**ЗАДАЧА 1**

**Пример 1**

*Расчет токов в ветвях электрической цепи по правилам Кирхгофа*

Задание:

Схема

 

|  |
| --- |
| Параметры цепи |
| R1 | R2 | R3 | R4 | R5 | R6 | E1 | E2 |
| Ом | В |
| 10 | 18 | 5 | 10 | 8 | 10 | 20 | 35 |

Первый закон Кирхгофа: 

Второй закон Кирхгофа: 

 

**Пример 2**

*Расчет токов в ветвях электрической цепи с использованием законов Кирхгофа с помощью MathCad*

**Задание:** Определите токи в каждой ветви цепи постоянного тока, (см. рис. ниже), если указаны следующие величины:

$R\_{1}=0,3 ОМ, $ $R\_{2}=1 ОМ,$ $R\_{3}=24 ОМ,$ ε1 $=246 В,$ ε2 $=230 B$



**Решение.**

Направления токов выбраны условно, если после решения значения каких-то токов получатся отрицательными, значит реальное направление для данного тока противоположно выбранному.

Т.к. ветвей вцепи три, составим три уравнения. Первое – по первому закону Киргофа – для узла А, и два – по второму закону Киргофа - для контура, по которому протекают токи I1 и I3, и для контура с токами I2 и I3. В обоих случаях направление обхода контура выберем по часовой стрелке.

Тогда получим следующую систему уравнений:



Подставляя известные числовые параметры (сопротивления и ЭДС), получим такую систему трёх уравнения с тремя неизвестными токами:



Решим эту систему методом Крамера. Этот метод решения систем линейных алгебраических уравнений подходит для систем с ненулевым определителем основной матрицы. Эта матрица строится по коэффициентам левых частей выровненных (т.е. с учётом, что некоторые коэффициенты в некоторых уравнениях могут быть равными нулю) уравнений, и имеет вид:









Значит, существует единственное решение.

Ответ:

I1 = 20;

I2 = 10 (направление тока противоположно условно выбранному);

I3 = 10.

**Пример 3**

**Схема в программе Multisim**

Исходная схема задания



«Собранная» в Multisim цепь с измерительными приборами имеет вид:

****

Примечание: для избавления от ненужных надписей (внутр. сопротивление приборов) амперметры и вольтметры настроены следующим образом:



**ЗАДАЧА 2**

***Моделирование работы семисегментного индикатора с помощью формирователя двоичных сигналов универсального типа Word Generator***

Соберите схему, показанную на рис. 3. Схема содержит уже запрограммированный генератор слова Word Generator (XWG1), семисегментный индикатор 16-теричных цифр DCD\_Hex и логические пробники X1, … X4. Генератор слова XWG находится на панели инструментов Instruments, откуда снимается щелчком ЛКМ и позиционируется в рабочем окне также щелчком мыши. Генератор слов (лицевая панель генератора открывается двойным щелчком ЛКМ) предназначен для генерации 1024-х 32-разрядных двоичных слов, которые набираются пользователем на экране, расположенном в правой части лицевой панели.

Каждый разряд формируется отдельным выходом или каналом, пронумерованными цифрами от 0 до 31. Выход любого канала является независимым от других каналов.

Рис 3. Схема моделирования работы "генератора слов"

Компонент **DCD\_Hex** представляет собой индикатор с 7 сегментами. Это позволяет подать на его 4 входа двоичный код от 0000 до 1111 *(младшая 16-теричная цифра, формируемая каналами 0 – 3 генератора слова)* и на буквенно-цифровом дисплее получить 16-теричный эквивалент от 0 до F. Система преобразования двоичного кода 8-4-2-1 в семисегментный встроена в компонент. Для наглядности, работа семисегментного индикатора контролируется логическими пробниками.

Чтобы начать процесс моделирования работы **Word Generator,** его надо запрограммировать на выдачу указанной последовательности двоичных комбинаций.

Существует *два способа набора кодовых комбинаций*.

1. Ручной режим. Выделяется нужная кодовая комбинация, в которой с помощью клавиатуры можно набрать произвольное 8-разрядное 16-теричное число (кнопка hex в окне Display) или 32-разрядное двоичное (кнопка Binary). Использование десятичных -и ASCII- чисел для нас не будут представлять практического интереса.

2. Автоматический режим. Например, создадим нужную нам возрастающую двоичную последовательность для демонстрации работы схемы. Нажмём на кнопку **Set.** Откроется ДО **Settings** (рис.4)**,** в котором выберем опцию **Up Counter** и нажмём кнопку **Accept.**



Рис. 4 Панель автоматической установки

кодовых комбинаций Word

Запрограммировав **Word Generator,** надо выбрать режим его работы. Режим **Cycle (Цикл)** обозначает, что генератор будет циклически выводить запрограммированную последовательность, пока не будет остановлено моделирование. В режиме **Burst (пакет)** цикл будет выполнен один раз. В режиме **Step (шаг)** будет выведено содержимое одной ячейки, после чего моделирование будет остановлено (пошаговой режим). Если в процессе программирования генератора, щёлкнуть по ячейки правой кнопкой мыши установит опцию **Set Breakpoint**, то данная ячейка будет настроена как точка прерывания. Word Generator остановится в этой точке в режимах Cycle и Burst. Для удаления точки прерывания выполните опцию **Delete** **Breakpoint.** Во время моделирования курсор ▶ в окне **Word Generator - XWG1** указывает на текущую ячейку.

Процесс моделирования включает следующие шаги:

1. Курсором выделим начальную комбинацию и установим опцию Set Cursor

2. Выбрать кнопку с режимом моделирования (Cycle, Burst, Step) и щёлкнуть по ней мышью.

Реализация п.2 автоматически установит переключатель Simulation Switch (рис. 3) в режим моделирования, пока не окончится генерация слов, предусмотренная выбранным вами режимом моделирования. По окончанию режима моделирования (кроме режима Cycle) переключатель Simulation Switch установится в состояние Pause. Для продолжения моделирования снова щёлкните по соответствующей кнопке в окне Word Generator-XWG1. Остановить циклический режим моделирования можно только переключателем Simulation Switch, установив его в исходное состояние.

**ЗАДАЧА 3**

 **Моделирование схем с микроконтроллерами в программе Proteus.Isis**

**Программирование микроконтроллеров в программе CodeVisionAVR.**

***Ход работы***

**1. Создание схемы в Proteus.Isis.**

Схема для эксперимента приведена на рис.1.

Наш светодиод рассчитан примерно на силу тока примерно в 20 мА. При этом на светодиоде падает примерно 2 В. Остаётся: 5 В (напряжение VCC) – 2 = 3 В.

По закону Ома I = U/R. Тогда R= 150 Ом.

Вычисленное значение R и вводим в качестве параметра Resistance свойств R1.



**2. Создание программного кода в *CodeVisionAVR***

Для того, чтобы микроконтроллер работал, ему нужен программный файл – «прошивка». Создадим программу в среде программирования для микроконтроллеров AVR CodeVisionAVR. Откройте в главном меню CodeVisionAVR в группе HP InfoTech .

Создаём проект без мастера.



Сохраните проект в своей папке.

Выбираем наш chip ATMega8A.

После создания проекта откроется окно его конфигурации (его всегда можно открыть так: Project → Configure → C Compiler). Убедимся, что тактовая частота (Clock) = 8 MHz, остальное тоже оставляем дефолтное.

 

Комментарии (неисполняемый код) пишутся после двух слешей: // Управление портом С

Подключаем заголовочные файлы директивами

 #include <io.h> //(ввод/вывод) и

<delay.h> // (задержка).

Пишем главную функцию

void main(void)

 {

   DDRC=0b11111111; // Data Direction Registr - регистр-переключатель направления данных порта С.

К DDR подключены пины порта, он определяет их направление выдачи сигнала.

1 – пин настроен на вывод сигнала.

0 (по умолчанию) – на вход (считывание состояние).

0b – в двоичной системе.

Все единицы – значит, все пины порта настроены на вывод сигнала.

Выбор направления данных можно задать и 16-ричными цифрами, тогда получим равносильный оператор DDRC = 0xFF (вспомним тетрады: 1111(=F) 1111(=F)).

Итак, пишем инструкции в программе.

Для начала научимся подавать напряжение:

PORTC=0b00000001; // подаём на нулевой (младший) пин логическую единицу (5 В), а на остальных оставляем логический 0.

Пока все, закрываем программу

}

Вот что, по минимуму, должно получиться:



Компилируем код (**F9**). Если всё хорошо, увидим диалог с информацией о программе, если нет – то, кроме диалога и список ошибок в нижней части окна программы.

 

Исправим (если они есть) ошибки (дабл-клик по описанию ошибки выделит некорректную строку, либо следующую за ней, если предыдущую забыли закрыть точкой с запятой) и построим (по **Ctrl+F9**) hex-файл прошивки (попутно создаются и отладочные файлы в других форматах, напр. cof – для отладки программ CVAVR).

Бинарный файл .hex (который исполняется МК) будет в папке \Debug\Exe, а .cof – в \Debug.

**3. Прошивка МК в Proteus.Isis.**

В свойствах МК ATMega8 изменяем только CKSEL Fuses – выставляем в 0100, что задаёт тактирование МК от внутреннего RC-генератора на частоте 8MHz, и указываем в качестве ProgramFile созданный в CodeVisionAVR .cof–файл (выбор .cof -файла вместо .hexпозволит вести отладку по исходному коду):

****

Всё, схема должна заработать – должен загореться светодиод.

**4.** После того, как мы научились зажигать светодиод, сможем его и потушить.

Для этого в исходном коде добавьте команду задержки после включения

delay\_ms(1000); // пауза в 1 сек.

Но для того, чтобы она заработала, необходимо подключить библиотеку, в которой она прописана. Для этого в начале кода добавьте

#include <delay.h>

После задержки отключите светодиод сбросом в 0 младшего разряда порта С командой

PORTC=0b00000000;



Перепрошейте МК и проверьте в Протеусе работу схемы – светодиод должен включиться на секунду и потухнуть.

**5.** Мигание светодиода можно организовать в бесконечном цикле

while (1)

  {

  }

Перенесите в него команды установки напряжения и задержки (конфигурирование направления данных оставьте перед циклом), добавьте задержку после гашения светодиода и проверьте результат (мигание) в Протеусе.

**ЗАДАЧА 4**

Составить систему уравнений относительно комплексов тока в схеме по законам Кирхгофа, если

R1 = R2 = 100 Ом; С1 = С2 = С3 = 1 мФ; L = 100 мГн;

параметры идеальных источников ЭДС одинаковы: E1 = E2 = 220 В, ν = 15,9155, ϕ0 = 0.



**РЕШЕНИЕ**

Составим уравнения по законам Кирхгофа, предварительно задавшись положительными направлениями токов и обходов контуров.



Поскольку в цепи имеются 2 независимых узла и 3 контура, запишем два уравнения по первому и три – по второму закону Кирхгофа. Система уравнений примет вид (Xc берётся с противоположным знаком):

Матрица в символическом виде:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | I1 | I2 | I3 | I4 | I5 | E |
| узел 1 | -1 | 0 | -1 | 1 | 0 | 0 |
| узел 2 | 0 | -1 | -1 | 0 | 1 | 0 |
| контур I | 0 | -jXc | 0 | 0 | -jXc | E |
| контур II | 0 | 0 |  R+jXL | R | jXc | 0 |
| контур III | -jXc | 0 | 0 | R | 0 | E |

Находим циклическую частоту: ω = 2$πν$ = 100 рад/с

Вычисляем реактивные сопротивления: Xc = 1/ω⋅C = 10 Ом, XL = ω⋅L = 10 Ом.

Далее подставляем числовые значения в матрицу.

Опираясь на технологию работы с матрицами и использование комплексных чисел в программе MathCad, получаем решение:



*Примечание: при вводе мнимой единицы не нужно вводить знак \*. При вводе числа i без коэффициента необходимо вводить 1i.*















**(пример)**



Оглавление

[**ЗАДАЧА 1** 1](#_Toc504850317)

[**ЗАДАЧА 2** 4](#_Toc504850318)

[**ЗАДАЧА 3** 6](#_Toc504850319)

[**ЗАДАЧА 4** 9](#_Toc504850320)