МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Технология машиностроения»

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Методические рекомендации к практическим занятиям для студентов направления подготовки 15.03.06 «Мехатроника и робототехника» дневной формы обучения

Могилев 2019

УДК 621.8
ББК 34.5
И 38

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Технология машиностроения» «19» ноября 2018 г., протокол № 6

Составители: д-р техн. наук, проф. Г. Ф. Шатуров;
канд. техн. наук, доц. И. Д. Камчицкая

Рецензент канд. техн. наук, доц. В. В. Кутузов

Представлены методические рекомендации к практическим занятиям по дисциплине «Информационные технологии».

Учебно-методическое издание

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Ответственный за выпуск В. М. Шеменков

Технический редактор А. Т. Червинская

Компьютерная верстка М. М. Дударева

Подписано в печать . Формат 60x84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 46 экз. Заказ №

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

Содержание

1 Практическое занятие № 1. Разработка и реализация линейного алгоритма

Цель практической работы – приобретение практических навыков создания и отладки программы с линейной структурой в Microsoft Visio.

1.1 Понятие линейных алгоритмов

Линейный алгоритм – это алгоритм, в котором все действия выполняются в строгом порядке, последовательно, одно за другим.

Программа, реализующая линейный алгоритм, называется программой с линейной структурой.

Приступая к разработке программ с линейной структурой, следует учитывать, что:

– программы с линейной структурой являются простейшими и используются, как правило, для реализации простых вычислений по формулам;

– в программах с линейной структурой инструкции выполняются последовательно, одна за другой;

– алгоритм программы с линейной структурой может быть представлен в виде блок-схемы.

1.2 Общая характеристика Microsoft Office Visio

В целях обеспечения унификации, быстроты и удобства исполнения, повышения качества представляемых графических изображений различных бизнес-схем и диаграмм, компанией Microsoft создана программа Microsoft Office Visio. В настоящее время используются версии 2003, 2007 и 2010 данной программы. В ряде случаев простая программа MS Visio может заменить дорогостоящие графические процессоры и системы визуального моделирования деловых процессов. Графические изображения, созданные в MS Visio, можно вставлять в виде объектов в файлы других программных продуктов Microsoft.

Интерфейс MS Visio стандартный для программных продуктов Microsoft Office (рисунок 1.1). В центральной части расположена область редактирования документа, которая может содержать несколько страниц. Слева – окно «Фигуры» (Shapes), в котором отображаются выбранные пользователем трафареты с наборами фигур. Формат файлов MS Visio – *.vsd.

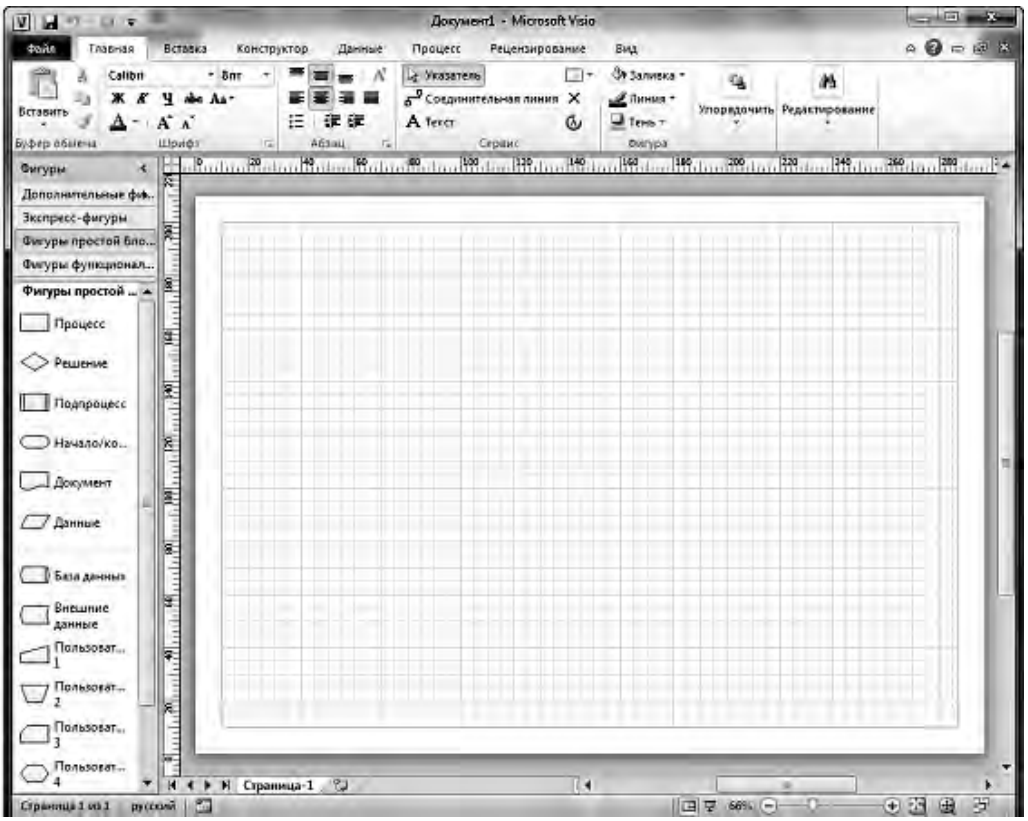


Рисунок 1.1 – Внешний вид Microsoft Office Visio

Шаблоны и трафареты Microsoft Office Visio

Шаблон (template) – это файл (*.vst), содержащий инструменты, установки, трафареты, фигуры, необходимые для сборки рисунков или диаграмм определенного типа.

Загрузка шаблона – команда «Файл» «Создать» (File New), или «Файл Фигуры» (File Shapes), или кнопка «Создать» (New) на панели инструментов, или кнопка «Фигуры» (Shapes).

Трафарет (stencils) – это файл (*.vss), содержащий фигуры, необходимые для сборки рисунков или диаграмм определенного типа.

Загрузка трафарета, не входящего в состав шаблона, – команда «Файл Фигуры Открыть набор элементов» (File ⇒ Shapes ⇒ Open stencils).

Шаблоны объединены в группы (категории). Каждый шаблон содержит трафареты, соответствующие его функциональному назначению.

Блок-схема алгоритма – графическое представление метода решения задачи, в котором используются символы для отображения операций и данных.

Конфигурацию, перечень и размеры условных изображений, а также правила построения схем алгоритмов устанавливает ГОСТ 19.701–90 «Схемы алгоритмов, программ, данных и систем».

На блок-схеме алгоритм представляется последовательностью графических символов, выполняющих определенные функции, и наличием связей между ними – линий, стрелок (потоками информации). Основное направление – сверху вниз и слева направо, при этом стрелки, указывающие направления, можно не ставить.

1.3 Методика и порядок построения блок-схемы

Пример – Требуется создать блок-схему алгоритма вычисления суммы факториалов.

Порядок выполнения примера.

1 Запустить программу MS Visio 2010 (или 2003, 2007).

2 В окне «Приступая к работе» выбрать в «Категориях шаблонов», расположенных в левой части, категорию «Блок-схема» (в версии 2003 Flowchart).

3 Выбрать шаблон «Простая блок-схема» (Basic Flowchart Shapes).

4 На панели инструментов выбрать масштаб 75 %.

5 Передвинуть границу между окном «Фигуры» (Shapes) и окном «Область вставки» так, чтобы «Область вставки» занимала не менее двух третей экрана.

6 Щелчок по кнопке «Текст» (Text Tool), в верхней части страницы области вставки нарисовать прямоугольник для будущего текста заголовка.

7 На панели инструментов установить шрифт Arial 18 пт жирный.

8 Ввести текст «СУММА ФАКТОРИАЛОВ», для окончания ввода щелкнуть вне прямоугольника текста на пустом месте (рисунок 1.2). Сохранить файл в своей рабочей папке.

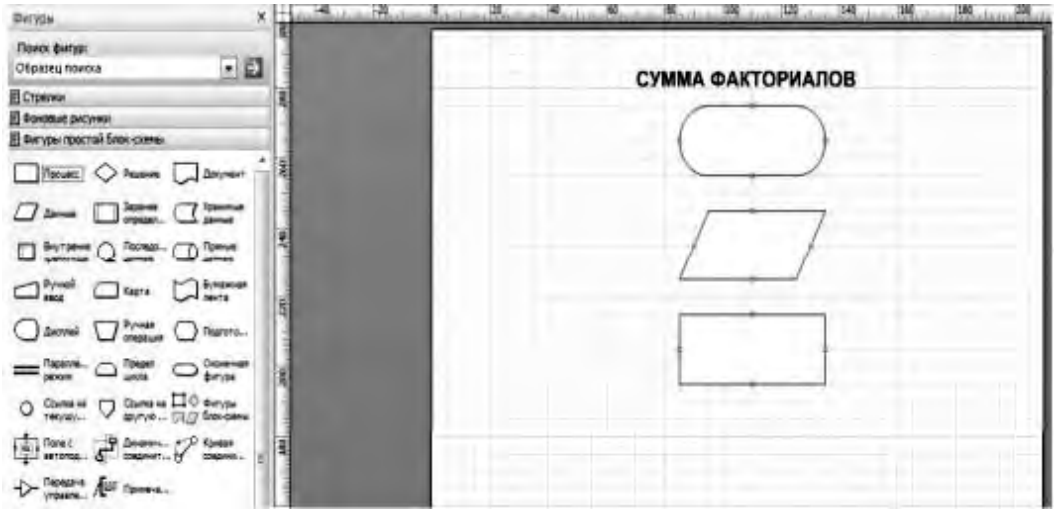


Рисунок 1.2 – Заготовка для построения алгоритма

9 Щелчок по кнопке «Указатель» (Pointer Tool), щелчок по введенному тексту, вокруг текста появится поле выделения с манипуляторами выделения.

10 Передвинуть мышью поле выделения текста в верхней части страницы. Для снятия выделения – щелчок вне прямоугольника выделения.

11 Перетащить с трафарета элемент «Оконечная фигура» (Terminator).

12 Передвинуть манипулятор в виде зеленого квадрата в нижнем левом углу поля выделения фигуры так, чтобы горизонтальный размер фигуры установился в десять клеток страницы, а вертикальный – в четыре.

13 Передвинуть полученную фигуру примерно под середину заголовка.

14 Аналогично перетащить элементы: «Данные» (Data), «Процесс» (Process), «Подготовка» (Preparation), «Заранее определенный процесс» (Predefined Process), «Документ» (Document) и, передвигая манипуляторы полей выделения (зеленые квадраты), установить одинаковые габаритные размеры фигур.

15 Скопировать элементы «Оконечная фигура» и «Процесс» с верхней части страницы в нижнюю часть страницы с помощью перетаскивания при нажатой клавише Ctrl. Результат представлен на рисунке 1.3.

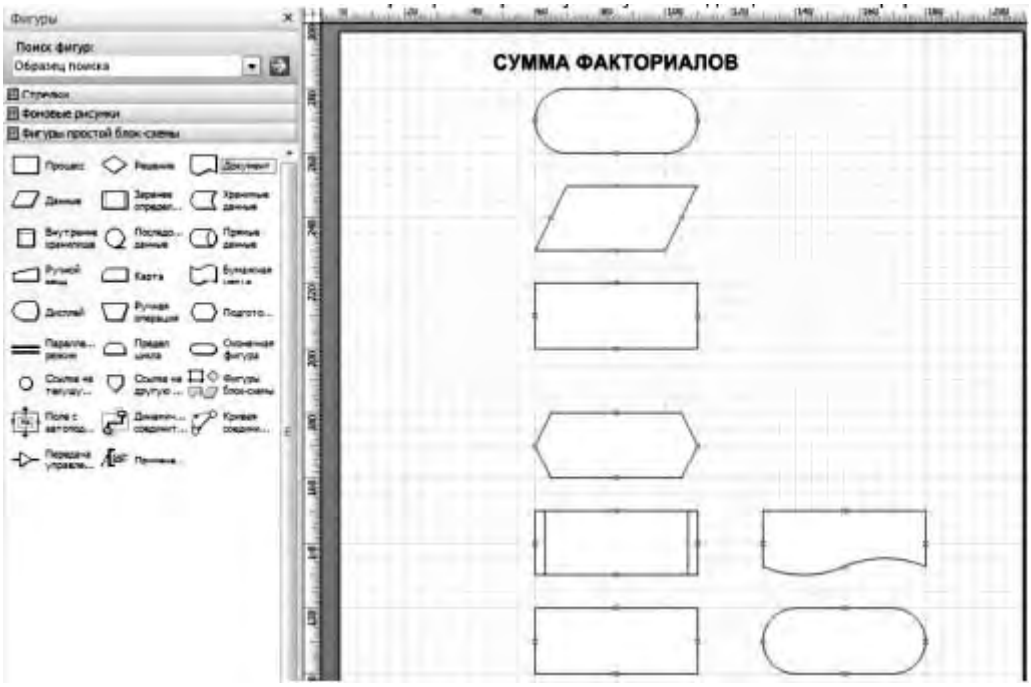


Рисунок 1.3 – Блоки алгоритма

16 Соединить две верхние фигуры. Для этого выполнить следующие действия:

– щелчок по кнопке «Соединительная линия» (Connector Tool) на панели инструментов;

– установить курсор на синий крест, расположенный в центре нижнего края фигуры «Оконечная фигура»; появление красного квадрата в этом месте означает наличие соединения;

– нажав левую клавишу мыши и удерживая ее, переместить курсор на синий крест, расположенный в центре верхнего края фигуры «Данные», появление красного квадрата в этом месте означает наличие соединения. Отпустить левую клавишу мыши.

17 Аналогично создать остальные соединения между элементами блок-схемы. При этом изгибы соединительных линий установятся автоматически.

18 Перенести соединительную линию, исходящую от последней фигуры «Процесс» и направленную вверх, на левую сторону схемы. Для этого выполнить следующие действия:

– щелкнуть по кнопке «Указатель»;

– щелкнуть по линии – появятся манипуляторы зеленого цвета;

– установить курсор на манипулятор, расположенный в середине вертикального сегмента линии, при этом курсор преобразуется в двуправленную стрелку;

– нажав левую клавишу мыши и удерживая ее, перетащить сегмент влево. Отмена выделения – клавиша Esc.

19 Ввести текст в элементы. Для того выполнить следующие действия:

– двойной щелчок внутри первой фигуры «Оконечная фигура»;

– на панели инструментов установить шрифт Times New Roman 14 пт;

– ввести текст с шрифтом Times New Roman 14 пт.

20 Изменить высоту шрифта во всех фигурах с помощью копирования формата. Для этого выполнить следующие действия:

– изменить высоту шрифта в тексте «Начало» первой фигуры до 18 пт;

– двойной щелчок по тексту «Начало» в первой фигуре – появится синее выделение текста;

– двойной щелчок по кнопке «Формат по образцу» (Format Painter) на панели инструментов;

– последовательно щелкая по текстам внутри элементов, перенести формат текста первой фигуры на все остальные;

– отжать кнопку «Формат по образцу»;

– снять выделение – кнопка Esc.

21 Изменить размер стрелок. Для этого выполнить следующие действия:

– на панели инструментов в списке кнопки «Указатель» выбрать «Выбор нескольких объектов» (Multiple Select);

– для выделения всех соединительных линий, удерживая нажатой клавишу Ctrl, последовательно щелкать по всем соединительным линиям;

– выполнить команду «Формат / Линия» (Format / Line);

– в окне «Линия» (Line) в строке «Конец» (End) выбрать стрелку 13, а в строке «Начальный размер» (Begin size) – «Очень крупный» (Extra Large), «Применить» (Apply), ОК;

– снять выделение клавишей Esc.

22 Ввести текст для соединительных линий фигуры с текстом «К=1+N». Порядок действий:

– двойной щелчок по соединительной линии между элементом с текстом «К=1+N» и элементом с текстом «Вывод SF»;

– в прямоугольнике с зеленой рамкой ввести текст «Нет»;

– щелчок вне прямоугольника на пустом месте;

– щелчок по слову «Нет». Внутри текста появится манипулятор в виде желтого ромба;

– перетащить этот манипулятор так, чтобы слово «Нет» расположилось над линией. Снять выделение клавишей Esc;

– выделить текст в любом блоке (фигуре). Появится синий прямоугольник;

– щелчок по кнопке «Формат по образцу»;

– щелчок по введенному слову «Нет». Снять выделение клавишей Esc;

– аналогично ввести слово «Да».

23 Выполнить контекстную команду «Переименовать» (Rename Page), выполненную на ярлыке «Страница 1» (Page 1). Ввести название страницы «Сумма факториалов» (рисунок 1.4).

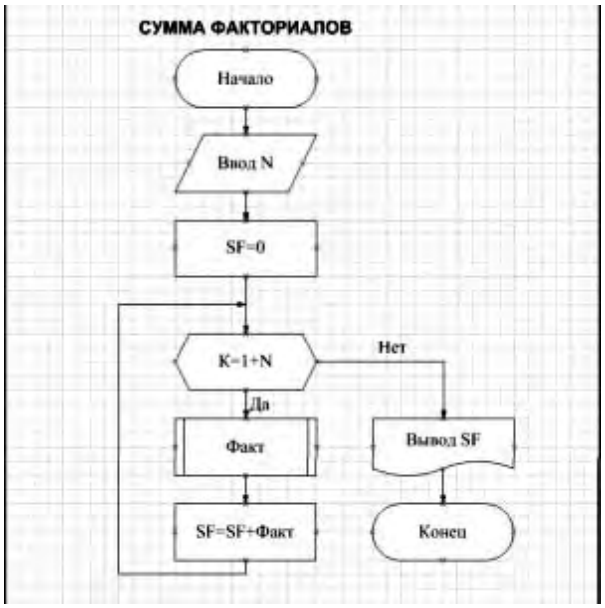


Рисунок 1.4 – Блок-схема алгоритма вычисления суммы факториалов

Порядок выполнения работы

Разработать линейный алгоритм и составить его блок-схему для вычисления выражения, указанного в индивидуальном варианте. Ввод исходных данных реализовать с клавиатуры.

Контрольные вопросы

1 Что такое линейный алгоритм?
2 Какая программа называется программой с линейной структурой?
3 Что представляет собой программа на языке С++?
4 Операторы и функции С++.

2 Практическое занятие № 2. Разработка и реализация ветвлений и циклов в вычислительных алгоритмах

Цель практической работы – приобретение практических навыков вычислений в Microsoft Visual Basic с использованием функции If, а также применяя циклы и разветвления.

2.1 Понятие ветвления

Ветвление – базовая алгоритмическая конструкция, являющаяся одним из самых популярных средств, изменяющих естественный порядок выполнения операторов программы. Алгоритмы, содержащие конструкцию ветвления, называют алгоритмами с ветвлением.

2.2 Разветвляющиеся алгоритмы

Алгоритм с ветвлением представлен на рисунке 2.1.

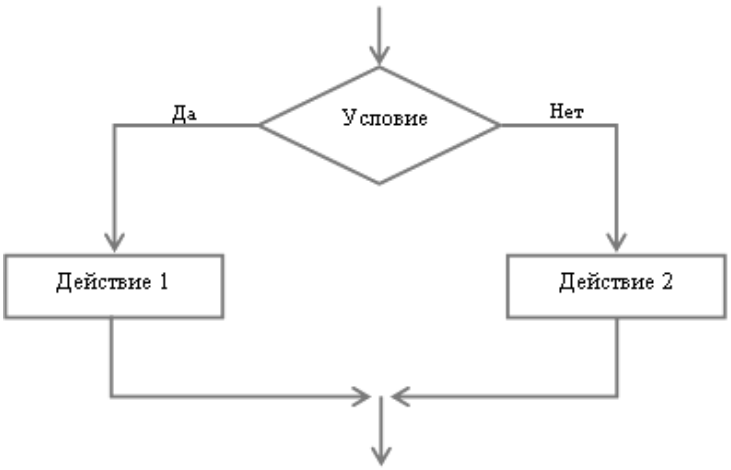


Рисунок 2.1 – Условие выбора

Пример – Разработать алгоритм вычисления наибольшего числа из двух чисел *x* и *y*.

Этап 1. Математическое описание решения задачи.

Из курса математики известно, если $x > y$, то наибольшее число x, если $x < y$, то наибольшее число y, если $x = y$, то число x равно числу y.

Этап 2. Определение входных и выходных данных.

Входными данными являются значения чисел *x* и *y*. Выходным данными являются:

– наибольшее число;

– любое из чисел, если числа равны.

Для решения задачи необходимо знать значения *x* и *y*.

Этап 3. Разработка алгоритма решения задачи.

В схеме алгоритма решения задачи (рисунок 2.2) цифрами указаны номера элементов алгоритма, которые соответствуют номерам шагов словесного описания алгоритма.

В рассматриваемом алгоритме имеются три ветви решения задачи:

– первая: это элементы 1, 2, 3, 4, 8;

– вторая: это элементы 1, 2, 3, 5, 6, 8;

– третья: это элементы 1, 2, 3, 5, 7, 8.

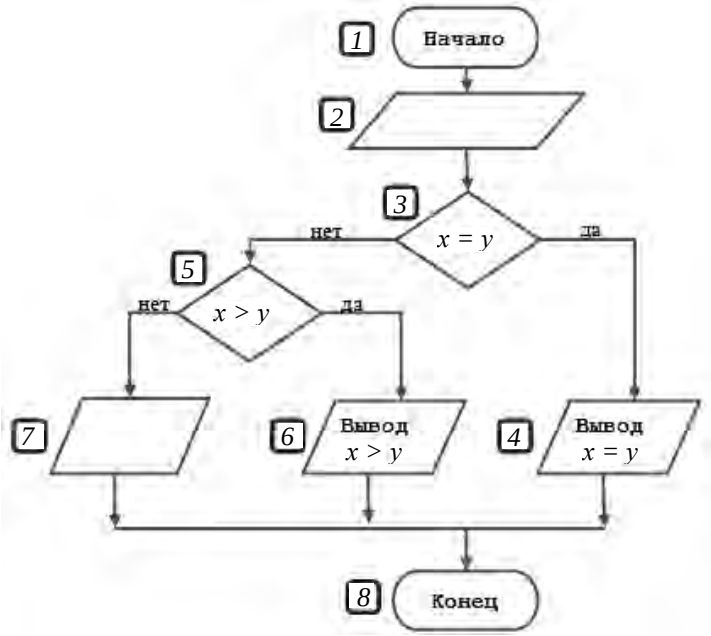
Словесное описание алгоритма	**Запись алгоритма на языке блок-схем**
1 Начало алгоритма. 2 Ввод значений *x* и *y*. 3 Сравниваем *x* и *y*. Если *x* = *y*, то переход к шагу 4, иначе к шагу 5. 4 Вывод информации: числа *x* и *y* равны. Переход к шагу 8. 5 Сравниваем *x* и *y*. Если *x* > *y*, то переход к шагу 6, иначе к шагу 7. 6 Вывод информации: число *x* больше *y*. Переход к шагу 8. 7 Вывод информации: число *y* больше *x*. Переход к шагу 8. 8 Конец алгоритма.	

Рисунок 2.2 – Схема алгоритма решения задачи

Выбор ветви определяется значениями *x* и *y* в элементах *3* и *5*, которые являются условиями, определяющими порядок выполнения элементов алгоритма. Если условие (равенство), записанное внутри символа «решение», выполняется при введенных значениях *x* и *y*, то следующими выполняются элементы *4* и *8*. Это следует из того, что они соединены линией с надписью «да» и направление (последовательность) вычислений обозначено стрелочкой.

Если условие в элементе *3* не выполняется, то следующим выполняется элемент *5*. Он соединен с элементом *3* линией с надписью «нет». Если условие, записанное в элементе *5*, выполняется, то выполняются элементы *6* и *8*, в противном случае – элементы *7* и *8*.

2.3 Циклические алгоритмы

Циклический алгоритм определяет повторение некоторой части действий (операций), пока не будет нарушено условие, выполнение которого проверяется

в начале цикла. Совокупность операций, выполняемых многократно, называется телом цикла (рисунок 2.3).

Рисунок 2.3 – Цикл

Алгоритмы, отдельные действия в которых многократно повторяются, называются *циклическими алгоритмами.* Совокупность действий, связанную с повторениями, называют *циклом*.

При разработке алгоритма циклической структуры выделяют следующие понятия:

– параметр цикла – величина, с изменением значения которой связано многократное выполнение цикла;

– начальное и конечное значения параметров цикла;

– шаг цикла – значение, на которое изменяется параметр цикла при каждом повторении.

Цикл организован по определенным правилам (рисунок 2.4). Циклический алгоритм состоит из подготовки цикла, тела цикла и условия продолжения цикла.

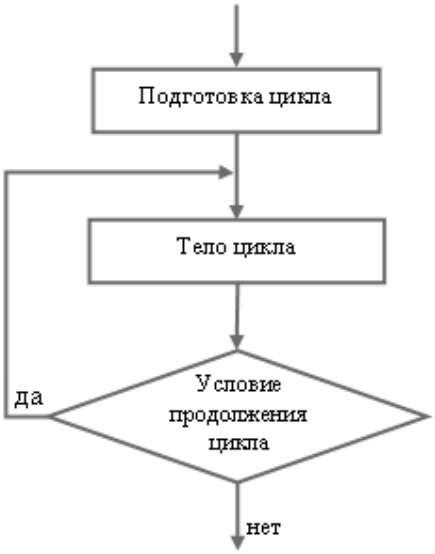


Рисунок 2.4 – Циклический алгоритм

В подготовку цикла входят действия, связанные с заданием исходных значений для параметров цикла:

– начальные значения цикла;

– конечные значения цикла;

– шаг цикла.

В тело цикла входят:

– многократно повторяющиеся действия для вычисления искомых величин;

– подготовка следующего значения параметра цикла;

– подготовка других значений, необходимых для повторного выполнения действий в теле цикла.

В условии продолжения цикла определяется допустимость выполнения повторяющихся действий. Если параметр цикла равен или превысил конечное значение цикла, то выполнение цикла должно быть прекращено.

Пример – Разработать алгоритм вычисления суммы натуральных чисел от 1 до 100.

Этап 1. Математическое описание решения задачи.

Обозначим сумму натуральных чисел через *S*. Тогда формула вычисления суммы натуральных чисел от 1 до 100 может быть записана в виде

$$S = 1 + 2 + 3 + ... + 97 + 98 + 99 + 100 = \sum_{i=1}^{n} X_i, \qquad (2.1)$$

где X_i – натуральное число X с номером i, который изменяется от 1 до n;

n – количество натуральных чисел, n = 100.

Этап 2. Определение входных и выходных данных.

Входными данными являются натуральные числа: 1, 2, 3, 4, …, 98, 99, 100.

Выходные данные – значение суммы членов последовательности натуральных чисел.

Параметр цикла – величина, определяющая количество повторений цикла. В нашем случае i – номер натурального числа.

Подготовка цикла заключается в задании начального и конечного значений параметра цикла:

– начальное значение параметра цикла равно 1;

– конечное значение параметра цикла равно n;

– шаг цикла равен 1.

Для корректного суммирования необходимо предварительно задать начальное значение суммы, равное 0.

Тело цикла. В теле цикла будет выполняться накопление значения суммы чисел, а также вычисляться следующее значение параметра цикла по формулам $S = S + i$; $I = I + 1$.

Условие продолжения цикла: цикл должен повторяться до тех пор, пока не будет добавлен последний член последовательности натуральных чисел,

т. е. пока параметр цикла будет меньше или равен конечному значению параметра цикла.

Этап 3. Разработка алгоритма решения задачи.

Введем обозначения: *S* – сумма последовательности; *i* – значение натурального числа. На рисунке 2.5 представлен алгоритм решения данной задачи. Начальное значение цикла *i* = 1, конечное значение цикла *i* = 100, шаг цикла 1.

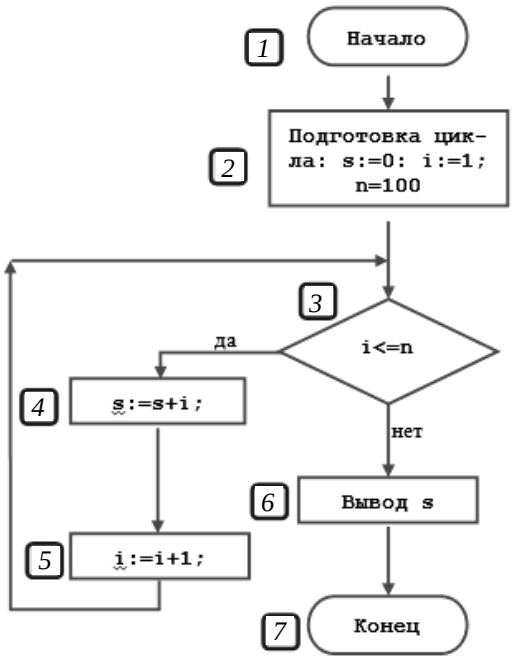
Словесное описание алгоритма	**Запись алгоритма на языке блок-схем**
1 Начало алгоритма. 2 Подготовка цикла: *S*:=0; *i*=1; *n*=100; 3 Проверка условия. Если i <=*n* , то перейти к шагу 4, иначе к шагу 6. 4 Накопление суммы: *S*:=*S*+*i*; 5 Вычисление следующего значения параметра цикла: *i*:=*i*+1; 6 Вывод информации: сумма натуральных чисел – *S*. 7 Конец алгоритма .	В схеме алгоритма решения задачи цифрами указаны номера элементов алгоритма. Номера элементов соответствуют номерам шагов словесного описания алгоритма. 

Рисунок 2.5 – Разработанный алгоритм решения задачи

Порядок выполнения работы

Разработать алгоритм решения задачи, указанной в индивидуальном варианте, на основе циклических конструкций. Составить блок-схему алгоритма для вычисления выражения.

Контрольные вопросы

1 Что понимают под термином «разветвляющаяся вычислительная структура»?

2 Как строится схема алгоритма разветвляющейся вычислительной структуры? Какой символ осуществляет проверку некоторых условий?

3 Отчего зависит количество ветвей в алгоритме?

3 Практическое занятие № 3. Исследование программных элементов раздела «Таймеры»

Цель практической работы – изучение назначения, режимов работ таймера-счетчика микроконтроллера ATmega. Получение практических навыков программирования таймера-счетчика.

3.1 Микроконтроллер ATmega128

В состав микроконтроллера ATmega128 входит три 16-разрядных счетчика и один 8-разрядный таймер-счетчик 0. 16-разрядные таймеры-счетчики предназначены для точного задания временных интервалов, генерации прямоугольных импульсов и измерения временных характеристик импульсных сигналов.

Таймер-счетчик 0 – модуль многофункционального одноканального 8-разрядного таймера-счетчика с аппаратным выходом для генерации ШИМ-сигнала и встроенным асинхронным опциональным тактовым генератором, который оптимизирован под использование часового кварца (32768 Гц) для асинхронного по отношению к системной синхронизации тактирования.

На данном практическом занятии рассматривается работа таймера-счетчика 0 при нормальном режиме работы и режиме сброса таймера при совпадении (CTC).

3.2 Функциональная схема. Назначение регистров

Укрупненная функциональная схема 8-разрядного таймера-счетчика представлена на рисунке 3.1.

Регистр таймера-счетчика (TCNT0) и регистр порога сравнения (OCR0) – 8-разрядные регистры.

Сигналы запроса на прерывание представлены как флаги прерываний таймера в регистре TIFR.

Все прерывания индивидуально маскируются с помощью регистра маски прерываний таймеров (TIMSK). Регистры TIFR и TIMSK не представлены на функциональной схеме, т.к. они совместно используются с другими таймерами микроконтроллера.

Таймер-счетчик может тактироваться через предделитель внутренне или асинхронно через внешние выводы TOSC1/2. Асинхронная работа управляется регистром асинхронного состояния (ASSR). Блок синхронизации осуществляет выбор, какой тактовый источник используется для инкрементирования (декрементирования) состояния таймера-счетчика. Если источник тактирования не задан, то таймер-счетчик находится в неактивном состоянии. Выход логики выбора синхронизации обозначен как синхронизация таймера (clkT0).

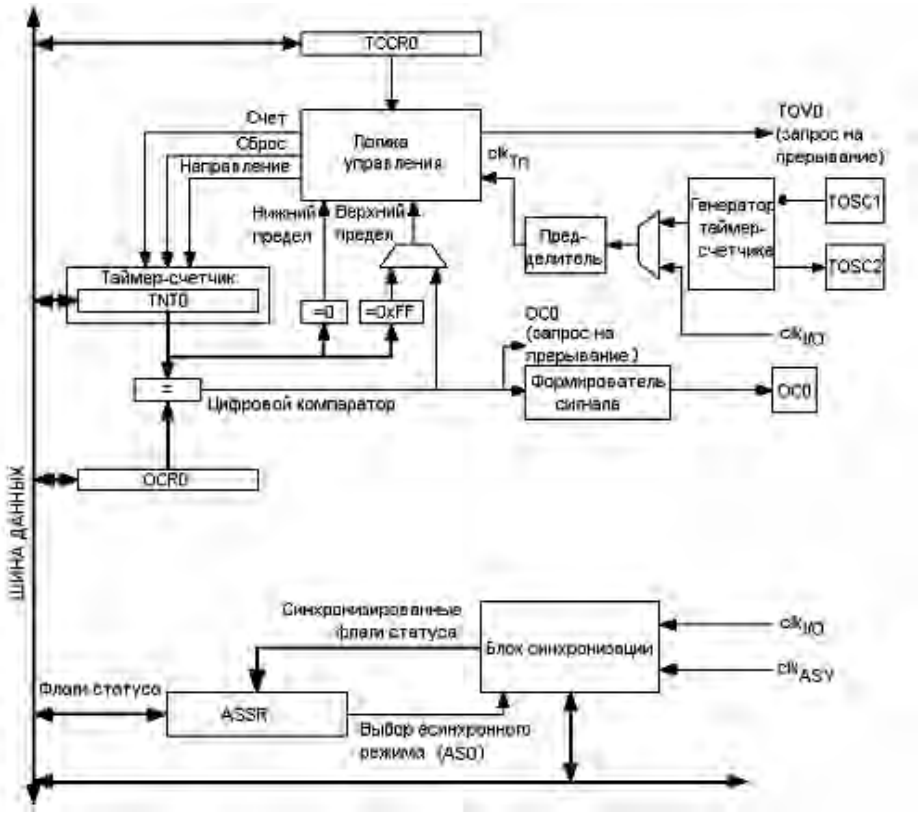


Рисунок 3.1 – Функциональная схема 8-разрядного таймера-счетчика

Значение регистра порога сравнения с двойной буферизацией (OCR0) непрерывно сравнивается со значением таймера-счетчика. Результат сравнения может использоваться для генерации сигналов с ШИМ или прямоугольных импульсов переменной частоты на выводе OC0.

Совпадение порога сравнения со значением таймера-счетчика приводит к установке флага результата сравнения (OCF0), который может использоваться для генерации запроса на прерывание по результату сравнения.

Таймер-счетчик 0 может тактироваться внутренне синхронно или внешне асинхронно (по отношению к внутренней системной синхронизации). По умолчанию используется тактовый сигнал clkT0, эквивалентный тактовому сигналу микроконтроллера clkI/O. Основу 8-разрядного таймера-счетчика 0 составляет программируемый двунаправленный счетчик. Рисунок 3.2 показывает функциональную схему счетчика и окружающих его элементов.

Рисунок 3.2 – Функциональная схема счетчика

В зависимости от выбранного режима работы счетчик сбрасывается, инкрементируется или декрементируется на каждом такте синхронизации (clkT0). Тактовый сигнал clkT0 может быть внутренним или внешним, а его частота выбирается с помощью бит выбора частоты синхронизации CS02-CS00 (регистр управления TCCR0). Если источник синхронизации не задан (CS02-CS00 = 0b000), то таймер останавливается.

Последовательность счета определяется установкой бит WGM01 и WGM00, расположенных в регистре управления таймером-счетчиком (TCCR0).

8-разрядный цифровой компаратор непрерывно выполняет сравнение содержимого регистра таймера-счетчика TCNT0 с регистром порога сравнения OCR0. Всякий раз, когда значение TCNT0 совпадает со значением OCR0, компаратор устанавливает флаг совпадения OCF0 следующим тактом синхронизации таймера. Если разрешено прерывание битом OCIE0 = 1, то установка флага совпадения вызывает запрос на прерывание. Флаг OCF0 автоматически сбрасывается во время выполнения процедуры обработки прерывания.

3.3 Режимы работы таймера-счетчика 0

Режим работы таймера задается комбинацией бит, задающих режим работы таймера (WGM01, WGM00) . Биты задания режима формирования выходного сигнала (COM01, COM00) не влияют на алгоритм счета, т. к. алгоритм счета зависит только от состояния бит задания режима работа таймера. В режимах с ШИМ биты COM01, COM00 позволяют включить/отключить инверсию на генерируемом ШИМ-выходе (т. е. выбрать ШИМ с инверсией или ШИМ без инверсии). Для режимов без ШИМ биты COM01:0 определяют, какое действие необходимо выполнить при выполнении условия сравнения: сбросить, установить или инвертировать выход.

3.4 Нормальный режим работы

Самым простым режимом работы является нормальный режим (WGM01, WGM00 = 0b00). В данном режиме счетчик работает как суммирующий (инкрементирующий), при этом сброс счетчика не выполняется. Переполнение счетчика происходит при переходе через максимальное 8-разрядное значение (верхний предел = 0xFF) к нижнему пределу счета (0x00). В нормальном режиме работы флаг переполнения таймера-счетчика TOV0 будет установлен на том же такте синхронизации, когда TCNT0 примет нулевое значение. Восьмибитный таймер 0 может отсчитать до импульсов без переполнения, то есть он считает от 0 до 255. Следующее значение – снова нуль и так далее. При переходе от состояния 255 к нулю в счетчике таймера возникает переполнение, что вызывает прерывание по переполнению таймера.

При выборе максимального коэффициента деления 1024 (рисунок 3.4) и частоте тактового генератора 4 Мгц максимальное время, в течение которого таймер может считать без переполнения, составит

$$\frac{1024 \cdot 2^8}{4000000} = 0,065536 \text{ с.}$$

3.5 Режим сброса таймера при совпадении

В режиме CTC (WGM01, WGM00 =0b10) регистр OCR0 используется для задания разрешающей способности счетчика. Если задан режим CTC и значение счетчика (TCNT0) совпадает со значением регистра OCR0, то счетчик обнуляется (TCNT0=0). Таким образом, OCR0 задает вершину счета счетчика, а следовательно, и его разрешающую способность. В данном режиме обеспечивается более широкий диапазон регулировки частоты генерируемых прямоугольных импульсов.

Временная диаграмма для режима CTC показана на рисунке 3.3. Значение счетчика (TCNT0) инкрементируется до тех пор, пока оно не станет равным значению в OCR0, после чего счетчик (TCNT0) обнулится.

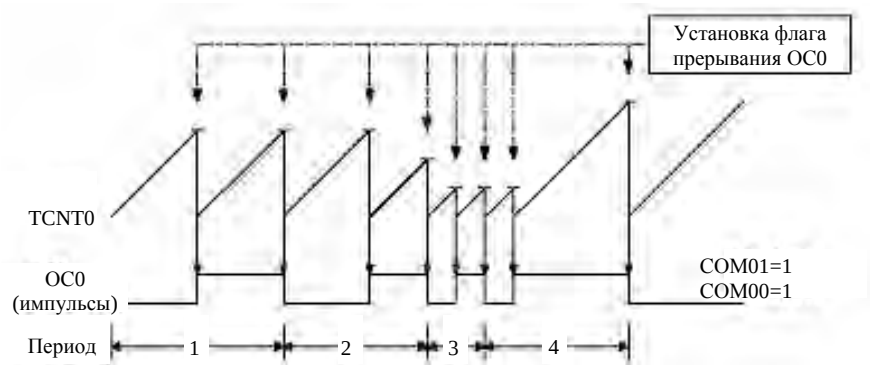


Рисунок 3.3 – Временная диаграмма для режима CTC

С помощью флага OCF0 прерывание может генерироваться всякий раз, когда счетчик достигает своего верхнего предела счета. Если работа прерывания разрешена, то процедура обработки прерывания может использоваться для обновления значения вершины счета.

Для генерации сигнала в режиме CTC выход компаратора OC0 может использоваться для изменения логического уровня при каждом совпадении, для чего необходимо задать режим переключения (COM01, COM00 = 0b01). Значение OC0 будет присутствовать на выводе порта, только если для данного вывода задано выходное направление. Максимальная частота генерируемого сигнала fOC0 = fclk_I/0/2, если OCR0 = 0x00. Для других значений OCR0 частоту генерируемого сигнала можно определить по формуле

$$f_{OC0} = \frac{f_{clk\_I/0}}{2N \cdot (1 + OCR0)}, \quad (3.1)$$

где переменная *N* задает коэффициент деления предделителя (1, 8, 32, 64, 128, 256 или 1024).

Как и для нормального режима работы, флаг TOV0 устанавливается на том же такте таймера, когда его значение изменяется с 0xFF на 0x00.

3.6 Предделитель таймера-счетчика 0

Тактовый источник таймера-счетчика 0 обозначен как clkT0. По умолчанию clkT0 подключен к системному источнику синхронизации ввода-вывода clkI/0.

Предделитель таймера-счетчика 0 позволяет выбрать следующие тактовые сигналы: clkT0S/8, clkT0S/32, clkT0S/64, clkT0S/128, clkT0S/256 и clkT0S/1024. Кроме того, имеется возможность остановить синхронизацию.

Установка необходимого тактового сигнала (clkT0S/8, clkT0S/32, clkT0S/64, clkT0S/128, clkT0S/256 и clkT0S/1024) осуществляется изменением бит CS02:00 регистра управления TCCR0.

3.7 Регистры таймера-счетчика

Регистр управления таймером-счетчиком 0 – TCCR0 представлен на рисунке 3.4.

Разряд	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Чтение/запись	Чт.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Рисунок 3.4 – Регистр управления таймером-счетчиком 0 – TCCR0

Разряд 7 – FOC0: Принудительная установка результата сравнения.
Разряды 6, 3 – WGM01:0: Режим работы таймера-счетчика 0.
Разряд 5:4 – COM01, COM00: режим формирования выходного сигнала.
Разряд 2:0 – CS02:0: настройка частоты синхронизации таймера.

С помощью трех настроечных бит имеется возможность выбрать различные тактовые частоты, кратные исходной частоте синхронизации (рисунок 3.5).

CS02	CS01	CS00	Описание
0	0	0	Нет синхронизации. Таймер-счетчик 0 оставлен.
0	0	1	clkT0S/1 (без предделения)
0	1	0	clkT0S/8 (с предделением)
0	1	1	clkT0S/32 (с предделением)
1	0	0	clkT0S/64 (с предделением)
1	0	1	clkT0S/128 (с предделением)
1	1	0	clkT0S/256 (с предделением)
1	1	1	clkT0S/1024 (с предделением)

Рисунок 3.5 – Тактовые частоты

Регистр таймера-счетчика – TCNT0 и регистр порога сравнения – OCR0 представлены на рисунке 3.6.

Разряд	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряд	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Рисунок 3.6 – Регистр таймера-счетчика – TCNT0 и регистр порога сравнения – OCR0

Регистр маски прерываний таймеров-счетчиков – TIMSK, счетчиков – TIFR общий для всех счетчиков (рисунок 3.7).

Разряд	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Рисунок 3.7 – Регистр TIMSK

Разряд 1 – OCIE0: Разрешение прерывания по результату сравнения таймера-счетчика 0.

Если OCIE0=1, а также установлен бит I в регистре статуса, то прерывание по результату сравнения таймера-счетчика 0 активизируется. В этом случае прерывание возникает, если обнаруживается совпадение значения таймера-счетчика 0 с порогом сравнения, т. е. когда установлен флаг OCF0 в регистре флагов прерываний таймеров-счетчиков TIFR.

Разряд 0 – TOIE0: Разрешение прерывания по переполнению таймера-счетчика 0.

Если TOIE0=1, а также установлен бит I в регистре статуса, то прерывание по переполнению таймера-счетчика 0 разрешается. В этом случае запрос на прерывание генерируется, если обнаруживается переполнение таймера-счетчика 0, т. е. когда установлен флаг TOV0 в регистре флагов прерываний таймеров-счетчиков TIFR (рисунок 3.8).

Разряд	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Рисунок 3.8 – Условие прерывания по переполнению таймера-счетчика

Разряд 1 – OCF0: Флаг совпадения таймера-счетчика 0 OCF0 равен лог. 1, если обнаруживается совпадением между значением таймера-счетчика 0 и данными в регистре OCR0 (регистр порога сравнения). OCF0 сбрасывается аппаратно при переходе на соответствующий вектор прерывания. Альтернативно флаг OCF0 может быть сброшен путем записи в него лог. 1. Если установлены бит I в регистре SREG, бит OCIE0 (разрешено прерывание по выполнению условия сравнения таймера-счетчика 0) и флаг OCF0, то генерируется прерывание по выполнению условия сравнения таймера-счетчика 0.

Разряд 0 – TOV0: Флаг переполнения таймера-счетчика 0 Флаг TOV0 устанавливается, если в таймере-счетчике 0 возникает переполнение. Флаг TOV0 сбрасывается аппаратно при переходе на соответствующий вектор прерывания. Альтернативно флаг TOV0 сбрасывается путем записи в него лог. 1. Если установлены бит I в регистре SREG, бит TOIE0 (разрешено прерывание по переполнению таймера-счетчика 0) и флаг TOV0, то генерируется прерывание по переполнению таймера-счетчика 0. В режиме ШИМ данный флаг устанавливается, если таймер-счетчик 0 изменяет направление счета на значении 0x00.

Порядок выполнения работы

Запрограммировать микроконтроллер для выполнения задачи, указанной в индивидуальном задании.

Контрольные вопросы

1 Опишите систему команд микроконтроллера.
2 Опишите режимы работы микроконтроллера.
3 Понятие таймера-счетчика.
4 Режимы работы тамера-счетчика.

4 Практическое занятие № 4. Исследование программных элементов раздела «Счетчики»

Цель практической работы – изучение назначения счетчиков, исследование режимов работы и характеристик счетчика в программе Multisim.

4.1 Понятие «счетчик»

Счётчик предназначен для счёта поступающих на его вход импульсов, в интервале между которыми он должен хранить информацию об их количестве. Поэтому счётчик состоит из запоминающих ячеек – триггеров обычно *D*- или *JK*-типа. Между собой ячейки счётчика соединяют таким образом, чтобы каждому числу импульсов соответствовали состояния 1 или 0 определенных ячеек. При этом совокупность единиц и нулей на выходах *n* ячеек, называемых

разрядами счетчика, представляет собой *n*-разрядное двоичное число, которое однозначно определяет количество прошедших через входы импульсов.

Каждый разряд счётчика может находиться в двух состояниях. Число устойчивых состояний, которое может принимать данный счётчик, называют *коэффициентом пересчёта* $K_{сч}$.

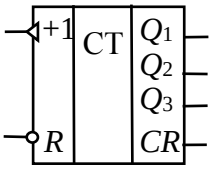
Если с каждым входным импульсом «записанное» в счётчике число увеличивается, то такой счётчик является *суммирующим*, если же оно уменьшается, – *вычитающим*. Счётчик, работающий как на сложение, так и на вычитание, называют *реверсивным*.

Счётчики, у которых под воздействием входного импульса переключение соответствующих разрядов происходит последовательно друг за другом, называют *асинхронными*, а когда переключение происходит одновременно – *синхронными*. Максимальное число N, которое может быть записано в счётчике, равно $(2^n - 1)$, где n – число разрядов счётчика.

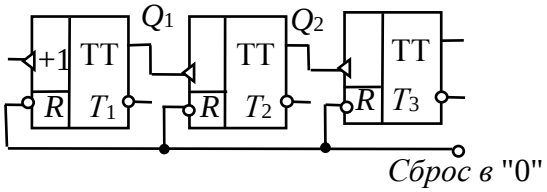
По способу кодирования последовательных состояний различают *двоичные счетчики* с коэффициентами пересчёта (обнуления) $K_{сч} = 2^n$, у которых порядок смены состояний триггеров соответствует последовательности двоичных чисел, и *недвоичные*, у которых $K_{сч} < 2^n$ (например, десятичные с коэффициентом $K_{сч} = 10$ или делители частоты с коэффициентом деления $K_{сч} \neq 2^n$).

Условное изображение трехразрядного *суммирующего* счётчика показано на рисунке 4.1, *а*, на котором символом R обозначен вход общего сброса, символами Q_1, Q_2 и Q_3 – выходы счетчика, CR – выход переноса единицы.

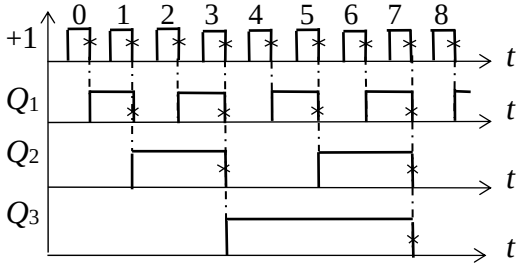
а)



б)



в)



г)

	Q_3	Q_2	Q_1	CR
0	0	0	0	
1	0	0	1	
2	0	1	0	
3	0	1	1	
4	1	0	0	
5	1	0	1	
6	1	1	0	
7	1	1	1	
				1
0	0	0	0	

а – условное обозначение; *б* – реализация на триггерах; *в* – временные диаграммы работы; *г* – таблица переключений

Рисунок 4.1 – Трёхразрядный суммирующий счётчик

Суммирующий вход счётчика обозначается +1, вычитающий –1. Это счетные входы. У асинхронных счётчиков данные входы помечены специальными символами, указывающими полярность перепада входного сигнала: 1/0 (задний фронт) или 0/1 (передний фронт), при которой происходит переключение триггеров счётчика. Для переключения триггеров в счётчиках используют следующие связи: непосредственную, тракт последовательного переноса, тракт параллельного переноса.

Схема счётчика с непосредственными связями показана на рисунке 4.1, *б*. Первый триггер T_1 счётчика образует младший разряд. Он пересчитывает входные импульсы по модулю 2, а состояние его выхода воспринимается следующим триггером T_2 как входные сигналы и снова пересчитываются на 2 и т. д.

Полное представление о состояниях счётчика, в зависимости от числа поданных на вход импульсов, дают таблица переключений (рисунок 4.1, *г*) и временные диаграммы (рисунок 4.1, *в*), где изображены последовательность входных импульсов (на входе +1), а также состояния триггеров – первого (Q_1), второго (Q_2) и третьего (Q_3). Фронты импульсов на диаграммах показаны идеальными: потенциал, соответствующий логическому 0, считается равным нулю, переключающие перепады для наглядности помечены крестиками.

Рассмотрим воздействие на счётчик, к примеру, шестого (обозначенного на диаграмме цифрой 5) импульса. По его спаду триггер T_1 устанавливается в 0, перепад 1/0 на его выходе Q_1 переключает в 1 триггер T_2, а триггер T_3 остается в прежнем (единичном) состоянии, так как перепад 0/1 на выходе Q_2 не является для него переключающим.

Из диаграммы видно, что частота импульсов на выходе каждого триггера вдвое меньше частоты импульсов на его входе. В момент, предшествующий переключению очередного разряда, все предыдущие разряды счётчика находятся в состоянии 1. Восьмой импульс для трехразрядного счётчика (рисунок 4.1, *г*) является импульсом переполнения: им все триггеры устанавливаются в 0 (счётчик «обнуляется»).

Если в счётчике используются триггеры, переключающиеся перепадом 0/1, то вход последующего триггера нужно соединить с инверсным выходом предыдущего, на котором формируется этот перепад, когда по основному выходу триггер переключается из 1 в 0.

Порядок выполнения работы

В программе Multisim собрать схему для испытания заданного преподавателем синхронного двоичного счётчика (рисунок 4.2) и установить в диалоговых окнах компонентов их параметры или режимы работы.

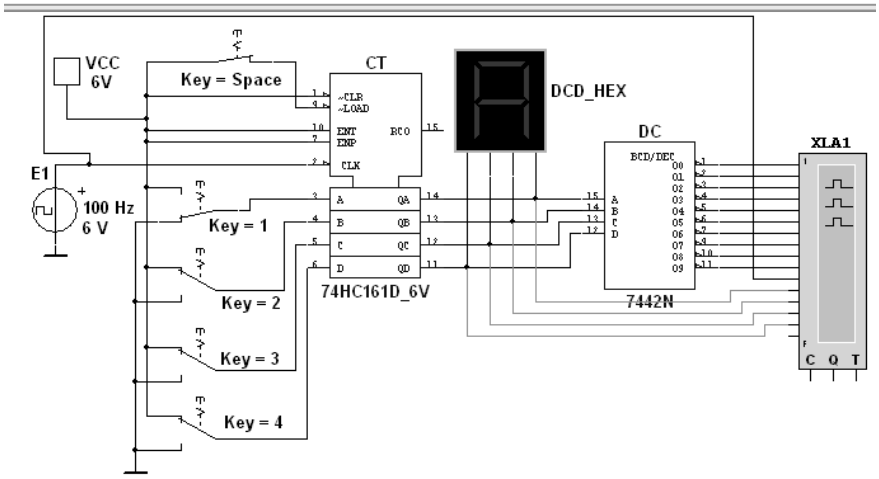


Рисунок 4.2 – Схема для моделирования работы синхронного двоичного счётчика

В схему включен синхронный двоичный 4-разрядный счётчик 74HC161, к входу $\overline{CLK}$ которого подключен источник тактовых импульсов E1, а к выходам QA, QB, QC и QD – шестнадцатеричный 7-сегментный индикатор DCD_HEX и дешифратор DC 4x10. Выходы счётчика и дешифратора соединены с входами логического анализатора XLA1.

К входам A, B, C и D счётчика CT подключены источник постоянного напряжения VCC, переключатели 1–4 для формирования входных двоичных кодов и ключ Space для изменения режима работы счётчика. В синхронном счётчике заданные с помощью ключей уровни сигналов подаются на входы всех триггеров, как и тактовые импульсы, которые подаются на счётные входы $\overline{CLK}$ всех разрядов счётчика.

При *замкнутом* ключе Space число поданных от генератора E1 на вход счётчика импульсов высвечивается на индикаторе DCD_HEX в десятичном коде, от 0 до 15, после чего счётчик обнуляется и вновь начинается счёт. При этом на одном из выходов дешифратора DC формируется сигнал низкого уровня (логический 0), номер которого соответствует коду входного числа: от 0000 до 1001 (9_{10}).

При *разомкнутом* ключе Space сформированное с помощью переключателей на входе счётчика 4-разрядное двоичное число высвечивается на индикаторе в десятичном коде, а на экране анализатора на одном из выходов, соответствующем входному коду счётчика, формируется логический 0.

Промоделировать работу счётчика и дешифратора, сняв временные диаграммы их работы (рисунок 4.3).

Разомкнуть ключ Space. Установить в диалоговом окне анализатора XLA1 напряжение V = 5 В, частоту таймера f_a = 2 кГц, число импульсов, приходящихся на одно деление, Clocks/div = 60 (при таком режиме лучи медленно перемещаются на экране анализатора). С помощью активных клавиш 1, 2, 3 и 4 клавиатуры сформировать произвольные (или по указанию преподавателя) двоичные входные числа (коды), например 1001, 0011, 0000, 1110, и подавать их на входы D, C, B и A счётчика. Зафиксировать показание семисегментного индикатора и дешифратора при данных значениях кода.

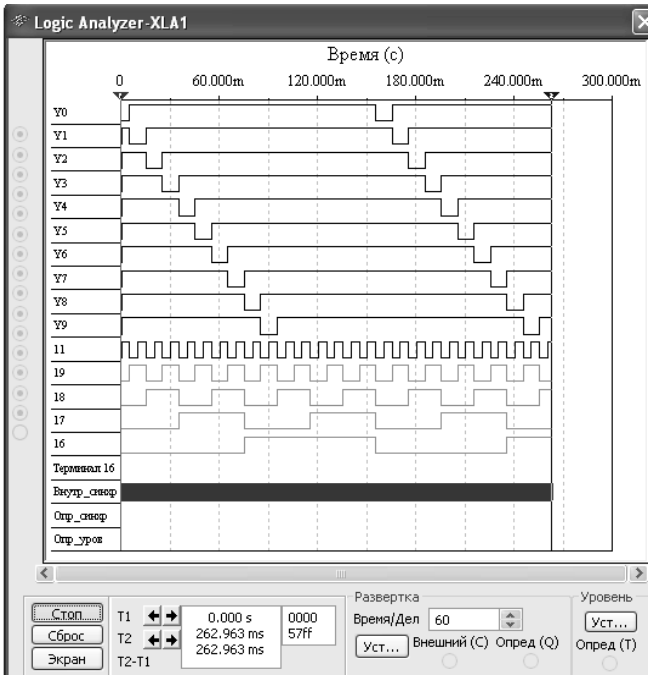


Рисунок 4.3 – Временные диаграммы работы счётчика и дешифратора

Контрольные вопросы

1 Назначение счетчика.
2 Суммирующий и вычитающий счетчики.
3Асинхронные, синхронные счетчики.
4 Двоичные, недвоичные счетчики.

5 Практическое занятие № 5. Исследование программных элементов по передаче информации

Цель практической работы – получение навыков записи в микроконтроллер, стирание, перезаписи и исследования схемы, управляемой программой микроконтроллера.

5.1 PIC

Peripheral-Interface Controller (периферийный контроллер интерфейса). Это малогабаритное интегральное электронное устройство, в которое можно записать (ввести) программу. Действует данная программа автоматически, выдает на выводы микроконтроллера (выводы регистра данных порта) сигналы «нуль» или «единица» («низкий» или «высокий» уровни напряжения, примерно 0,1или 4,9 В).

Представленные обычно в двоичном коде, эти сигналы управляют различного типа устройствами автоматики, начиная с простейших электронных часов, заканчивая робототехникой. То же самое, что делают программы в микроконтроллерах, можно реализовать на дискретных (отдельных) электронных

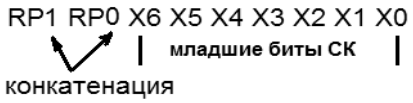
схемах, но габариты при этом увеличиваются (от нескольких раз до десятков тысяч раз). Тем не менее и для микроконтроллера, интегрального и малогабаритного, требуется компьютер, с помощью которого разрабатывается программа; программатор – отдельное устройство (вносит программу в микроконтроллер).

5.2 Организация памяти

В микроконтроллерах PIC существует два блока памяти – память программ и память данных (гарвардская структура микроконтроллера) в отличие от структуры ЭВМ (структура Неймана – данные хранятся наравне с командами). Каждый блок имеет собственную шину, таким образом, доступ к блокам может происходить одновременно (минимизация времени). Память данных, в свою очередь, разделена на специальные регистры и регистры общего применения (ОЗУ пользователя). Специальные регистры применяются для хранения битов состояния, определяющих работу портов ввода/вывода, таймеров и других периферийных модулей контроллера. Подробно каждый специальный регистр описан при рассмотрении соответствующего модуля.

Кроме специальных регистров и ОЗУ, пространство памяти данных может содержать ячейки постоянной памяти EEPROM(в некоторых микроконтроллерах отсутствует), запись и перезапись которых осуществляется электрическим способом. Эта область памяти не может быть адресована непосредственно, и доступ к ней получают через специальный регистр косвенной адресации EEADR, в который записывают порядковый номер ячейки. Обычно EEPROM используется для хранения констант, значения которых не должны пропадать при отключении питания, например кодов управления, индивидуальных номеров и т.п. Важным достоинством EEPROM является то, что данные в ней могут быть изменены даже после занесения программы в однократно программируемый кристалл. Число циклов перезаписей перезапуска 10 000.

Микроконтроллеры группы PIC имеют 13-битный счётчик команд (СК) (содержит адрес памяти, с которого считывается команда на выполнение, а его значение автоматически увеличивается на единицу – автоинкремент). 13 разрядов позволяют адресовать до 8К (К=2^10=1024) ячеек памяти. Семь младших бит(разрядов)счётчика (так решил производитель) работают в соответствии с автоинкриментом и позволяют адресовать 2^7 =128 адресов. Так как по каждому адресу хранится информация в 8-ми битах (микроконтроллер байтовый), то пишут о 128 адресуемых битах. Когда в счётчике все биты (семь) равны единице (111111^1_2), то следующее прибавление единицы сбрасывает значение счётчика в 0 (000000^0_2) с потерей единицы переноса со старшего разряда, то есть 128 байт адресует счётчик самостоятельно. Такое адресуемое самим счётчиком пространство называется банком памяти. Разработчики отвели дополнительно два старших разряда 7 и 8 биты счётчика для адресации уже банков памяти. Эти разряды принудительно берутся из специального регистра STATUS (5-й, 6-й разряды). Для упрощения написания программ им присвоены специальные имена (RP0, RP1). Это позволяет адресовать 4 банка памяти (рисунок 5.1).



RP1: RP0	Банк
00	0
01	1
10	2
11	3

Рисунок 5.1 – Временные диаграммы работы счётчика и дешифратора банка памяти

На рисунке 5.2 представлен пример карты памяти для микроконтроллеров P16F887.

Карта памяти для микроконтроллеров P16F887							
Регистр косвенной адресации	**Адрес 00h**	**Регистр косвенной адресации**	**Адрес 80h**	**Регистр косвенной адресации**	**Адрес 100h**	**Регистр косвенной адресации**	**Адрес 180h**
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD(1)	08h	TRISD(1)	88h		108h		188h
PORTE(1)	09h	TRISE(1)	89h		109h		189h
PCLATCH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Резерв	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Резерв	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h	Регистры общего назначения	115h	Регистры общего назначения	195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h		117h		197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h	16 байт	119h	16 байт	199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch		9Ch		11Ch		19Ch
CCP2CON	1Dh		9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCONO	1Fh	ADCON1	9Fh		11Fh		19Fh
Регистры общего назначения 96 байт	20h	Регистры общего назначения 80байт	A0h	Регистры общего назначения 80 байт	120h	Регистры общего назначения 80 байт	1A0h
			EFh		16Fh		1EFh
		Доступ к 70h-7Fh (16 байт)	F0h	Доступ к 70h-7Fh (16 байт)	170h	Доступ к 70h-7Fh (16 байт)	1F0h
	7Fh		FFh		17Fh		1FFh
Банк 0		**Банк 1**		**Банк 2**		**Банк 3**	

Рисунок 5.2 – Карта памяти для микроконтроллеров P16F887

Информация разрядов RP0 и RP1 путём конкатенации (соединение) объединяется с информацией, содержащейся в младших битах счётчика команд, что позволяет адресовать 4·128 = 512 байт.

5.3 Система команд

Полный список команд приведён в таблице 5.1.

Таблица 5.1 – Описание команд

Мнемоника команды	Описание	Изм. флаги
ADDWF f,d	Сложение W и f	C,DC,Z
ANDWF f,d	Побитное 'И' W и f	Z
CLRF f	Очистить f	Z
CLRW -	Очистить W	Z
COMF f,d	Инвертировать f	Z
DECF f,d	Вычесть 1 из f	Z
DECFSZ f,d	Вычесть 1 из f и пропустить, если 0	
INCF f,d	Прибавить 1 к f	Z
INCFSZ f,d	Прибавить 1 к f и пропустить, если 0	
IORWF f,d	Побитное 'ИЛИ' W и f	Z
MOVF f,d	Переслать f	Z
MOVWF f	Переслать W в f	
NOP -	Нет операции	
RLF f,d	Циклический сдвиг f влево через перенос	C
RRF f,d	Циклический сдвиг f вправо через перенос	C
SUBWF f,d	Вычесть W из f	C,DC,Z
SWAPF f,d	Поменять местами полубайты в регистре f	
XORWF f,d	Побитное 'исключающее ИЛИ' W и f	Z
BCF f,b	Очистить бит b в регистре f	
BSF f,b	Установить бит b в регистре f	
BTFSC f,b	Проверить бит b в регистре f, пропустить, если 0	
BTFSS f,b	Проверить бит b в регистре f, пропустить, если 1	
ADDLW k	Сложить константу с W	C,DC,Z
ANDLW k	Побитное 'И' константы и W	Z
CALL k	Вызов подпрограммы	
CLRWDT -	Очистить WDT	-TO,-PD
GOTO k	Безусловный переход	
IORLW k	Побитное 'ИЛИ' константы и W	Z
MOVLW k	Переслать константу в W	
RETFIE -	Возврат из подпрограммы с разрешением прерываний	
RETLW k	Возврат из подпрограммы с загрузкой константы в W	
SLEEP -	Перейти в режим SLEEP	-TO,-PD
SUBLW k	Вычесть W из константы	C,DC,Z
XORLW k	Побитное 'исключающее ИЛИ' константы и W	Z

Система команд аккумуляторного типа (одноадресная), оригинальна и разделена на три основные группы:

1) байт ориентированные команды;
2) бит ориентированные команды;
3) команды управления и операций с константами.

Для байт ориентированных команд *f* является указателем регистра, а *d* – указателем адресата результата. Указатель регистра определяет, какой регистр должен использоваться в команде. Указатель адресата определяет, где будет сохранен результат. Если *d* = 0, результат сохраняется в регистре *W*. Если *d* = 1, результат сохраняется в регистре, который используется в команде. В бит ориентированных командах *b* определяет номер бита, участвующего в операции, а *f* – указатель регистра, который содержит этот бит. Во всех примерах используется следующий формат шестнадцатеричных чисел: 0x*hh*, где *h* – шестнадцатеричная цифра.

5.4 Программный симулятор MPLAB-SIM

Симулятор MPLAB-SIM позволяет проследить выполнение программы микроконтроллеров PICmicro на уровне команд по шагам или в режиме анимации. Под этим термином подразумевается, что в библиотечных файлах есть программа, имитирующая работу микроконтроллера на выполнение написанной программы с некоторыми сервисными функциями (в основе лежит «виртуальный» микроконтроллер). На любой команде выполнение программы может быть остановлено для проверки и изменения памяти. MPLAB-SIM полностью поддерживает символьную отладку, используя MPLAB-C17, MPLAB-C18 и MPASM. MPLAB-SIM является доступным и удобным средством отладки программ микроконтроллеров PICmicro.

Порядок выполнения работы

1 Произвести очистку «памяти» микроконтроллера, ввести начальную программу в память микроконтроллера, запустить PIC-программатор в работу. Наблюдать свечение нулевого светодиода.

2 Видоизменить начальную программу с учётом номеров зажигания светодиодов (номера уточнить у преподавателя). Нумерация светодиодов справа налево, нумерация начинается с 0. Прогнать программу.

3 Изменить интервал мигания светодиодов в соответствии с заданием преподавателя. Увеличить интервал свечения по отношению к исходной программе. Зафиксировать время свечения *t*1.

Контрольные вопросы

1 Назначение программатора.
2 Среда программирования, возможные источники ее получения.
3 Процедура установки (ввода) среды в компьютер.
4 Директивы и инструкции (команды) в программе.
5 Банки памяти. Их объём, адреса.

6 Двоичные, десятичные и шестнадцатеричные числа. Алгоритм перевода.
7 Ввод программы с диска в память компьютера.
8 Отладка программы. Эмуляция и симуляция.

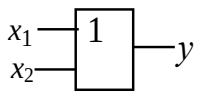
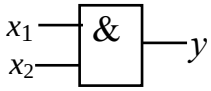
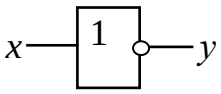
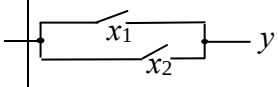
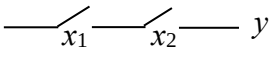
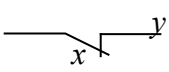
6 Практическое занятие № 6. Исследование логических команд

Цель практической работы – изучение принципа функционирования и характеристик логических элементов и триггеров.

6.1 Общие сведения

В ЭВМ, импульсных и других цифровых устройствах широко применяются *логические элементы*. Каждый логический элемент выполняет вполне определенную логическую операцию. Основными логическими операциями являются: логическое отрицание НЕ (инверсия), логическое сложение ИЛИ (дизъюнкция), логическое умножение И (конъюнкция) (таблица 6.1). К базовым логическим элементам относятся элементы Пирса и Шеффера (таблица 6.2).

Таблица 6.1 – Формы отображения основных логических функций

Наименование функции	Дизъюнкция	Конъюнкция	Инверсия
Символическая	$\vee$ или +	$\wedge$ или ·	$\bar{x}$
Буквенная	ИЛИ	И	НЕ
Условная графическая			
Аналитическая	$y=x_1 \vee x_2 = x_1 + x_2$	$y = x_1 \wedge x_2 = x_1 x_2$	$y = \bar{x}$
Табличная (истинности)	x_1 x_2 y 0 0 0 0 1 1 1 0 1 1 1 1	x_1 x_2 y 0 0 0 0 1 0 1 0 0 1 1 1	x y 0 1 1 0
Контактная			

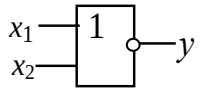
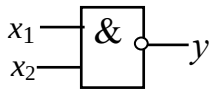
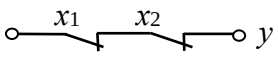
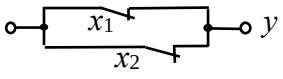
На основе этих простых операций могут строиться и более сложные. Для описания логических операций используется алгебра логики. Алгебра логики широко применяется в теории цифровой техники, в которой используются устройства, имеющие два устойчивых состояния равновесия. При этом одно из состояний, соответствующее, например, высокому уровню напряжения, обозначается *единицей*, а соответствующее низкому уровню напряжения – *нулем*. Уровень выходного напряжения логического элемента зависит от уровня

входного (или нескольких входных) напряжения. Эта связь отображается таблицей состояний (таблицей истинности).

Триггер – это устройство последовательностного типа с двумя устойчивыми состояниями равновесия, предназначенное для записи и хранения информации. Под действием входных сигналов триггер может переключаться из одного устойчивого состояния в другое. При этом напряжение на его выходе скачкообразно изменяется с низкого уровня на высокий или наоборот.

По способу записи информации триггеры делят на *асинхронные*, которые переключаются в момент подачи входного сигнала, и *синхронные* (тактируемые), которые переключаются только при подаче синхронизирующих импульсов, а момент переключения связан с определённым уровнем синхросигнала (*статические* триггеры) или с моментом перепада напряжения на тактируемом входе (*динамические* триггеры).

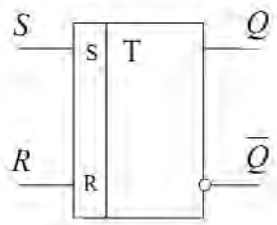
Таблица 6.2 – Формы отображения базовых логических функций

Наименование функции	Функция Пирса	Функция Шеффера
Символическая	$\downarrow$	$\mid$
Буквенная	ИЛИ-НЕ	И-НЕ
Условная графическая		
Аналитическая	$y=x_1 \downarrow x_2$	$y=x_1 \mid x_2$
Табличная (истинности)	x_1 x_2 y 0 0 1 0 1 0 1 0 0 1 1 0	x_1 x_2 y 0 0 1 0 1 1 1 0 1 1 1 0
Контактная		

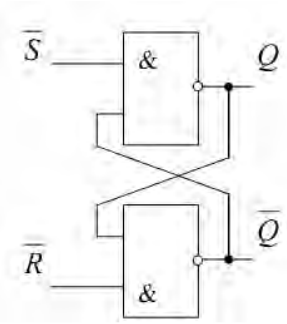
Как правило, триггер имеет два выхода: прямой Q и инверсный $\overline{Q}$. Число входов зависит от структуры и функций, выполняемых триггером. Например, асинхронные *RS*-триггеры имеют два входа: вход S установки в *единичное* состояние прямого выхода Q и вход R установки *в нулевое* состояние выхода Q. Синхронные триггеры для занесения в них информации, помимо информационных входов S (J) и R (K), имеют синхронизирующий C или счётный T вход, а триггеры задержки – информационный вход D.

Наибольшее распространение в цифровых устройствах получили триггеры *RS* (рисунки 6.1 и 6.2), *D* (рисунок 6.3), *JK* (рисунок 6.4) и *T* (рисунок 6.5).

а)



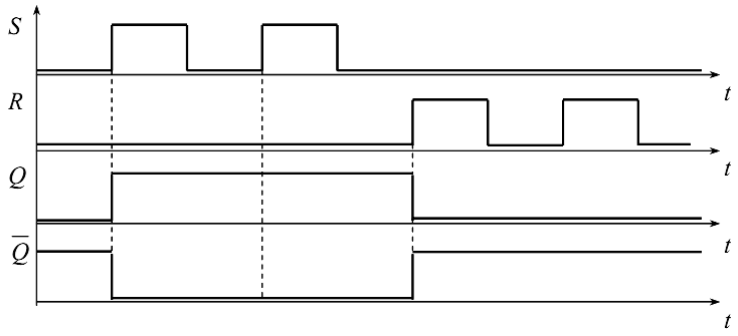
б)



в)

R	S	Q^{n+1}
0	0	Q^n
0	1	1
1	0	0
1	1	×

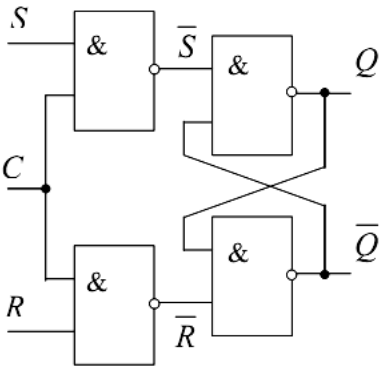
г)



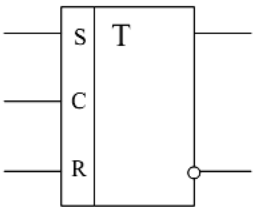
а – условное графическое изображение; *б* – схема реализации *RS*-триггера на базовых элементах И-НЕ; *в* – таблица истинности; *г* – временная диаграмма, иллюстрирующая работу триггера

Рисунок 6.1 – Асинхронный *RS*-триггер

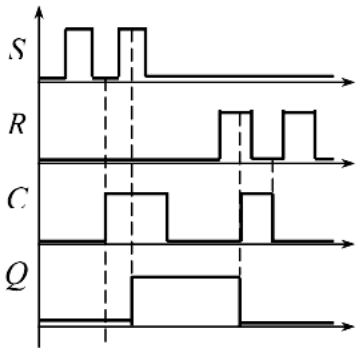
а)



б)



в)



а – схема реализации *RS*-триггера на базовых элементах И-НЕ; *б* – условно-графическое; *в* – временная диаграмма, иллюстрирующая работу триггера

Рисунок 6.2 – Синхронный *RS*-триггер

RS-триггер имеет два управляющих входа: S (set) и R (reset), с помощью которых выполняются установки триггера в то или иное состояние: $Q = 1$ при $S = 1$ и $R = 0$ (установка триггера); $Q = 0$ при $S = 0$ и $R = 1$ (сброс триггера); $Q^{n+1} = Q^n$ при $S = R = 0$ (режим хранения предыдущего состояния); $S = R = 1$ – запрещенная комбинация управляющих сигналов, которая может привести к неопределенному состоянию триггера.

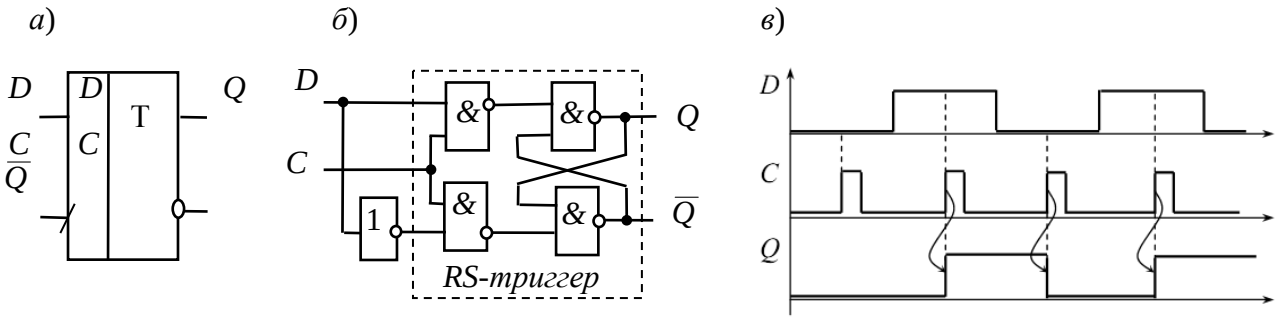
Рассматриваемый триггер является асинхронным, т. к. изменение его состояния происходит непосредственно с поступлением управляющих сигналов. Принцип работы асинхронного *RS*-триггера поясняется временными диаграммами, показанными на рисунке 6.1, *г*.

Схемотехнически *RS*-триггер может быть реализован на элементах 2ИЛИ-НЕ (см. рисунок 6.1, *б*) и 2И-НЕ с использованием перекрестных положительных обратных связей. В триггере на элементах 2И-НЕ изменение состояния происходит при низких уровнях сигналов *S* и *R*.

В синхронных *RS*-триггерах могут быть использованы различные способы синхронизации. На рисунке 6.2, *а* и *б* показаны схемотехническая реализация и условное обозначение *RS*-триггера с синхронизацией по уровню (высокому). На рисунке 6.2, *в* приведены диаграммы работы такого триггера. Изменение состояний происходит только при высоких уровнях сигнала синхронизации *C*.

В *RS*-триггере с синхронизацией по фронту изменение состояния происходит в момент изменения уровня сигнала *C*. При этом возможна синхронизация как по переднему, так и по заднему фронту (срезу). Такие триггеры строятся по двухступенчатой схеме и в них процессы приема и записи данных разделены во времени.

Отличительной особенностью *D*-триггера (триггера задержки) является то, что он сохраняет информацию, поступившую на *D*-вход в предыдущем такте работы до прихода синхроимпульса, т. е. его состояние может изменяться с задержкой на один такт. Синхронизация работы производится по переднему или заднему фронту. Условное обозначение *D*-триггера с синхронизацией по переднему фронту и диаграммы его работы показаны на рисунке 6.3.

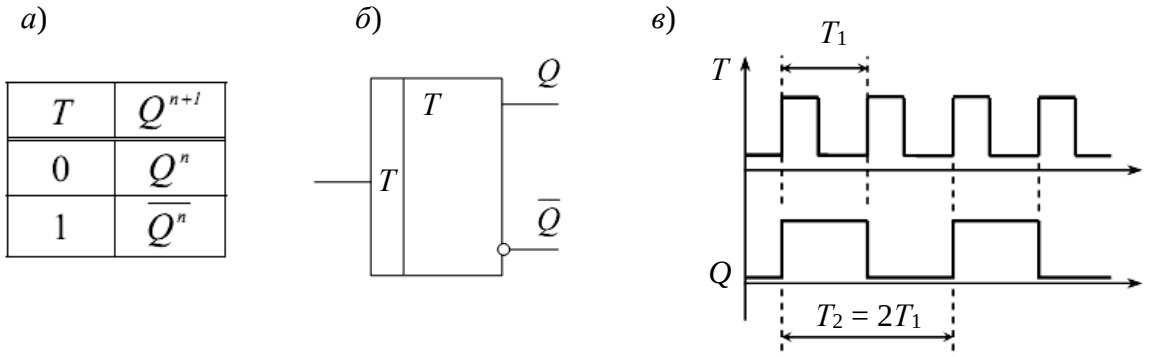


а – условное графическое изображение; *б* – схема реализации *D*-триггера на базовых элементах И-НЕ; *в* – временная диаграмма, иллюстрирующая работу триггера

Рисунок 6.3 – *D*-триггер

T-триггер иначе называется счетным и применяется для построения счетчиков и делителей частоты. Такой триггер имеет один тактовый вход и его состояние меняется каждый раз при подаче счетного импульса $T = 1$ и остается неизменным при $T = 0$. Таблица состояния триггера, его обозначение и диаграммы работы приведены на рисунке 6.4.

Как видно из диаграмм, *T*-триггер делит частоту входных импульсов в 2 раза. Для получения больших значений коэффициента деления частоты применяется каскадное соединение *T*-триггеров. Как самостоятельное изделие *T*-триггер в виде интегральной микросхемы не выпускается и при необходимости реализуется на базе других типов триггеров.



а – таблица истинности; *б* – условное графическое изображение; *в* – временная диаграмма, иллюстрирующая работу триггера

Рисунок 6.4 – *T*-триггер

JK-триггер имеет два управляющих входа: *J* (jump) и *K* (keep) и функционирует подобно *RS*-триггеру, но при этом не имеет запрещенных комбинаций управляющих сигналов. *J*-вход подобен *S*-входу, а *K*-вход подобен *R*-входу. При всех комбинациях сигналов на входе, кроме $J = K = 1$, он действует подобно *RS*-триггеру. При $J = K = 1$ в каждом такте происходит «опрокидывание» триггера и его состояние меняется на противоположное.

JK-триггеры относятся к универсальным устройствам в отношении их применения как для построения других типов триггеров, так и более сложных устройств последовательного принципа действия. Во всех сериях цифровых интегральных микросхем выпускаются *JK*-триггеры с различными функциональными возможностями.

Порядок выполнения работы

1 В программе Multisim собрать схему для испытания основных и базовых логических элементов (рисунок 6.5) OR (ИЛИ), AND (И), NOT (НЕ), NAND (И-НЕ) и XOR (ИЛИ-НЕ), расположенных в библиотеке Misc Digital/TIL с уровнем высокого напряжения 5 В. В схему включены ключи SB1 и SB2, пробники X1, X2 и Y1…Y5 с пороговыми напряжениями 5 В. Если входной или выходной сигнал элемента равен логической единице, то включенный на выходе этого элемента пробник светится.

Результаты моделирования занести в таблицу 6.3.

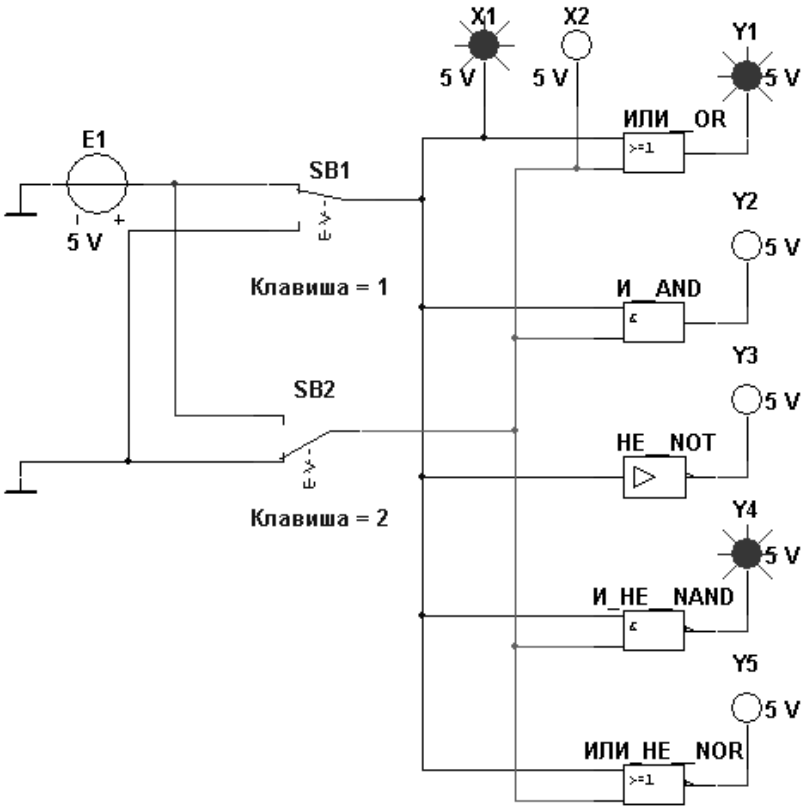


Рисунок 6.5 – Схема для исследования основных и базовых логических элементов

Таблица 6.3 – Результаты моделирования

Дизъюнктор [ИЛИ (**OR**)]			Конъюнктор [И (**AND**)]			Инвертор [НЕ (**NOT**)]		Штрих Шеффера [И-НЕ (**NAND**)]			Стрелка Пирса [ИЛИ-НЕ (**NOR**)]		
x_1	x_2	y	x_1	x_2	y	x	y	x_1	x_2	y	x_1	x_2	y
0	0		0	0		0		0	0		0	0	
0	1		0	1				0	1		0	1	
1	0		1	0		1		1	0		1	0	
1	1		1	1				1	1		1	1	

2 По заданию преподавателя исследовать работу 3-х логических элементов одной из серий интегральных микросхем (рисунок 6.6), предварительно выписав из справочника их параметры и условное обозначение. Составить таблицу истинности для данных элементов.

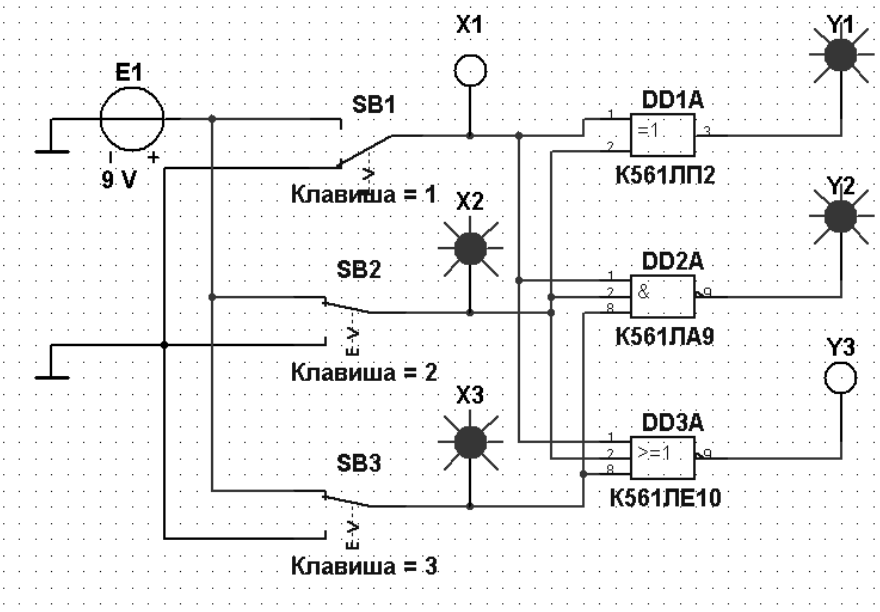


Рисунок 6.6 – Схема для исследования логических элементов серии К561

Контрольные вопросы

1 Логические элементы.
2 Конъюнкция, дизъюнкция, инверсия.
3 Понятие «триггер».
4 Виды триггеров.

7 Практическое занятие № 7. Разработка и исследование программы управления цикловым механизмом

Цель практической работы – ознакомление с особенностями программирования и управления цикловым механизмом на примере траспортного робота.

7.1 Общие положения

В автоматических транспортных системах гибкого автоматизированного производства для осуществления транспортных операций используются робокары. По принципу построения, т. е. по виду кинематической схемы, их можно разделить на несколько групп:

1) классическая четырехколесная с рулевым приводом;

2) трехколесная, в которой переднее колесо является ведущим и рулевым;

3) одноосная, в которой нет рулевого привода, а управление направлением движения осуществляется за счет разности скоростей движения ведущих колес, расположенных по бортам [9].

Последняя схема имеет существенные преимущества по сравнению с другими, обеспечивая симметрию движения вперед и назад и минимальные радиусы поворота, вплоть до разворота на месте. Кинематическая схема такого транспортного робота изображена на рисунке 7.1.

Ведущие колеса *1* и *2* с индивидуальными силовыми приводами *СП*1 и *СП*2 расположены на оси симметрии по бортам. По углам установлены четыре опорных (флюгерных) колеса. Впереди и сзади на продольной оси симметрии расположены датчики трассы *Д*1 и *Д*2, выдающие сигнал, пропорциональный величине отклонения датчика от трассы.

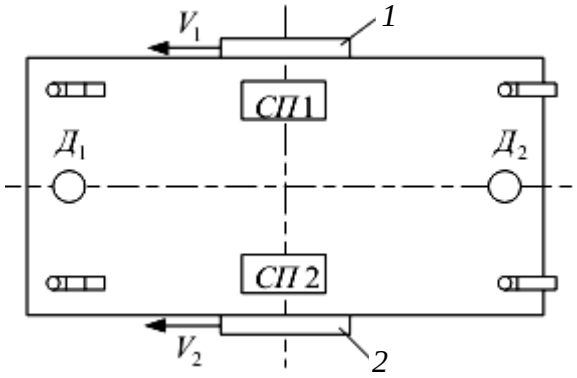


Рисунок 7.1 – Кинематическая схема транспортного робота

Управление движением осуществляется бортовой системой программного управления. Верхний, программный, уровень осуществляет:

1) управление силовыми приводами (направление и скорость движения);

2) выбор направления движения на разветвлениях трассы;

3) управление приводом погрузочного-разгрузочного устройства (рольганг, сталкиватель);

4) фиксацию груза во время движения;

5) управление приводами тормозов.

Нижний уровень – следящая система – обеспечивает слежение за трассой путем управления силовыми приводами по сигналу датчика трассы.

Скорость движения задается программно величиной задания на привода $U_з$. Слежение за трассой осуществляется по сигналу датчика $U_{датч}$, который суммируется с сигналом задания, обеспечивая отрицательную обратную связь. Скорости вращения колес, пропорциональные сигналам задания, будут изменяться по сигналу датчика, стремясь удержать датчик над трассой.

Порядок выполнения работы

1 Изучить кинематическую схему транспортного робота, представленную в индивидуальном задании.

2 На основе анализа кинематики разработать математическую модель транспортного робота, связывающую отклонение датчика от трассы с разностью скоростей вращения колес, которая является управляющим сигналом. При разработке модели считать, что колеса жесткие и их проскальзывание во время движения отсутствует. Входами модели будут являться разность скоростей колес Δ_V и скорость поворота трассы ω_{TP}, а выходом – отклонение датчика от трассы.

Контрольные вопросы

1 Основные принципы автоматического управления цикловым механизмом.

2 Параметры математической модели управления движением транспортным роботом.

Список литературы

1 **Марченко, А. Л.** Лабораторный практикум по электротехнике и электронике в среде Multisim: учебное пособие для вузов / А. Л. Марченко, С. В. Освальд – Москва : ДМК Пресс, 2010. – 448 с.

2 **Лачин, В. И.** Электроника : учебное пособие / В. И. Лачин, Н. С. Савелов. – 7-е изд., перераб. и доп. – Ростов на Дону : Феникс, 2009. – 703 с.

3 **Панфилов Д. И.** Электротехника и электроника в экспериментах и упражнениях: Лаборатория на компьютере: в 2 т. / Под общ. ред. Д. И. Панфилова. – Москва : Изд-во МЭИ, 2004. – Т. 1. – 304 с.

4 **Ткаченко, Ф. А.** Техническая электроника / Ф. А. Ткаченко. – Минск: Дизайн ПРО, 2002. – 368 с.

5 **Батоврин, В. К.** LabVIEW: практикум по электронике и микропроцессорной технике: учебное пособие для вузов / В. К. Батоврин, А. С. Бессонов, В. В. Мошкин. – Москва : ДМК Пресс, 2005. – 182 с.

6 **Перельман, Б. Л.** Отечественные микросхемы и зарубежные аналоги: справочник / Б. Л. Перельман, В. В. Шевелев. – Москва : Микротех, 2000. – 375 с.

7 **Мортон, Дж.** Микроконтроллеры AVR. Вводный курс: пер. с англ. / Дж. Мортон. – Москва : Изд. дом «Додека – XXI», 2006. – 272 с.

8 Электроника и микропроцессорная техника. Электротехника и электроника: методические рекомендации / Сост.: С. В. Болотов, Ф. М. Трухачёв, И. А. Черкасова. – Могилев : БРУ, 2011. – Ч. 2. – 34 с.

9 **Красовский, А. Я.** Локальные системы автоматики / А. Я. Красовский. – Минск : БГУиР, 2008. – 173 с.