно включаются вертикальные сегменты 4-х разрядного семисегментного индикатора. Скорость (3-4 режима) заполнения меняется нажатием клавиши.                                                                                                                                                                                   |
| 10      | Спроектировать схему работы со светодиодами, реализующую 2 режима работы бегущие огни «стенка»: горит один светодиод (порядок 1-2-3-4-3-2-1), то же самое, но один светодиод погашен. Режим изменяется по нажатию кнопки.                                                                                                                                                                                                             |
| 11      | Спроектировать схему не стандартного использования семисегментных                                                                                                                                                                                                                                                                                                                                                                     |

|    |                                                                                                                                                                                                                                     |
|----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|    | индикаторов – бегущие огни «стенка». Поочередно включаются/гасятся пары вертикальных сегментов 4-х разрядного семисегментного индикатора.                                                                                           |
| 12 | Спроектировать схему преобразователя двоичного кода (отображается на светодиодах) в восьмеричный (отображается на семисегментном индикаторе). Двоичная комбинация задаётся кнопками.                                                |
| 13 | Спроектировать схему преобразователя двоичного кода (отображается на светодиодах) в шестнадцетиричный (отображается на семисегментном индикаторе). Двоичная комбинация задаётся кнопками.                                           |
| 14 | Спроектировать схему сумматора для двух десятичных значений, отображаемых на 1 и 2 разряде 4-х разрядного семисегментного индикатора, результат суммирования должен выводиться на 3 и 4 разряды. Значения задаются кнопками.        |
| 15 | Спроектировать схему сумматора для двух шестнадцетиричных значений, отображаемых на 1 и 2 разряде 4-х разрядного семисегментного индикатора, результат суммирования должен выводиться на 3 и 4 разряды. Значения задаются кнопками. |
| 16 | Спроектировать схему сумматора для двух восьмеричных значений, отображаемых на 1 и 2 разряде 4-х разрядного семисегментного индикатора, результат суммирования должен выводиться на 3 и 4 разряды. Значения задаются кнопками.      |
| 17 | Спроектировать схему умножителя для двух десятичных значений, отображаемых на 1 и 2 разряде 4-х разрядного семисегментного индикатора, результат произведения должен выводиться на 3 и 4 разряды. Значения задаются кнопками.       |
| 18 | Спроектировать схему асинхронного реверсивного сдвигового регистра. Направление сдвига и записываемые значения задаются кнопками.                                                                                                   |

В качестве доказательства выполненного задания необходимо в отчет приводить:

- скрин файла Verilog или его непосредственный текст;
- скрин схемы проекта;
- скрин фрагмента окна программы Quartus при успешном завершении компиляции;
- скрин окна назначения выводов;
- скрины результатов выполнения в виде графических результатов моделирования из окна Simulation Waveform отчета моделирующей программы;
- проект контрольного примера и выполненного согласно варианта должны быть сохранены в программном обеспечении Altera Quartus II для демонстрации при защите.

3 Задание на обработку данных.

3.1 Сделать выводы о методике установки выводов.

3.2 Сделать выводы о назначении и методике создания файла Verilog.

3.3 Сделать выводы о методике создания схемного компонента.

3.4 Сделать выводы о методике назначения выводов.

3.5 Сделать выводы о полученных результатах работы разработанного проекта устройства согласно варианту.

### **Содержание отчета**

Отчет по лабораторной работе выполняется на листах формат А4 согласно требований ГОСТ и ЕСКД и должен содержать следующие пункты:

а) титульный лист с названием работы, дисциплины, кода группы, ф.и.о. исполнителя и преподавателя;

б) цель и содержание работы;

в) краткое описание аппаратной части и программного обеспечения, необходимого для выполнения работы;

г) скрины: файла Verilog или его непосредственный текст, схемы проекта, фрагмента окна программы Quartus при успешном завершении компиляции, окна назначения выводов, результатов выполнения в виде графических результатов моделирования из окна Simulation Waveform отчета моделирующей программы;

д) выводы по результатам работы.

### **Контрольные вопросы**

1. Что представляет собой ПЛИС? Каковы основные параметры ПЛИС EP4CE10E22C8?

2. Каковы преимущества ПЛИС?

3. Что такое файл конфигурации ПЛИС?
4. Каким образом производится конфигурирование ПЛИС?
5. Назначение системы автоматизированного проектирования Altera Quartus II?
6. Этапы создания проекта в системе Altera Quartus II.
7. Работа с графическим редактором Altera Quartus II.
8. Как производится подключение электрической схемы внутри ПЛИС к внешним выводам?
9. Основные блоки отладочной платы ПЛИС Cyclone IV? Каково их функциональное назначение?

**Список литературы, рекомендованный к использованию по данной теме**

**Основная учебная литература**

**Дополнительная литература**

# ПРОГРАММИРОВАНИЕ ЭЛЕКТРОННЫХ УЗЛОВ И УСТРОЙСТВ

## **Методические указания к выполнению практических работ по направлению подготовки 11.03.04 Электроника и нанoeлектроника**

Квалификация выпускника

**Бакалавр**

Составитель: Лапин А.А.

Редактор:

---

|                                                      |                    |                      |                  |
|------------------------------------------------------|--------------------|----------------------|------------------|
| Подписано в печать _____                             |                    |                      |                  |
| Формат _____                                         | Усл. п. л. – _____ | Уч.-изд. л. – _____. |                  |
| Бумага _____.                                        | Печать _____.      | Заказ _____          | Тираж _____ экз. |
| ФГАОУ ВО «Северо-Кавказский федеральный университет» |                    |                      |                  |
| 355029, г. Ставрополь, пр. Кулакова, 2               |                    |                      |                  |

---

Издательство Северо-Кавказского федерального университета  
Отпечатано в типографии СКФУ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ**

для по организации и проведению самостоятельной работы  
по дисциплине «**Встраиваемые программируемые системы**»  
для студентов направления подготовки

**11.03.04 Электроника и наноэлектроника**

Направленность (профиль):

**Интеллектуальные программируемые системы и устройства**

Ставрополь  
2026

## **Содержание**

Введение

Самостоятельная работа как важнейшая форма учебного процесса

Цели и основные задачи СРС

Виды самостоятельной работы

Организация СРС

Общие рекомендации по организации самостоятельной работы

Самостоятельная работа студента - необходимое звено становления исследователя

Методические рекомендации для студентов по отдельным формам самостоятельной работы

Учебно-методическое и информационное обеспечение дисциплины

## ВВЕДЕНИЕ

### 1. Цель и задачи освоения дисциплины (модуля)

*Цель освоения дисциплины:* формирование набора профессиональных компетенций будущего бакалавра по направлению подготовки 11.03.04 Электроника и нанoeлектроника.

*Задачи освоения дисциплины:*

- сформировать знания, умения и навыки программирования электронных узлов и устройств электронной техники.

### 2. Место дисциплины (модуля) в структуре образовательной программы

Дисциплина Б1.В.20 «Встраиваемые программируемые системы» относится к вариативной части (В) блока дисциплин (Б1). Её освоение происходит в 8 семестре.

### 3. Перечень планируемых результатов обучения по дисциплине (модулю), соотнесённых с планируемыми результатами освоения образовательной программы

| Код, формулировка компетенции                                                                                                                                                                                                                                                                   | Код, формулировка индикатора                                                                                                                         | Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов             |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| ПК-2.<br>Способен производить расчет, моделирование, тестирование, программирование интеллектуальных систем и устройств с применением цифровых и микроконтроллерных устройств в соответствии с техническим заданием с использованием специализированных сред интегрированной разработки (ПК-2). | ИД-1 <sub>ПК-2</sub> Выполняет расчет электронных приборов, схем и устройств в области электроники и нанoeлектроники                                 | Выполняет расчет электронных приборов, схем и устройств в области электроники и нанoeлектроники                                 |
|                                                                                                                                                                                                                                                                                                 | ИД-2 <sub>ПК-2</sub> Использует средства автоматизации проектирования приборов, схем и устройств различного функционального назначения               | Использует средства автоматизации проектирования приборов, схем и устройств различного функционального назначения               |
|                                                                                                                                                                                                                                                                                                 | ИД-3 <sub>ПК-2</sub> Проектирует электронные приборы, схемы и устройства различного функционального назначения в соответствии с техническим заданием | Проектирует электронные приборы, схемы и устройства различного функционального назначения в соответствии с техническим заданием |
| ПК-4.<br>Способен проектировать, моделировать и проводить отладку схемотехнических и конструкторских проектов интеллектуальных электронных средств и систем.                                                                                                                                    | ИД-1 <sub>ПК-4</sub> Применяет правила и нормы монтажа и испытаний сложного электронного оборудования                                                | ИД-1 <sub>ПК-4</sub> Применяет правила и нормы монтажа и испытаний сложного электронного оборудования                           |
|                                                                                                                                                                                                                                                                                                 | ИД-2 <sub>ПК-4</sub> Разрабатывает локальную нормативную документацию для обслуживания приборов электроники                                          | ИД-2 <sub>ПК-4</sub> Разрабатывает локальную нормативную документацию для обслуживания приборов электроники                     |
|                                                                                                                                                                                                                                                                                                 | ИД-3 <sub>ПК-4</sub> Демонстрирует способность сдачи в эксплуатацию приборов и систем электроники                                                    | ИД-3 <sub>ПК-4</sub> Демонстрирует способность сдачи в эксплуатацию приборов и систем электроники                               |
| ПК-6.<br>Способен к сервисному обслуживанию измерительного, диагностического и технологического оборудования, к управлению программно-аппаратными средствами информационных служб                                                                                                               | ИД-1 <sub>ПК-6</sub> Выявляет и диагностирует неполадки измерительного, диагностического, технологического оборудования                              | Выявляет и диагностирует неполадки измерительного, диагностического, технологического оборудования                              |
|                                                                                                                                                                                                                                                                                                 | ИД-2 <sub>ПК-6</sub> Осуществляет частичный ремонт измерительного, диагностического, технологического оборудования                                   | Осуществляет частичный ремонт измерительного, диагностического, технологического оборудования                                   |
|                                                                                                                                                                                                                                                                                                 | ИД-3 <sub>ПК-6</sub> Осуществляет мониторинг диагностического, технологического оборудования                                                         | Осуществляет мониторинг диагностического, технологического оборудования                                                         |

|                                           |                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                         |
|-------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| инфокоммуникационной системы предприятия. | ИД-4 <sub>ПК-6</sub> Понимает общие принципы установки, функционирования и настройки аппаратных, программных и программно-аппаратных средств телекоммуникационной сети предприятия                                                                                           | Понимает общие принципы установки, функционирования и настройки аппаратных, программных и программно-аппаратных средств телекоммуникационной сети предприятия                                                                                           |
|                                           | ИД-5 <sub>ПК-6</sub> Применяет специальные программно-аппаратные средства контроля доступа пользователей к программно-аппаратным средствам информационных служб инфокоммуникационной системы предприятия, осуществляет конфигурирование периферийных и абонентских устройств | Применяет специальные программно-аппаратные средства контроля доступа пользователей к программно-аппаратным средствам информационных служб инфокоммуникационной системы предприятия, осуществляет конфигурирование периферийных и абонентских устройств |

### **САМОСТОЯТЕЛЬНАЯ РАБОТА КАК ВАЖНЕЙШАЯ ФОРМА УЧЕБНОГО ПРОЦЕССА**

Самостоятельная работа - планируемая учебная, учебно-исследовательская, научно-исследовательская работа студентов, выполняемая во внеаудиторное (аудиторное) время по заданию и при методическом руководстве преподавателя, но без его непосредственного участия (при частичном непосредственном участии преподавателя, оставляющем ведущую роль за работой студентов).

Самостоятельная работа студентов в ВУЗе является важным видом учебной и научной деятельности студента. Самостоятельная работа студентов играет значительную роль в рейтинговой технологии обучения. Обучение в ВУЗе включает в себя две, практически одинаковые по объему и взаимовлиянию части – процесса обучения и процесса самообучения. Поэтому СРС должна стать эффективной и целенаправленной работой студента.

Концепцией модернизации российского образования определены основные задачи профессионального образования - "подготовка квалифицированного работника соответствующего уровня и профиля, конкурентоспособного на рынке труда, компетентного, ответственного, свободно владеющего своей профессией и ориентированного в смежных областях деятельности, способного к эффективной работе по специальности на уровне мировых стандартов, готового к постоянному профессиональному росту, социальной и профессиональной мобильности".

Решение этих задач невозможно без повышения роли самостоятельной работы студентов над учебным материалом, усиления ответственности преподавателей за развитие навыков самостоятельной работы, за стимулирование профессионального роста студентов, воспитание творческой активности и инициативы.

К современному специалисту общество предъявляет достаточно широкий перечень требований, среди которых немаловажное значение имеет наличие у выпускников определенных способностей и умения самостоятельно добывать знания из различных источников, систематизировать полученную информацию, давать оценку конкретной финансовой ситуации. Формирование такого умения происходит в течение всего периода обучения через участие студентов в лабораторных занятиях, выполнение контрольных заданий и тестов, написание курсовых и выпускных квалификационных работ. При этом самостоятельная работа студентов играет решающую роль в ходе всего учебного процесса.

Формы самостоятельной работы студентов разнообразны. По данной дисциплине «Основы нейронных сетей» предусмотрены следующие:

### **Использование материала учебно-методического комплекса дисциплины**

На первом этапе необходимо ознакомиться с обязательным минимумом содержания основной образовательной программы по изучаемой дисциплине. Далее необходимо изучить рабочую программу дисциплины, в которой рассмотрено содержание тем дисциплины лекционного курса, взаимосвязь тем лекций с лабораторных занятий, темы и виды самостоятельной работы. По каждому виду самостоятельной работы предусмотрены определённые формы отчетности, представленные в учебной программе по дисциплине «Основы нейронных сетей».

### **Работа с литературой**

Для успешного освоения дисциплины, необходимо самостоятельно детально изучить представленные темы по рекомендуемым источникам информации, которые представлены в учебной программе.

Самостоятельная работа приобщает студентов к научному творчеству, поиску и решению актуальных современных проблем.

### **ЦЕЛИ И ОСНОВНЫЕ ЗАДАЧИ СРС**

Ведущая цель организации и осуществления СРС должна совпадать с целью обучения студента – подготовкой бакалавра с высшим образованием. При организации СРС важным и необходимым условием становятся формирование умения самостоятельной работы для приобретения знаний, навыков и возможности организации учебной и научной деятельности.

Целью самостоятельной работы студентов является овладение фундаментальными знаниями, профессиональными умениями и навыками деятельности по профилю, опытом творческой, исследовательской деятельности. Самостоятельная работа студентов способствует развитию самостоятельности, ответственности и организованности, творческого подхода к решению проблем учебного и профессионального уровня.

Задачами СРС являются:

- систематизация и закрепление полученных теоретических знаний и практических умений студентов;
- углубление и расширение теоретических знаний;
- формирование умений использовать нормативную, правовую, справочную документацию и специальную литературу;
- развитие познавательных способностей и активности студентов: творческой инициативы, самостоятельности, ответственности и организованности;
- формирование самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;
- развитие исследовательских умений;
- использование материала, собранного и полученного в ходе самостоятельных занятий на семинарах, на лабораторных занятиях, при написании курсовых и выпускной квалификационной работ, для эффективной подготовки к итоговым зачетам и экзаменам.

### **ВИДЫ САМОСТОЯТЕЛЬНОЙ РАБОТЫ**

В образовательном процессе высшего профессионального образовательного учреждения выделяется два вида самостоятельной работы – аудиторная, под руководством преподавателя, и внеаудиторная. Тесная взаимосвязь этих видов работ предусматривает дифференциацию и эффективность результатов ее выполнения и зависит от организации, содержания, логики учебного процесса (межпредметных связей, перспективных знаний и др.):

Аудиторная самостоятельная работа по дисциплине выполняется на учебных занятиях под непосредственным руководством преподавателя и по его заданию.

Внеаудиторная самостоятельная работа выполняется студентом по заданию преподавателя, но без его непосредственного участия.

Основными видами самостоятельной работы студентов без участия преподавателя по данной дисциплине являются:

- формирование и усвоение содержания конспекта лекций на базе рекомендованной лектором учебной литературы, включая информационные образовательные ресурсы (электронные учебники, электронные библиотеки и др.);
- подготовка к лабораторным работам, их оформление.

Основными видами самостоятельной работы студентов с участием преподавателей являются:

- текущие консультации;
- прием и защита лабораторных работ (во время проведения л/р).

### **ОРГАНИЗАЦИЯ СРС**

Методика организации самостоятельной работы студентов зависит от структуры, характера и особенностей дисциплины «Основы нейронных сетей», объема часов на ее изучение, вида заданий для самостоятельной работы студентов, индивидуальных качеств студентов и условий учебной деятельности.

Процесс организации самостоятельной работы студентов включает в себя следующие этапы:

- подготовительный (определение целей, составление программы, подготовка методического обеспечения, подготовка оборудования);
- основной (реализация программы, использование приемов поиска информации, усвоения, переработки, применения, передачи знаний, фиксирование результатов, самоорганизация процесса работы);
- заключительный (оценка значимости и анализ результатов, их систематизация, оценка эффективности программы и приемов работы, выводы о направлениях оптимизации труда).

### **Деятельность студентов по формированию и развитию навыков учебной самостоятельной работы**

В процессе самостоятельной работы студент приобретает навыки самоорганизации, самоконтроля, самоуправления, саморефлексии и становится активным самостоятельным субъектом учебной деятельности.

*Выполняя самостоятельную работу под контролем преподавателя студент должен:*

– освоить минимум содержания, выносимый на самостоятельную работу студентов и предложенный преподавателем в соответствии с образовательными стандартами высшего образования;

– планировать самостоятельную работу в соответствии с графиком самостоятельной работы, предложенным преподавателем;

– самостоятельную работу студент должен осуществлять в организационных формах, предусмотренных учебным планом и рабочей программой преподавателя;

– выполнять самостоятельную работу и отчитываться по ее результатам в соответствии с графиком представления результатов, видами и сроками отчетности по самостоятельной работе студентов.

*студент может:*

сверх предложенного преподавателем (при обосновании и согласовании с ним) и минимума обязательного содержания, определяемого программой по данной дисциплине:

– самостоятельно определять уровень (глубину) проработки содержания материала;

– предлагать темы и вопросы для самостоятельной проработки;

– в рамках общего графика выполнения самостоятельной работы предлагать обоснованный индивидуальный график выполнения и отчетности по результатам самостоятельной работы;

– предлагать свои варианты организационных форм самостоятельной работы;

– использовать для самостоятельной работы методические пособия, учебные пособия, разработки сверх предложенного преподавателем перечня;

– использовать не только контроль, но и самоконтроль результатов самостоятельной работы в соответствии с методами самоконтроля, предложенными преподавателем или выбранными самостоятельно.

Самостоятельная работа студентов должна оказывать важное влияние на формирование личности будущего бакалавра, она планируется студентом самостоятельно. Каждый студент самостоятельно определяет режим своей работы и меру труда, затрачиваемого на овладение учебным содержанием по каждой дисциплине. Он выполняет внеаудиторную работу по личному индивидуальному плану, в зависимости от его подготовки, времени и других условий.

## **ОБЩИЕ РЕКОМЕНДАЦИИ ПО ОРГАНИЗАЦИИ САМОСТОЯТЕЛЬНОЙ РАБОТЫ**

Основной формой самостоятельной работы студента является изучение конспекта лекций, их дополнение, рекомендованной литературы, активное участие на лабораторных занятиях. Но для успешной учебной деятельности, ее интенсификации, необходимо учитывать следующие субъективные факторы:

1. Знание школьного программного материала, наличие прочной системы знаний, необходимой для усвоения основных вузовских курсов. Это особенно важно для математических дисциплин. Необходимо отличать пробелы в знаниях, затрудняющие усвоение нового материала, от малых способностей. Затратив силы на преодоление этих пробелов, студент обеспечит себе нормальную успеваемость и поверит в свои способности.

2. Наличие умений, навыков умственного труда:

а) умение конспектировать на лекции и при работе с книгой;

б) владение логическими операциями: сравнение, анализ, синтез, обобщение, определение понятий, правила систематизации и классификации.

3. Специфика познавательных психических процессов: внимание, память, речь, наблюдательность, интеллект и мышление. Слабое развитие каждого из них становится серьезным препятствием в учебе.

4. Хорошая работоспособность, которая обеспечивается нормальным физическим состоянием. Ведь серьезное учение — это большой многосторонний и разнообразный труд. Результат обучения оценивается не количеством сообщаемой информации, а качеством ее усвоения, умением ее использовать и развитием у себя способности к дальнейшему самостоятельному образованию.

5. Соответствие избранной деятельности, профессии индивидуальным способностям. Необходимо выработать у себя умение саморегулировать свое эмоциональное состояние и устранять обстоятельства, нарушающие деловой настрой, мешающие намеченной работе.

6. Овладение оптимальным стилем работы, обеспечивающим успех в деятельности. Чередование труда и пауз в работе, периоды отдыха, индивидуально обоснованная норма продолжительности сна, предпочтение вечерних или утренних занятий, стрессоустойчивость на экзаменах и особенности подготовки к ним,

7. Уровень требований к себе, определяемый сложившейся самооценкой.

Адекватная оценка знаний, достоинств, недостатков - важная составляющая самоорганизации человека, без нее невозможна успешная работа по управлению своим поведением, деятельностью.

Одна из основных особенностей обучения в высшей школе заключается в том, что постоянный внешний контроль заменяется самоконтролем, активная роль в обучении принадлежит уже не столько преподавателю, сколько студенту.

Зная основные методы научной организации умственного труда, можно при наименьших затратах времени, средств и трудовых усилий достичь наилучших результатов.

Эффективность усвоения поступающей информации зависит от работоспособности человека в тот или иной момент его деятельности.

*Работоспособность* - способность человека к труду с высокой степенью напряженности в течение определенного времени. Различают внутренние и внешние факторы работоспособности.

К внутренним факторам работоспособности относятся интеллектуальные особенности, воля, состояние здоровья.

К внешним:

- организация рабочего места, режим труда и отдыха;
- уровень организации труда - умение получить справку и пользоваться информацией;
- величина умственной нагрузки.

Выдающийся русский физиолог Н. Е. Введенский выделил следующие условия продуктивности умственной деятельности:

- во всякий труд нужно входить постепенно;
- мерность и ритм работы. Разным людям присущ более или менее разный темп работы;
- привычная последовательность и систематичность деятельности;

- правильное чередование труда и отдыха.

Отдых не предполагает обязательного полного бездействия со стороны человека, он может быть достигнут простой переменой дела. В течение дня работоспособность изменяется. Наиболее плодотворным является *утреннее время (с 8 до 14 часов)*, причем максимальная работоспособность приходится на период с 10 до 13 часов, затем *послеобеденное* - (с 16 до 19 часов) и *вечернее* (с 20 до 24 часов). Очень трудный для понимания материал лучше изучать в начале каждого отрезка времени (лучше всего утреннего) после хорошего отдыха. Через 1-1,5 часа нужны перерывы по 10 - 15 мин, через 3 - 4 часа работы отдых должен быть продолжительным - около часа.

Составной частью научной организации умственного труда является овладение техникой умственного труда.

Время, которым располагает студент для выполнения учебного плана, складывается из двух составляющих: одна из них — это аудиторная работа в вузе по расписанию занятий, другая - внеаудиторная самостоятельная работа. Задания и материалы для самостоятельной работы выдаются во время учебных занятий по расписанию, на этих же занятиях преподаватель осуществляет контроль за самостоятельной работой, а также оказывает помощь студентам по правильной организации работы.

Самостоятельные занятия требуют интенсивного умственного труда, который необходимо не только правильно организовать, но и стимулировать. При этом очень важно уметь поддерживать устойчивое внимание к изучаемому материалу. Выработка внимания требует значительных волевых усилий. Именно поэтому, если студент замечает, что он часто отвлекается во время самостоятельных занятий, ему надо заставить себя сосредоточиться. Подобную процедуру необходимо проделывать постоянно, так как это является тренировкой внимания. Устойчивое внимание появляется тогда, когда человек относится к делу с интересом.

Следует правильно организовать свои занятия по времени: 50 минут - работа, 5-10 минут - перерыв; после 3 часов работы перерыв - 20-25 минут. Иначе нарастающее утомление повлечет неустойчивость внимания. Очень существенным фактором, влияющим на повышение умственной работоспособности, являются систематические занятия физической культурой. Организация активного отдыха предусматривает чередование умственной и физической деятельности, что полностью восстанавливает работоспособность человека.

### **САМОСТОЯТЕЛЬНАЯ РАБОТА СТУДЕНТА - НЕОБХОДИМОЕ ЗВЕНО СТАНОВЛЕНИЯ ИССЛЕДОВАТЕЛЯ**

Прогресс науки и техники, информационных технологий приводит к значительному увеличению научной информации, что предъявляет более высокие требования не только к моральным, нравственным свойствам человека, но и в особенности, постоянно возрастающие требования в области образования – обновление, модернизация общих и профессиональных знаний, умений специалиста.

Всякое образование должно выступать как динамический процесс, присущий человеку и продолжающийся всю его жизнь. Овладение научной мыслью и языком науки является необходимой составляющей в самоорганизации будущего специалиста исследователя. Под этим понимается не столько накопление знаний, сколько овладение

научно обоснованными способами их приобретения. В этом, вообще говоря, состоит основная задача вуза.

Специфика вузовского учебного процесса, в организации которого самостоятельной работе студента отводятся все больше места, состоит в том, что он является как будто бы последним и самым адекватным звеном для реализации этой задачи. Ибо во время учебы в вузе происходит выработка стиля, навыков учебной (познавательной) деятельности, рациональный характер которых будет способствовать постоянному обновлению знаний высококвалифицированного выпускника вуза.

Однако до этого пути существуют определенные трудности, в частности, переход студента от синтетического процесса обучения в средней школе, к аналитическому в высшей. Это связано как с новым содержанием обучения (расширение общего образования и углубление профессиональной подготовки), так и с новыми, неизвестными до сих пор формами: обучения (лекции, семинары, лабораторные занятия и т.д.). Студент получает не только знания, предусмотренные программой и учебными пособиями, но он также должен познакомиться со способами приобретения знаний так, чтобы суметь оценить, что мы знаем, откуда мы это знаем и как этого знания мы достигли. Ко всему этому приходят через собственную самостоятельную работу.

Это и потому, что самостоятельно приобретенные знания являются более оперативными, они становятся личной собственностью, а также мотивом поведения, развивают интеллектуальные черты, внимание, наблюдательность, критичность, умение оценивать. Роль преподавателя в основном заключается в руководстве накопления знаний (по отношению к первокурсникам), а в последующие годы учебы, на старших курсах, в совместном установлении проблем и заботе о самостоятельных поисках студента, а также контролирования за их деятельностью. Отметим, что нельзя ограничиваться только приобретением знаний, предусмотренных программой изучаемой дисциплины, надо постоянно углублять полученные знания, сосредотачивая их на какой-нибудь узкой определенной области, соответствующей интересам студента. Углубленное изучение всех предметов, предусмотренных программой, на практике является возможным, и хорошая организация работы позволяет экономить время, что создает условия для глубокого, систематического, заинтересованного изучения самостоятельно выбранной студентом темы.

Конечно, все советы, примеры, рекомендации в этой области, даваемые преподавателем, или определенными публикациями, или другими источниками, не гарантируют никакого успеха без проявления собственной активности в этом деле, т. е. они не дают готовых рецептов, а должны способствовать анализу собственной работы, ее целей, организации в соответствии с индивидуальными особенностями. Учитывая личные возможности, существующие условия жизни и работы, навыки, на основе этих рекомендаций, возможно, выработать индивидуально обоснованную совокупность методов, способов, найти свой стиль или усовершенствовать его, чтобы изучив определенный материал, иметь время оценить его значимость, пригодность и возможности его применения, чтобы, в конечном счете, обеспечить успешность своей учебы с будущей профессиональной деятельности

## **МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ СТУДЕНТОВ ПО ОТДЕЛЬНЫМ ФОРМАМ САМОСТОЯТЕЛЬНОЙ РАБОТЫ**

Система вузовского обучения подразумевает значительно большую самостоятельность студентов в планировании и организации своей деятельности.

### **Работа с литературой**

При работе с литературой необходимо подобрать литературу, научиться правильно ее читать, вести записи. Для подбора литературы в библиотеке используются алфавитный и систематический каталоги.

Важно помнить, что рациональные навыки работы с книгой — это всегда большая экономия времени и сил.

Правильный подбор учебников рекомендуется преподавателем, читающим лекционный курс. Необходимая литература может быть также указана в методических разработках по данному курсу.

Изучая материал по учебнику, следует переходить к следующему вопросу только после правильного уяснения предыдущего, описывая на бумаге все выкладки и вычисления (в том числе те, которые в учебнике опущены или на лекции даны для самостоятельного вывода).

При изучении любой дисциплины большую и важную роль играет самостоятельная индивидуальная работа.

Особое внимание следует обратить на определение основных понятий курса. Студент должен подробно разбирать примеры, которые поясняют такие определения, и уметь строить аналогичные примеры самостоятельно. Нужно добиваться точного представления о том, что изучаешь. Полезно составлять опорные конспекты. При изучении материала по учебнику полезно в тетради (на специально отведенных полях) дополнять конспект лекций. Там же следует отмечать вопросы, выделенные студентом для консультации с преподавателем.

Выводы, полученные в результате изучения, рекомендуется в конспекте выделять, чтобы они при перечитывании записей лучше запоминались.

Опыт показывает, что многим студентам помогает составление листа опорных сигналов, содержащего важнейшие и наиболее часто употребляемые формулы и понятия. Такой лист помогает запомнить формулы, основные положения лекции, а также может служить постоянным справочником для студента.

Различают два вида чтения; первичное и вторичное. *Первичное* - это внимательное, неторопливое чтение, при котором можно остановиться на трудных местах. После него не должно остаться ни одного непонятого слова. Содержание не всегда может быть понятно после первичного чтения.

Задача *вторичного* чтения полное усвоение смысла целого (по счету это чтение может быть и не вторым, а третьим или четвертым).

Чтение научного текста является частью познавательной деятельности. Ее цель — извлечение из текста необходимой информации. От того на сколько осознанно читающим собственная внутренняя установка при обращении к печатному слову (найти нужные сведения, усвоить информацию полностью или частично, критически проанализировать материал и т.п.) во многом зависит эффективность осуществляемого действия.

Выделяют **четыре основные установки в чтении научного текста:**

1. информационно-поисковый (задача — найти, выделить искомую информацию)
1. усваивающая (усилия читателя направлены на то, чтобы как можно полнее осознать и запомнить как сами сведения, излагаемые автором, так и всю логику его рассуждений)

2. аналитико-критическая (читатель стремится критически осмыслить материал, проанализировав его, определив свое отношение к нему)

3. творческая (создает у читателя готовность в том или ином виде – как отправной пункт для своих рассуждений, как образ для действия по аналогии и т.п. – использовать суждения автора, ход его мыслей, результат наблюдения, разработанную методику, дополнить их, подвергнуть новой проверке).

С наличием различных установок обращения к научному тексту связано существование и нескольких **видов чтения**:

1. библиографическое – просматривание карточек каталога, рекомендательных списков, сводных списков журналов и статей за год и т.п.;

2. просмотровое – используется для поиска материалов, содержащих нужную информацию, обычно к нему прибегают сразу после работы со списками литературы и каталогами, в результате такого просмотра читатель устанавливает, какие из источников будут использованы в дальнейшей работе;

3. ознакомительное – подразумевает сплошное, достаточно подробное прочтение отобранных статей, глав, отдельных страниц, цель – познакомиться с характером информации, узнать, какие вопросы вынесены автором на рассмотрение, провести сортировку материала;

4. изучающее – предполагает доскональное освоение материала; в ходе такого чтения проявляется доверие читателя к автору, готовность принять изложенную информацию, реализуется установка на предельно полное понимание материала;

5. аналитико-критическое и творческое чтение – два вида чтения близкие между собой тем, что участвуют в решении исследовательских задач. Первый из них предполагает направленный критический анализ, как самой информации, так и способов ее получения и подачи автором; второе – поиск тех суждений, фактов, по которым или в связи, с которыми, читатель считает нужным высказать собственные мысли.

Из всех рассмотренных видов чтения основным для студентов является изучающее – именно оно позволяет в работе с учебной литературой накапливать знания в различных областях. Вот почему именно этот вид чтения в рамках учебной деятельности должен быть освоен в первую очередь. Кроме того, при овладении данным видом чтения формируются основные приемы, повышающие эффективность работы с научным текстом.

### **Основные виды систематизированной записи прочитанного:**

1. Аннотирование – предельно краткое связное описание просмотренной или прочитанной книги (статьи), ее содержания, источников, характера и назначения;

1. Планирование – краткая логическая организация текста, раскрывающая содержание и структуру изучаемого материала;

2. Тезирование – лаконичное воспроизведение основных утверждений автора без привлечения фактического материала;

3. Цитирование – дословное выписывание из текста выдержек, извлечений, наиболее существенно отражающих ту или иную мысль автора;

4. Конспектирование – краткое и последовательное изложение содержания прочитанного.

Конспект – сложный способ изложения содержания книги или статьи в логической последовательности. Конспект аккумулирует в себе предыдущие виды записи, позволяет

всесторонне охватить содержание книги, статьи. Поэтому умение составлять план, тезисы, делать выписки и другие записи определяет и технологию составления конспекта.

### **Методические рекомендации по составлению конспекта:**

1. Внимательно прочитайте текст. Уточните в справочной литературе непонятные слова. При записи не забудьте вынести справочные данные на поля конспекта.
2. Выделите главное, составьте план.
3. Кратко сформулируйте основные положения текста, отметьте аргументацию автора.
4. Законспектируйте материал, четко следуя пунктам плана. При конспектировании старайтесь выразить мысль своими словами. Записи следует вести четко, ясно.
5. Грамотно записывайте цитаты. Цитируя, учитывайте лаконичность, значимость мысли.

В тексте конспекта желательно приводить не только тезисные положения, но и их доказательства. При оформлении конспекта необходимо стремиться к емкости каждого предложения. Мысли автора книги следует излагать кратко, заботясь о стиле и выразительности написанного. Число дополнительных элементов конспекта должно быть логически обоснованным, записи должны распределяться в определенной последовательности, отвечающей логической структуре произведения. Для уточнения и дополнения необходимо оставлять поля.

Овладение навыками конспектирования требует от студента целеустремленности, повседневной самостоятельной работы.

### **Лабораторные занятия**

Следует подчеркнуть, что только после усвоения лекционного материала с определенной точки зрения (а именно с той, с которой он излагается на лекциях) он будет закрепляться на лабораторных занятиях как в результате обсуждения и анализа лекционного материала, так и с помощью выполнения заданий лабораторной работы, поставленных задач. При этих условиях студент не только хорошо усвоит материал, но и научится применять его на практике, а также получит дополнительный стимул (и это очень важно) для активной проработки лекции.

При самостоятельном выполнении заданий нужно обосновывать каждый этап решения, исходя из теоретических положений курса. Если студент видит несколько путей решения проблемы (задачи), то нужно сравнить их и выбрать самый рациональный. Полезно до начала вычислений составить краткий план решения проблемы (задачи). Решение задач или примеров следует излагать подробно, вычисления располагать в строгом порядке, отделяя вспомогательные вычисления от основных. Решения при необходимости нужно сопровождать комментариями, схемами, чертежами и рисунками.

Следует помнить, что выполнение каждого учебного задания должно доводиться до окончательного логического ответа, которого требует условие, и по возможности с выводом. Выполнение заданий данного типа нужно продолжать до приобретения твердых навыков в их решении.

### **Самопроверка**

После изучения определенной темы по записям в конспекте и учебнику, а также решения достаточного количества соответствующих заданий на лабораторных занятиях и самостоятельно студенту рекомендуется, используя лист опорных материалов, воспроизвести по памяти определения, выводы формул, формулировки основных положений и доказательств.

В случае необходимости нужно еще раз внимательно разобраться в материале.

Иногда недостаточность усвоения того или иного вопроса выясняется только при изучении дальнейшего материала. В этом случае надо вернуться назад и повторить плохо усвоенный материал. Важный критерий усвоения теоретического материала - умение решать задачи или пройти тестирование по пройденному материалу. Однако следует помнить, что правильное решение задачи может получиться в результате применения механически заученных формул без понимания сущности теоретических положений.

### **Консультации**

Если в процессе самостоятельной работы над изучением теоретического материала или при выполнении заданий у студента возникают вопросы, разрешить которые самостоятельно не удастся, необходимо обратиться к преподавателю для получения у него разъяснений или указаний. В своих вопросах студент должен четко выразить, в чем он испытывает затруднения, характер этого затруднения. За консультацией следует обращаться и в случае, если возникнут сомнения в правильности ответов на вопросы самопроверки.

### **Подготовка к экзамену по дисциплине**

Изучение многих общепрофессиональных и специальных дисциплин завершается экзаменом. Подготовка к экзамену способствует закреплению, углублению и обобщению знаний, получаемых, в процессе обучения, а также применению их к решению практических задач. Готовясь к экзамену, студент ликвидирует имеющиеся пробелы в знаниях, углубляет, систематизирует и упорядочивает свои знания.

Экзамен - форма промежуточной аттестации студентов по результатам освоения теоретических знаний, приобретения практических навыков и компетенций, целью которой является контроль результатов освоения студентами учебного материала по программе конкретной дисциплины, проверка и оценка знаний, полученных за семестр (курс), их прочности, развития творческого мышления, приобретения навыков самостоятельной работы, умения применять теоретические знания при решении практических задач. Экзамен может быть проведен в устной форме, в устно-письменной форме, в форме письменной работы или тестирования.

Число вопросов, включаемых в экзаменационный билет, должно быть не менее двух и не более пяти, при этом вопросы могут носить как теоретический, так и прикладной характер. На экзамен могут выноситься типовые задачи, проработанные в течение семестра на аудиторных занятиях и в процессе самостоятельной работы. Содержание вопросов и задач, включаемых в экзаменационный билет, должно соответствовать учебной программе дисциплины и фондам оценочных средств. Перечень примерных вопросов, заданий и критерии оценки уровня обученности и уровня сформированности компетенций доводятся до сведения обучающихся в начале семестра изучения дисциплины.

Систематическая подготовка к занятиям в течение семестра позволит использовать время экзаменационной сессии для систематизации знаний.

Текущая аттестация студентов проводится преподавателями, ведущими лабораторные занятия по дисциплине в следующих формах: собеседование при защите лабораторной работы с предоставлением письменного отчета. Допуск к лабораторным

работам происходит при наличии у студентов печатного варианта отчета. Защита отчета проходит в форме собеседования. Максимальное количество баллов студент получает, если оформление отчета соответствует установленным требованиям, а отчет полностью раскрывает суть работы. Основанием для снижения оценки являются:

- не качественно выполненная работа на персональном компьютере,
- письменный отчет выполнен в небрежной форме и с ошибками,
- студент не дает исчерпывающий ответ на вопрос преподавателя,
- путается или затрудняется в демонстрации работы на компьютере.

Отчет может быть отправлен на доработку в следующих случаях:

- небрежное выполнение и с ошибками,
- сокращенное содержание отчета.

Критерии оценивания собеседования приведены в Фонде оценочных средств по дисциплине Основы нейронных сетей.

## УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

8.1. Перечень основной и дополнительной литературы, необходимой для освоения дисциплины (модуля)

8.1.1. Перечень основной литературы:

Огородников, И.Н. Микропроцессорная техника. Введение в Cortex-M3 Электронный ресурс : учебное пособие / И.Н. Огородников. - Микропроцессорная техника. Введение в Cortex-M3, 2022-08-31. - Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2015. - 116 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-7996-1499-7

8.1.2. Перечень дополнительной литературы:

Бибило, П. Н. Системы проектирования интегральных схем на основе языка VHDL. StateCAD, ModelSim, LeonardoSpectrum / П.Н. Бибило. - М. : СОЛОН-Пресс, 2005. - 384 с. : ил. - (Системы проектирования). - Библиогр.: с. 382-384. - ISBN 5-98003-197-9

Бибило, П.Н. Основы языка VHDL Электронный ресурс : учебное пособие / П.Н. Бибило. - Основы языка VHDL, 2019-05-25. - Москва : СОЛОН-ПРЕСС, 2010. - 200 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 5-93455-056-X

Майо, Дж. Самоучитель Microsoft Visual Studio 2010 : [практ. пособие] / Дж. Майо. - СПб. : БХВ-Петербург, 2011. - 464 с. : ил. - Прил.: с. 419. - ISBN 978-5-9775-0609-0

Муровмцев, Д. Ю. Микропроцессоры и микроЭВМ / Д.Ю. Муровмцев ; Е.Н. Яшин. - Тамбов : Издательство ФГБОУ ВПО «ТГТУ», 2013. - 97 с. - ISBN 978-5-8265-1172-5

Поляков, А. К. Языки VHDL и VERILOG в проектировании цифровой аппаратуры : Учебное пособие / Поляков А. К. - Москва : СОЛОН-ПРЕСС, 2009. - 314 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 5-98003-016-6

Практическое руководство по программированию STM-микроконтроллеров / С.Н. Торгаев ; М.В. Тригуб ; И.С. Мусоров ; Д.С. Чертихина. - Томск : Издательство Томского политехнического университета, 2015. - 111 с.

Практическое руководство по программированию STM-микроконтроллеров Электронный ресурс : учебное пособие / Д.С. Чертихина / И.С. Мусоров / М.В. Тригуб / С.Н. Торгаев. - Томск : Томский политехнический университет, 2015. - 111 с. - Книга находится в базовой версии ЭБС IPRbooks.

Ревич, Ю. В. Практическое программирование микроконтроллеров Atmel AVR на языке ассемблера / Юрий Ревич. - Санкт-Петербург : БХВ-Петербург, 2008. - 384 с. : ил., табл. - (Аппаратные средства). - Библиогр.: с. 381-382. - Предм. указ.: с. 383-384. - ISBN 978-5-9775-0277-1

Секаев, В. Г. Основы программирования на Ассемблере : Учебное пособие / Секаев В. Г. - Новосибирск : Новосибирский государственный технический университет, 2010. - 100 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-7782-1473-6

Царев, Р. Ю. Программирование на языке Си : учебное пособие / Р.Ю. Царев ; Министерство образования и науки Российской Федерации ; Сибирский Федеральный университет. - Красноярск : Сибирский федеральный университет, 2014. - 108 с. : табл., схем. - <http://biblioclub.ru/>. - Библиогр. в кн. - ISBN 978-5-7638-3006-4

Шарапов, А. В. Основы микропроцессорной техники : Учебное пособие / Шарапов А. В. - Томск : Томский государственный университет систем управления и радиоэлектроники, 2008. - 240 с. - Книга находится в базовой версии ЭБС IPRbooks.

8.2. Перечень учебно-методического обеспечения самостоятельной работы обучающихся по дисциплине (модулю)

Лاپин, А.А. Программирование электронных узлов и устройств: методические указания к выполнению практических работ. – Ставрополь: Изд-во СКФУ, 2023. – 210 с.

Лاپин, А.А. Программирование электронных узлов и устройств: методические указания к самостоятельной работе студентов. – Ставрополь: Изд-во СКФУ, 2023. – 11 с.

8.3. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины (модуля)

<http://arduino.ru/projects> – Сайт «Материалы по программированию Arduino»

<http://arduino-project.net/videoouroki-po-arduino-ot-jeremy-blum/> – Сайт Arduino-проекты

<http://www.atmel.com/ru/ru/tools/atmelstudio.aspx?tab=overview> – Сайт компании Atmel

<http://www.gaw.ru/avr.htm> – Сайт проекта «РЫНОК МИКРОЭЛЕКТРОНИКИ»

<http://www.gaw.ru/html.cgi/txt/doc/micros/arm/index.htm> – Сайт проекта «РЫНОК МИКРОЭЛЕКТРОНИКИ»

<http://www.gaw.ru/html.cgi/txt/ic/Altera/index.htm> – Сайт проекта «РЫНОК МИКРОЭЛЕКТРОНИКИ»

<https://www.arduino.cc/> – Официальный сайт проекта Ардуино