МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

АППАРАТНОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И СЕТЕЙ

Методические рекомендации к лабораторным работам для студентов специальности 1-53 01 02 «Автоматизированные системы обработки информации» очной и заочной форм обучения

Часть 1

Могилев 2019

УДК 004.412
ББК 32.973.26-018.2
А76

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Автоматизированные системы управления» «12» ноября 2019 г., протокол № 4

Составители: доц. А. И. Якимов;
ст. преподаватель В. Т. Садовский

Рецензент ст. преподаватель Ю. С. Романович

Методические рекомендации к лабораторным работам предназначены для студентов специальности 1-53 01 02 «Автоматизированные системы обработки информации» дневной и заочной форм обучения.

Учебно-методическое издание

АППАРАТНОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И СЕТЕЙ

Часть 1

Ответственный за выпуск А. И. Якимов
Редактор А. А. Подошевко
Компьютерная верстка Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 31 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 07.03.2019.
Пр-т Мира, 43, 212022, Могилев.

Содержание

Введение

Актуальность дисциплины «Аппаратно-программное обеспечение ЭВМ и сетей» обусловлена бурным развитием вычислительной техники, сетевых и телекоммуникационных технологий (в том числе «облачных»), появлением новых версий операционных систем, средств виртуализации, а также новых веб-услуг.

Сетевые технологии являются неотъемлемой частью почти любой современной информационной системы. При этом с одной стороны, сетевые средства органически входят в состав операционных систем современных ЭВМ, с другой – коммуникационные средства сетевых технологий, составляющие основу локальных и глобальных сетей, построены на микропроцессорной технике и специализированных ЭВМ.

Целью изучения учебной дисциплины «Аппаратно-программное обеспечение ЭВМ и сетей» является подготовка студентов в области аппаратно-программных средств электронных вычислительных машин (ЭВМ) и современных сетевых технологий.

В связи с этим подготовка студентов по специальности 1-53 01 02 «Автоматизированные системы обработки информации» в области аппаратно-программных средств электронных вычислительных машин и современных сетевых технологий является актуальной.

Задачи учебной дисциплины:

– изучение основных принципов функционирования, построения и программирования микропроцессоров и микроконтроллеров;

– изучение аппаратных и программных средств ЭВМ и сетей;

– приобретение фундаментальных знаний в области современных сетевых технологий;

– приобретение навыков практической работы в области современных сетевых технологий, включая их разработку, эксплуатацию и администрирование.

При выполнении лабораторных работ каждый студент составляет отчет.

Содержание отчета

1 Титульный лист.
2 Цели и задачи лабораторной работы.
3 Перечень использованного оборудования и программного обеспечения.
4 Постановка задачи.
5 Схема топологии, алгоритм, исходный код задачи лабораторной работы.
6 Методика выполнения работы.
7 Анализ результатов и выводы по работе.

Отчет оформляется в письменном виде вместе с рисунками Screen Shot’s либо оформляется с помощью графического редактора.

1 Лабораторная работа № 1. Машинный язык Ассемблер (MASM)

Цель работы: ознакомиться с возможностями машинно-ориентированного языка программирования аппаратных средств низкого уровня – языка Ассемблер.

Основные теоретические положения

Язык Ассемблер – это низкоуровневый язык программирования для компьютеров или других программируемых устройств, он разработан для максимального использования конкретной специфики компьютера, микропроцессорных систем. Для программы на Ассемблере используют MASM.

Регистры.

Операции процессора в основном связаны с обработкой данных. Эти данные могут быть сохранены в памяти и доступны оттуда, но для обработки данных применяются регистры процессора (рисунок 1.1), что позволяет ускорить вычислительный процесс.

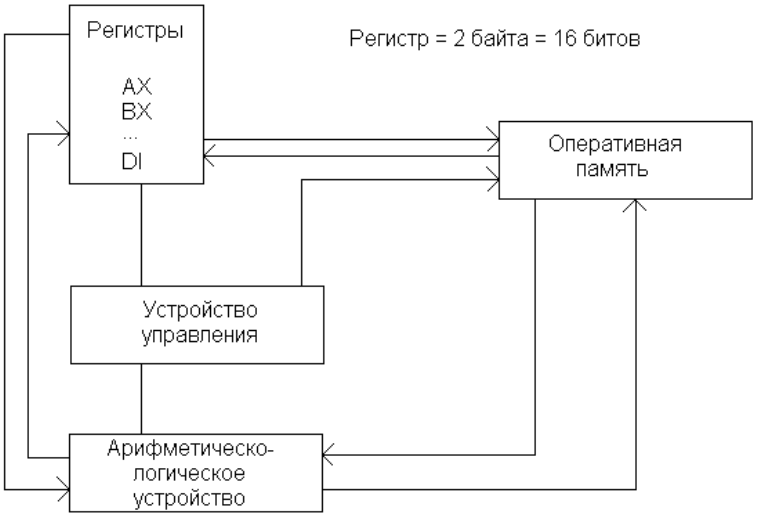


Рисунок 1.1 – Обработка данных с помощью оперативной памяти и регистров

Внутренние регистры.

Микропроцессор IBM PC содержит три группы регистров, данных и адресов, 16-битовый указатель команд IP (Instruction Pointer) и 16-битовый регистр флагов.

Регистры работы с данными.

К первой группе относятся четыре регистра общего назначения: AX, BX, CX, DX. Их можно рассматривать как четыре 16-битовых или восемь 8-битовых регистров. Эти регистры образованы из 8-битовых регистров: регистр AX состоит из двух байтовых регистров AL и AH, регистр BX из BL, BH, CX – из CL, CH, и DX из DL и DH. Здесь L и H означают младшие (Low-order) и стар-

шие (High-order) байты 16-битовых регистров. Регистр AX, аккумулятор (accumulator), используется при умножении и делении слов, в операциях ввода-вывода и в некоторых операциях над строками; регистр BX, базовый регистр (base register), часто используется при адресации данных в памяти; регистр CX, счетчик (count register), используется как счетчик. Регистр CL используется как счетчик при операциях сдвига и циклического сдвига на несколько битов; регистр DX, регистр данных (data register), используется при умножении и делении слов.

Регистры сегментов.

Оперативная память разделена на сегменты. Начальные адреса этих сегментов содержатся в четырех регистрах сегментов. Регистр сегмента команд CS (code segment) указывает на сегмент, содержащий текущую исполняемую программу; регистр сегмента стека SS (stack segment) указывает на текущий сегмент стека; регистр сегмента данных DS (data segment) указывает на текущий сегмент данных.

Регистры указателей и индексов.

Для вычисления адреса команды в сегменте команд процессор извлекает номер сегмента памяти из регистра CS, а смещение – из регистра IP. Подобным образом осуществляется доступ к данным других сегментов. Для доступа к сегменту данных извлекается номер сегмента из регистра DS, а смещение – из регистра BX. Для доступа к сегменту стека процессор извлекает номер сегмента из регистра SS, а смещение – из регистра указателя (SP или BP).

Флаги.

Очень важным является регистр, в котором содержатся признаки последней выполненной операции, именуемый флагами. В 16-битовом регистре флагов размещены девять флагов, а именно:

1) флаг переноса CF (carry flag) (Бит 0) изменяется во многих операциях;
2) флаг четности PF (parity flag) (Бит 2), PF = 1;
3) флаг вспомогательного переноса AF (auxiliary carry flag) (Бит 4);
4) флаг нуля ZF (zero flag) (Бит 6), ZF = 1, если результат операции равен 0;
5) флаг знака SF (sign flag) (Бит 7), SF = 1, если результат отрицательный;
6) флаг трассировки TF (trap flag) (Бит 8);
7) флаг прерывания IF = 0 (interrupt enable flag), прерывания запрещены;
8) флаг направления DF (direction flag) (Бит 10);
9) флаг переполнения OF (overflow flag) (Бит 11).

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Выполнить типовое задание.

3 Сделать выводы по результатам исследований.

4 Оформить отчет.

Контрольные вопросы

1 Что такое оперативная память?
2 Какие регистры имеются в IBM PC?
3 Назовите регистры для работы с данными.
4 Назовите регистры сегментов. Для чего они нужны?
5 Какую роль выполняют регистры указателей и индексов?
6 Какова длина и назначение регистров?
7 Что находится в регистре сегмента?
8 Что такое сегмент?
9 Сколько флагов содержится в регистре флагов?
10 Общее назначение регистра флагов.
11 Назовите флаги регистра и их функции.

2 Лабораторная работа № 2. Ассемблер. Обработка данных и вывод на экран

Цель работы: изучить способы ввода по прерыванию INT 21H и 16H; научиться писать фрагменты программ ввода-вывода.

Основные теоретические положения

Режимы ввода/вывода определяются значением в регистре AH.

Операции ввода/вывода по прерыванию INT 21H

AH = 01: Данная операция считывает с консоли символ и помещает его в регистр AL. После считывания содержимое данного регистра представляет собой ASCII-символ. Если после считывания символа регистр AL будет равен 0, то с клавиатуры были нажаты специальные клавиши (F1, PgUp и т. д). Для определения их кода необходимо повторить запрос.

AH = 02: Вывод символа. Для ввода символа на экран в текущую позицию курсора необходимо поместить код данного символа в регистр DL.

AH = 06: Данный режим может использоваться как для ввода, так и для вывода. Для вывода в консоль символа необходимо поместить его в DL и вызвать прерывание INT 21H, для ввода – занести в DL FFh и вызвать прерывание INT 21H. При вводе символ поместится в регистр AL.

AH = 07: Прямой ввод с клавиатуры без эхо-отображения. Введенный символ не отображается на экране и отсутствует реакция на запрос Ctrl Break.

AH = 08: Ввод с клавиатуры без эхо-отображения. Данная функция действует аналогично функции 01, но нет эха.

AH = 09: Вывод строки символов. Выводимая строка должна заканчиваться знаком доллара $. Адрес начала строки должен быть помещен в DX. Знак доллара не выводится.

AH = 0AH: Ввод значений в буфер. При этом режиме работы данные помещаются в буфер, где определена максимальная длина введенной строки. Если при вводе строка оказывается меньше длины буфера, то окончание ввода строки определяется символом клавиши Enter, если больше длины буфера, тогда пользователь предупреждается звуковым сигналом.

Например:

```
String DB "Введите символ$     Строка для вывода;
Mov dx, offset string          Смещение строки в регистр DX;
Mov ah, 9                      Выполняем функцию вывода строки;
Int 21h                        Прерываемся для вывода строки;
```

Далее приведен пример, в котором определен список параметров для области ввода. Первый байт содержит максимальную длину вводимых данных. Так как это однобайтовое поле, то возможное его максимальное значение – шестнадцатеричное FF или 255. Второй байт необходим DOS для занесения в него действительного числа введенных символов. Третьим байтом начинается поле, которое будет содержать введенные символы.

```
MAXLEN DB 20                   максимальная длина;
ACTLEN DB ?                    реальная длина;
NAMEFLD DB 20 DUP ('')         введенные символы;
```

Для запроса на ввод необходимо поместить в регистр AH номер функции – 10 (0Ah), загрузить адрес списка параметров (MAXLEN в данном примере) в регистр DX и выполнить INT 21H:

```
MOV AH, 0Ah                    запрос функции ввода;
LEA DX MAXLEN                  загрузить адрес буфера в DX;
INT 21h                        вызвать системную программу DOS;
```

Команда INT ожидает, пока пользователь введет с клавиатуры текст, проверяя при этом, чтобы число введенных символов не превышало максимального значения, указанного в списке параметров (20 в данном примере). Для указания конца ввода пользователь нажимает клавишу Enter. Код этой клавиши (0D) также заносится в поле ввода (NAMEFLD в данном примере). Если, например, пользователь ввел имя BROWN (Return), то список параметров будет содержать информацию:

```
Десятичные и символьные:      20 5 B R O W N # ;
Шестнадцатеричные             14 05 42 52 4F 57 4E OD 2.
```

Во второй байт списка параметров (ACTLEN в данном примере) команда заносит длину введенного имени – 05. Код Return находится по адресу NAMEFLD-5.

AH = 0Bh: Проверка состояния клавиатуры. Данная функция возвращает шестнадцатеричное значение FF в регистре AL. Если ввод с клавиатуры возмо-

жен (в противном случае – 00), это средство связано с функциями 01, 07, 08, которые не ожидают ввода с клавиатуры.

Ввод с клавиатуры по команде BIOS INT 16H

Команда BIOS **INT 16H** выполняет специальную операцию, которая в соответствии с кодом в регистре **AH** обеспечивает следующие три функции ввода с клавиатуры.

AH = 00: Чтение символа. Данная функция помещает в регистр AL очередной ASCII-символ, введенный с клавиатуры, и устанавливает скэн-код в регистре AH.

AH = 01: Определение наличия введенного символа. Данная функция сбрасывает флаг нуля (ZF = 0), если имеется символ для чтения с клавиатуры; очередной символ и скэн-код будут помещены в регистры AL и AH соответственно, и данный элемент останется в буфере.

AH = 02: Определение текущего состояния клавиатуры. Данная функция возвращает в регистре AL состояние клавиатуры из адреса памяти 417H:

Бит:

7 – Состояние вставки активно (Ins);

6 – Состояние фиксации верхнего регистра (Caps Lock) включено;

5 – Состояние фиксации цифровой клавиатуры (Num Lock) включено;

4 – Состояние фиксации прокрутки (Scroll Lock) включено;

3 – Нажата комбинация клавиш Alt/Shft;

2 – Нажата комбинация клавиш Ctrl/Shft;

1 – Нажата левая клавиша Shift;

0 – Нажата правая клавиша Shift.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Написать программу на Ассемблере в соответствии с заданием.

3 Выполнить компиляцию с помощью MASM или TASM.

4 Распечатать листинг программы.

5 Распечатать результаты работы программы.

6 Сделать выводы по результатам исследований.

7 Подготовить отчет по лабораторной работе.

Варианты заданий.

1 В цикле ввести символ с клавиатуры и вывести его двоичное представление на экран. Если введен символ «*», закончить работу программы.

2 В цикле ввести десятичное число с клавиатуры (Функция **AH = 2 INT 21H**). Число десятичных разрядов – от 1 до 5. Признак конца ввода – нажатие клавиши Enter (код 13). Преобразовать число в двоичное и вывести его двоичное представление на экран.

3 В цикле ввести десятичное число с клавиатуры (Функция **AH = 0 AH INT 21H**). Число десятичных разрядов – от 1 до 5. Признак конца ввода – нажатие клавиши Enter (код 13). Преобразовать число в двоичное и вывести его двоичное представление на экран.

4 Вывести в текстовом режиме прямоугольную рамку на экран. Координаты левого верхнего и правого нижнего углов (0–24 и 0–79) ввести с клавиатуры. Символы для вывода рамки – коды ASCII.

5 Вывести 8 битов с клавиатуры – последовательность 0 и 1. Вывести на экран преобразованную последовательность в виде символа ASCII в заданной позиции экрана, которая вводится с клавиатуры (строка, столбец).

Контрольные вопросы

1 Что такое прерывание?
2 Что делает команда INT 21H, INT 16H?
3 Как задается функция для выполнения прерывания?
4 Куда вводятся символы при нажатии клавиши?
5 Что надо выполнить для вывода символа?

3 Лабораторная работа № 3. Управление ресурсами МПС

Цель работы: ознакомиться с архитектурой микропроцессорных систем (МПС), функциональными узлами, ресурсами МПС, на базе микропроцессоров 8051/8052, программными средствами их управления.

Основные теоретические положения

Микроконтроллер 8051 – это 8-битное семейство микроконтроллеров, разработанное Intel в 1981 г. Это одно из популярных семейств микроконтроллеров, которые используются во всем мире. Микроконтроллер 8051 (MSC-51) имеет 128 байт оперативной памяти, 4 Кбайт ПЗУ, два таймера, один последовательный порт и четыре порта на одном кристалле. Процессор может обрабатывать до 8 бит данных одновременно. Если данные больше 8 бит, то они должны быть разбиты на части, чтобы процессор мог легко их обрабатывать, объем ПЗУ может быть расширен до 64 Кбайт (рисунок 3.1).

В микропроцессоре (микроконтроллере) 8051 (MSC-51) все вычисления выполняются в арифметико-логическом устройстве, являющемся частью базового процессорного модуля (CPU). Обмен данными, находящимися в оперативной памяти микроконтроллера, а также считывание команд выполняется по внутренней шине 8051. По этой шине осуществляется и обмен данными с портами ввода/вывода P1…P3, с последовательным портом и таймерами. Внутренний контроллер шины формирует необходимые сигналы (**EA, ALE, PSEN, RD/WR**) для работы с внешней памятью программ и данных, а также сигнал сброса/начальной установки **RST**.

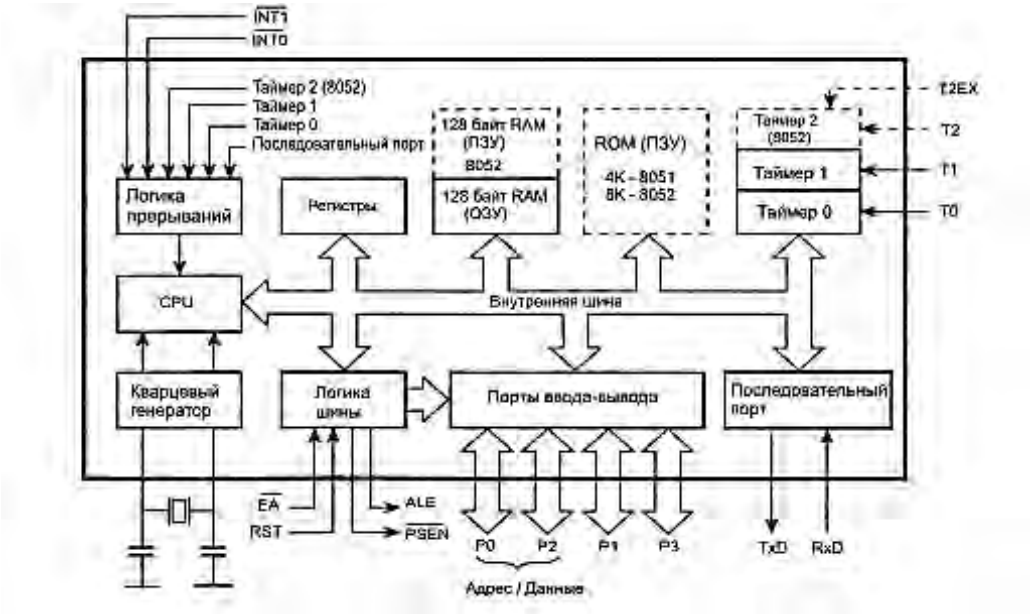


Рисунок 3.1 – Аппаратная архитектура микропроцессора 8051 (MSC-51)

Микроконтроллеры 8051 рассчитаны на работу с системами реального времени, которые могут генерировать определенные сигналы, требующие немедленной реакции микроконтроллера. Для обработки таких сигналов (или событий) служит аппаратно-реализованная логика прерываний. Скорость выполнения операций в системе на базе 8051 зависит от тактовой частоты, с которой работает кристалл и которая может варьироваться от единиц до нескольких десятков мегагерц. В архитектуру классического микроконтроллера 8051 были внесены некоторые изменения (к двум существующим таймерам добавлен третий, а также расширена внутренняя память), которые привели к созданию устройства 8052, наиболее популярного в настоящее время. Микроконтроллер 8051 реализован в виде однокристального устройства с внешними выводами.

К ресурсам микропроцессора можно отнести внутреннюю шину данных, регистры, таймеры, набор модулей памяти ОЗУ, ПЗУ, дополнительную память Flash, EPROM или ROM-CODE, порты ввода/вывода (ЦАП, АЦП), последовательный порт UART.

В качестве примера рассмотрим методику управления последовательным портом UART. Микроконтроллеры 8051/8052 имеют два вывода для передачи-приема данных в последовательном формате. Эти выводы обозначаются как P3.17TXD и P3.0/RXD. Фактически в микроконтроллерах 8052 используется минимально необходимая конфигурация для организации обмена данными по стандарту RS-232, показанная на рисунке 3.2. Таким образом, микроконтроллеры 8051/8052 могут легко сопрягаться с терминальными устройствами, например, с персональным компьютером, в котором установлена программа Hyper Terminal.

Последовательный порт устройств, совместимых с 8051, имеет несколько режимов работы и может быть легко запрограммирован всего несколькими командами Ассемблера.

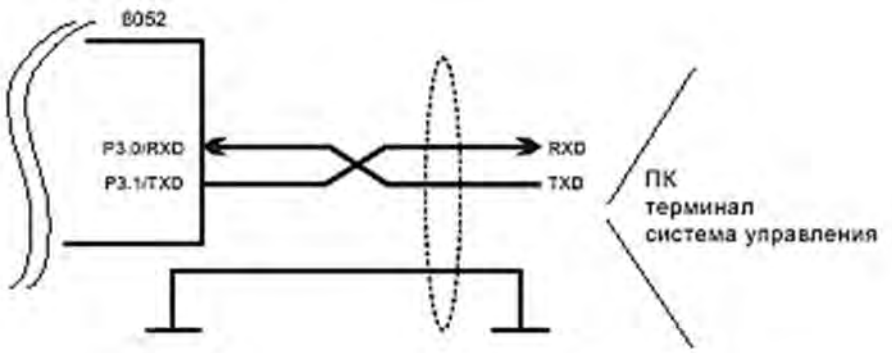


Рисунок 3.2 – Схема обмена данными с внешними устройствами по RS-232

Рассмотрим пример программного кода на Ассемблере, позволяющего передачу строки в последовательный порт:

```
NAME    PROCS
MAIN SEGMENT CODE
myDATA   SEGMENT CODE
CSEG AT 0
USING    0
JMP
start
RSEG MAIN Start:
MOV SCON, #50h
MOV  THl,  #0FDh
ORL  TMOD,  #20h
SETB TR1
MOV  DPTR,  #txt
again:
CLR  TI
CLR  A
MOVC A, @A+DPTR
CJNEA, UBh,
write_char:
JMP  $
MOV  SBUF, A
JNB TI, $
INC  DPTR
SJMP again
RSEG my DATA
txt:    DB   'HELLO', 1Bh
END
```

Команды ***MOV SCON, #50h и MOV THl, #0FDh*** устанавливает 8-битовый режим обмена данными и скорость обмена, определяемую таймером 1. Кроме того, устанавливается бит разрешения приема данных через последовательный

порт. В программе определена константная строка txt, которая размещается в памяти программы и заканчивается символом ESC (1Вп). Символ ESC используется программой для определения конца строки и завершения вывода.

Для передачи строки txt загружаем ее адрес в регистр DPTR командой MOV DPTR, #txt

Далее каждый байт строки загружается в регистр A, для чего используются команды ***CLR A и MOVC A, @A+DPTR***. Содержимое регистра-аккумулятора A анализируется на равенство символу ESC, что будет свидетельствовать об окончании передачи. Это выполняется командой ***CJNE A, JtlBh, write_char.*** Если обнаружен символ ESC, программа входит в бесконечный цикл, если же это иной символ, то он передается в последовательный порт. После окончания передачи очередного символа регистр DPTR инкрементируется, указывая на следующий символ строки txt, и цикл передачи байта повторяется.

Порядок выполнения работы

1 Написать программу на Ассемблере в соответствии с выше приведенным кодом, используя **Интегрированную среду PROVIEW**.

2 В качестве строковой переменной использовать собственное ФИО в английской интерпретации.

3 Выполнить компиляцию с помощью **PROVIEW**.

4 Распечатать листинг программы.

5 Распечатать результаты работы программы.

6 Подготовить отчет по лабораторной работе.

Контрольные вопросы

1 Какие модули микропроцессора 8051 (MSC-51) являются ресурсами?

2 Какую функцию выполняют таймеры MSC-51?

3 Как происходит считывание данных и команд?

4 Какие устройства можно подключить к последовательному порту?

5 Какие команды необходимы для настройки таймеров?

6 В какой регистр записываются данные для передачи в последовательный порт?

7 Можно ли использовать внешнюю память для MSC-51?

4 Лабораторная работа № 4. Разработка клиент-серверного приложения на C и Ассемблере

Цель работы: изучить технологию совместного программирования на языках C/C++ и Ассемблер; научиться создавать модули на языке Ассемблер, используемые в программах на C/C++; изучить транспортные сетевые протоколы TCP и UDP, используемые в клиент-серверных приложениях.

Организация интерфейса C/C++ и Ассемблера

Соглашения об именах. Для компоновки модулей C/C++ и Ассемблера необходимо, чтобы модули Ассемблера соблюдали соглашения об именах, принятые в C/C++. Все имена в ассемблерных модулях, которые будут использоваться в модулях на C/C++, должны начинаться с символа «_».

Модели памяти. Для того чтобы та или иная ассемблерная функция могла быть вызвана из C/C++, она должна использовать ту же модель памяти, что и программа на C/C++, а также совместимый с C/C++ кодовый сегмент и сегмент данных.

Сохранение регистров. В создаваемых ассемблерных программах можно использовать все регистры процессора. Но, чтобы предотвратить путаницу с функциями C и C++, необходимо восстанавливать BP, CS, SP и SS, которые они имели до запуска созданной подпрограммы. Тогда можно быть совершенно уверенным, что обращение к другим функциям не изменит эти регистры. Также нельзя забывать, что C использует регистры SI и DI для регистровых переменных, поэтому при использовании встроенного Ассемблера замедляется общая работа программы.

Вызываемые из C/C++ ассемблерные функции могут делать все, что угодно, при условии, что они обеспечат сохранность следующих регистров: EBP/BP, ESP/SP, CS, DS, SS.

Регистры EAX/AX, EBX/BX, ECX/CX, EDX/DX, ES и регистр EFLAGS/FLAGS могут меняться любым образом.

Передача параметров. C/C++ передает функциям параметры через стек. Перед вызовом функции C/C++ сначала помещает параметры, которые требуется передать этой функции, в стек, начиная с самого правого параметра и заканчивая левым.

Возврат значений. Вызываемая из C/C++ ассемблерная функция может возвращать значения так же, как и функция на C/C++.

Ассемблерные операторы Inline. Ассемблерные операторы Inline начинаются словом asm, за которым следует инструкция и ее операнды. Например, чтобы синхронизировать программу с внешним сигналом прерывания, можно написать:

```
/* ожидание прерывания*/
asm sti
asm hlt
printf(“Прерывание получено\n”)
```

Когда ранние версии Turbo C компилируют программу со встроенными командами asm, компилятор сперва создает ассемблерный текст для всей программы, вставляя в текст ассемблерные инструкции вместе с откомпилированным кодом для остальных операторов C. Затем компилятор вызывает Turbo Assembler и Linker (компоновщик), чтобы провести ассемблирование и подключить программу к конечному файлу кода. Более поздние версии Turbo и

Borland C++ могут компилировать операторы asm без вызова TASM. Полный синтаксис asm:

asm[метка] мнемоника/директива операнды [;] [/*C комментарий*/]

Точки с запятыми в конце строк asm и комментарии C, расположенные между /*и*/, удаляются из текста перед ассемблированием, поэтому их можно опускать в тексте программы.

Следующий пример демонстрирует организацию интерфейса C/C++ и Ассемблера с возвратом значения функцией на Ассемблере. Представленная в примере функция FindLC возвращает указатель на последний символ переданной ей строки (рисунок 4.1).

```
#include <stdio.h>
extern "C" char * FindLC(char *);

void main(void)
{
  char * TestString = "Кончается на У";
  printf("Строка \"%s\" заканчивается символом \'%c\'.\n",
        TestString, *FindLC(TestString));
}
;Программа на ассемблере
.model small
.code
public _FindLC
_FindLC proc
  push bp  mov bp, sp
  push di
  cld
  push ds
  pop es
  mov di, [bp+4] ; установить es:di как указатель на
                 ; начало переданной строки
  mov al, 0      ; ищется оканчивающий строку ноль
  mov cx, 0ffffh ; поиск выполняется в 64 Кбайтах -1
  repnz scasb    ; поиск нуля
  dec di         ; возврат указателя к нулю
  dec di         ; возврат указателя к последнему символу
  mov ax, di     ; возврат ближнего указателя через ax
  pop di
  pop bp
  ret
  _FindLC endp
end FindLC
```

Рисунок 4.1 – Листинг программного кода модуля на C/C++ с вложенной функцией на Ассемблере

Конечный результат, а именно ближний указатель на последний символ в переданной строке возвращается в вызывающий модуль через AX.

Для работы с сетью в программировании на языках высокого уровня в операционной среде Windows используются встроенные компоненты и их функции, такие как WinSocket2 или Socket. На рисунке 4.2 приведен пример кода для C/C++ функции работы с сетью.

Параметрами сокета (Socket) являются IP-адрес пунктов источника назначения и портов.

Порядок выполнения работы

1 Выполнить каждый программный листинг (см. рисунки 4.1 и 4.2) в отдельности, проверить работоспособность и объяснить каждую команду.

2 Используя листинг, приведенный на рисунке 4.2, вместо комментариев с символами «***» вставить подкорректированный ассемблерный модуль, аналогичный представленному на рисунке 4.1, с учетом следующего: имена аргументов в ассемблерном модуле и соответствующего параметра основной программы на C/C++ должны совпадать.

3 Строковый параметр на клиентском компьютере ввести с клавиатуры.

4 Сделать выводы по результатам исследований.

5 Оформить отчет.

```
DWORD WINAPI Cl_SrvThread(LPVOID lpParam)
    {  SOCKET sock=(SOCKET)lpParam;
       char szRecvBuff[1024],
         szSendBuff[1024];
       int ret;
       // Запуск бесконечного цикла
       while(1)
       {
         // Получение данных
         ret = recv(sock, szRecvBuff, 1024, 0);
         // Проверка полученных данных
         if (ret == 0)
           break;
         else if (ret == SOCKET_ERROR)
         {
           MessageBox(0, "Recive data filed", "Error", 0);
           break;
         }
         szRecvBuff[ret] = '\0';

         //***Здесь можно поставить проверку принятого текста (см. выше)
         // в переменной szRecvBuffer

         // Подготовка строки для отправки клиенту
         strcpy(szSendBuff, "Command get OK");

         // Отправка содержимого переменной szSendBuff клиенту
         ret = send(sock, szSendBuff, sizeof(szSendBuff), 0);
         if (ret == SOCKET_ERROR)
         {
           break;
         }
       }
       return 0;
    }
```

Рисунок 4.2 – Функции работы с сетью

Контрольные вопросы

1 Что означает директива ***#include <stdio.h>*** ?

2 Какой тип имеет строковая переменная?

3 Какие требования для именования переменных и модели памяти необходимы для того, чтобы корректно совместить ассемблерный модуль с модулем на C/C++ ?

4 Какие регистры сохраняются в случае вызова ассемблерной функции из программы на C/C++?

5 Каким образом осуществляется передача параметров из программы на C/C++ в функцию на Ассемблере и наоборот?

5 Лабораторная работа № 5. Разработка МПС с помощью симулятора PDS-51

Цель работы: изучить инструментальные средства и интегрированную среду разработки программного обеспечения Keil uVision с функцией симулятора семейства микроконтроллеров PDS-51.

Основные теоретические положения

Интегрированная среда разработки (Integrated Development Environment (IDE)) **uVision IDE** фирмы **Keil** позволяет непосредственно вызывать симулятор или внутрисхемный эмулятор и содержит богатый набор инструментальных опций.

***Редактор исходного кода* (*Source Code Editor*)**

Интегрированный в ***uVision*** редактор значительно облегчает подготовку исходного текста за счет многооконного выделения синтаксиса цветом и исправления ошибок в режиме диалога. Редактор настраивается на конкретный проект. Редактирование остается доступным и во время отладки программы. Это создает все условия для быстрого тестирования и корректировки приложения.

***Симулятор центрального процессора и интегрированной периферии* (*CPU &Peripheral Simulator*)**

uVision Simulator – чисто программный продукт, который осуществляет отладку в исходных кодах, симуляцию на уровне символов и отладку непосредственно на рабочей плате ***target debugging***. Моделируется вся система команд и все периферийные устройства.

Для просмотра и изменений установок периферии служат специальные диалоговые окна. Симулятор полностью поддерживает периферийные устройства микроконтроллера посредством специальных драйверов *xxx.DLL*.

Для симуляции аппаратной части PDS-51 (МК 8051) Keil предлагает ***Advanced Generic Simulation Interface* (*AGSI*)**. *AGSI* является спецификацией *API*, расширяющей возможности симуляции с помощью диалоговых настроек.

В распоряжение пользователя предоставляется ряд окон, отображающих состояния таймеров, портов, прерываний, сторожевого таймера, последовательного порта, аналого-цифрового преобразователя и т. д. Параметры этих устройств могут быть установлены и изменены в соответствии с контекстом приложения. ***uVision Simulator*** позволяет проводить пошаговую отладку программы, просматривая ее в окне *Debug*. Трассировщик запоминает команды и позволяет просматривать их в окне *Trace*. Изменение переменных отслеживает окно *Watch*. Вызовы процедур отображаются в окне Call-Stack.

Используя симулятор ***uVision,*** можно разработать микропроцессорную систему (МПС) даже без аппаратной части. Центральным модулем МПС является контроллер-микропроцессор. К нему подключаются электронные устройства,

такие как ЦАП, АЦП, модули внешней памяти, электронные компоненты, например, светодиоды и пр. Составляется алгоритм работы МПС. В соответствии с алгоритмом пишется программа на C/C++ или (и) на Ассемблере в среде ***uVision*** и сохраняется на диске ПЭВМ. Далее программа загружается в виртуальный МПС с помощью ***uVision***, проверяется, компилируется и запускается в тестовом режиме. По результатам теста выявляются ошибки и при необходимости корректируются.

Средства разработки (CA51, DK51, PK51) для классических 8051 микроконтроллеров (с поддержкой до 32 банков по 64 Кб памяти программ) содержат:

– C51 Compiler – компилятор C;

– A51 Macro Assembler – макроассемблер;

– BL51 Lincer/Locater – динамический загрузчик/компоновщик.

Пример программы для управления выходами P1 PDS-51, к которым подключены светодиоды, представлен на рисунке 5.1.

```
1 #include <reg51.h>
2
3 #define LED P1
4
5 void delay (unsigned int time)
6 {
7   unsigned int i;
8   for (i = 0; i < time; i++);
9 }
10
11 void main()
12 {
13   while (1)
14   {
15     LED=0x00;
16     delay(500);
17     LED=0xff;
18     delay(500);
19   }
20 }
```

Рисунок 5.1 – Пример программы для управления выходами P1, к которым подключены светодиоды

Интерфейс программы keil uVision в режиме отладки приведен на рисунке 5.2.

В режиме отладки keil uVision есть возможность выполнять программу по шагам, не имея под рукой микроконтроллера, а также ряд других возможностей, таких как просмотр ассемблерного кода, сгенерированного программой, и стека вызовов.

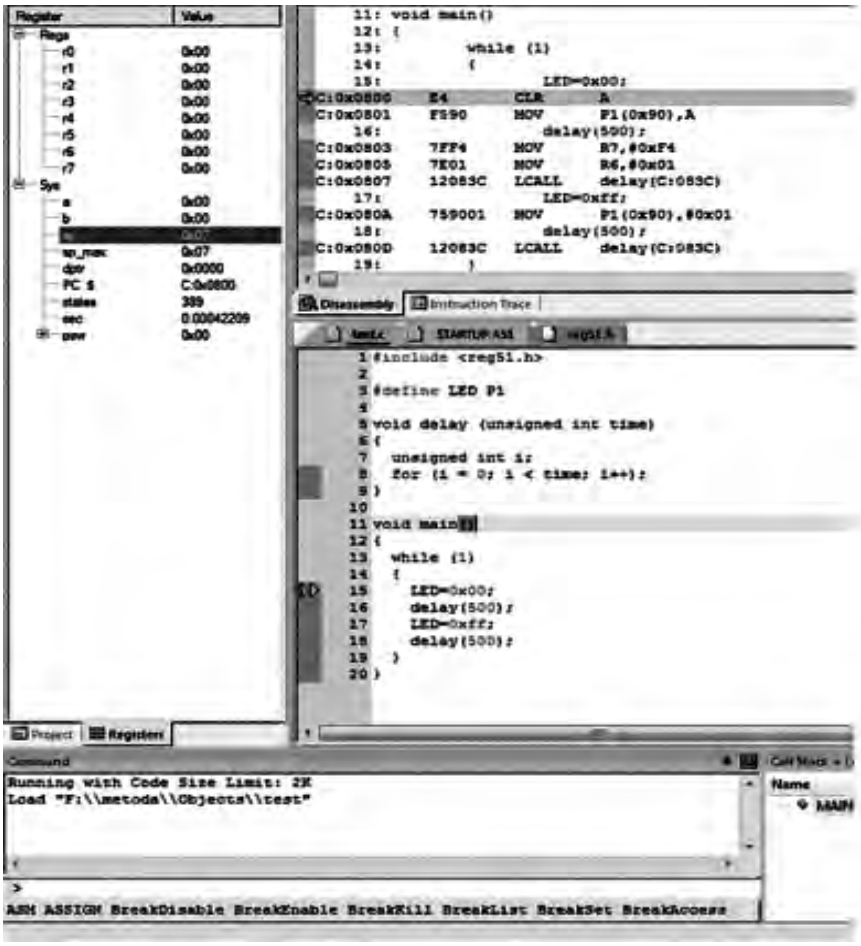


Рисунок 5.2 – Интерфейс программы keil uVision в режиме отладки

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Написать программу для микроконтроллера с подключенными светодиодами.

3 Выполнить отладку программы в среде keil uVision.

4 Оформить отчет.

6 Лабораторная работа № 6. Моделирование МПС на ПЭВМ

Цель работы: изучить инструментальные средства программы для автоматизированного проектирования электронных схем Proteus ISIS professional 7.

Основные теоретические положения

Пакет представляет собой систему схемотехнического моделирования, базирующуюся на основе моделей электронных компонентов, принятых в PSpice.

Отличительной чертой пакета PROTEUS VSM является возможность моделирования работы программируемых устройств: микроконтроллеров, микропроцессоров, DSP и пр. Причем в Proteus полностью реализована концепция сквозного проектирования, когда, например, инженер меняет что-то в логике работы схемотехники и программный пакет тут же «подхватывает» данные изменения в системе трассировки. Библиотека компонентов содержит справочные данные. Дополнительно в пакет PROTEUS VSM входит система проектирования печатных плат. Пакет Proteus состоит из двух частей, двух подпрограмм: ISIS – программы синтеза и моделирования непосредственно электронных схем и ARES – программы разработки печатных плат. Вместе с программой устанавливается набор демонстрационных проектов для ознакомления.

Также в состав восьмой версии входит среда разработки VSM Studio, позволяющая быстро написать программу для микроконтроллера, используемого в проекте, и скомпилировать.

Пакет является коммерческим. Бесплатная ознакомительная версия характеризуется полной функциональностью, но не имеет возможности сохранения файлов.

Примечательной особенностью является то, что в ARES можно увидеть 3D-модель печатной платы, что позволяет разработчику оценить своё устройство ещё на стадии разработки.

Система поддерживает подключение новых элементов (SPICE) и подключение разных компиляторов (PICOLO, ARM-подобные, AVR и далее).

При запуске программы автоматически загружается рабочая область Proteus, на которой можно размещать компоненты. Готовые модели сохраняются в файлах с расширением dsn. В результате будет открыто окно Pick Devices, в котором можно выбрать желаемый микроконтроллер либо любой другой электронный компонент. Выбор конкретного устройства сопровождается выводом графической информации в окнах Schematic Preview и PCB Preview. Под последним также можно выбирать тип корпуса и необходимо указать точку расположения элемента. Подобным образом на рабочую область можно добавить любые элементы.

Пояснения по выполнению работы

Для создания файла прошивки в программе keil uVision необходимо открыть настройки для группы файлов с исходным кодом, в настройках выходных параметров поставить галочку рядом с пунктом Create Hex file и выбрать формат файла – HEX-80 (рисунок 6.1). После этого следует пересобрать проект, и файл прошивки появится в папке Objects в директории проекта.

Для добавления элементов на схему в программе ISIS professional 7 необходимо нажать на кнопку pick from libraries (рисунок 6.2).

Для загрузки прошивки на микроконтроллер необходимо двойным кликом нажать на него на схеме, и откроется окно, в котором необходимо указать путь к .hex файлу с прошивкой (рисунок 6.3).

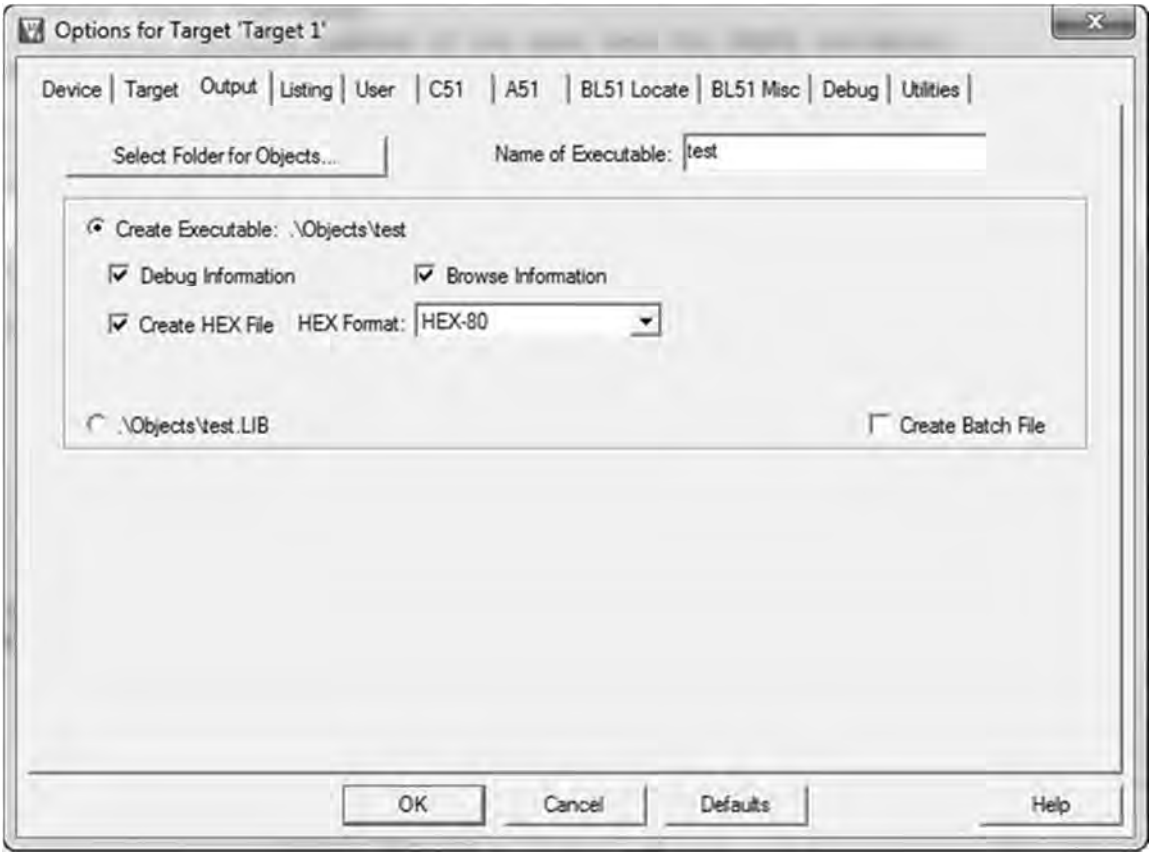


Рисунок 6.1 – Включение настройки формирования hex файла прошивки

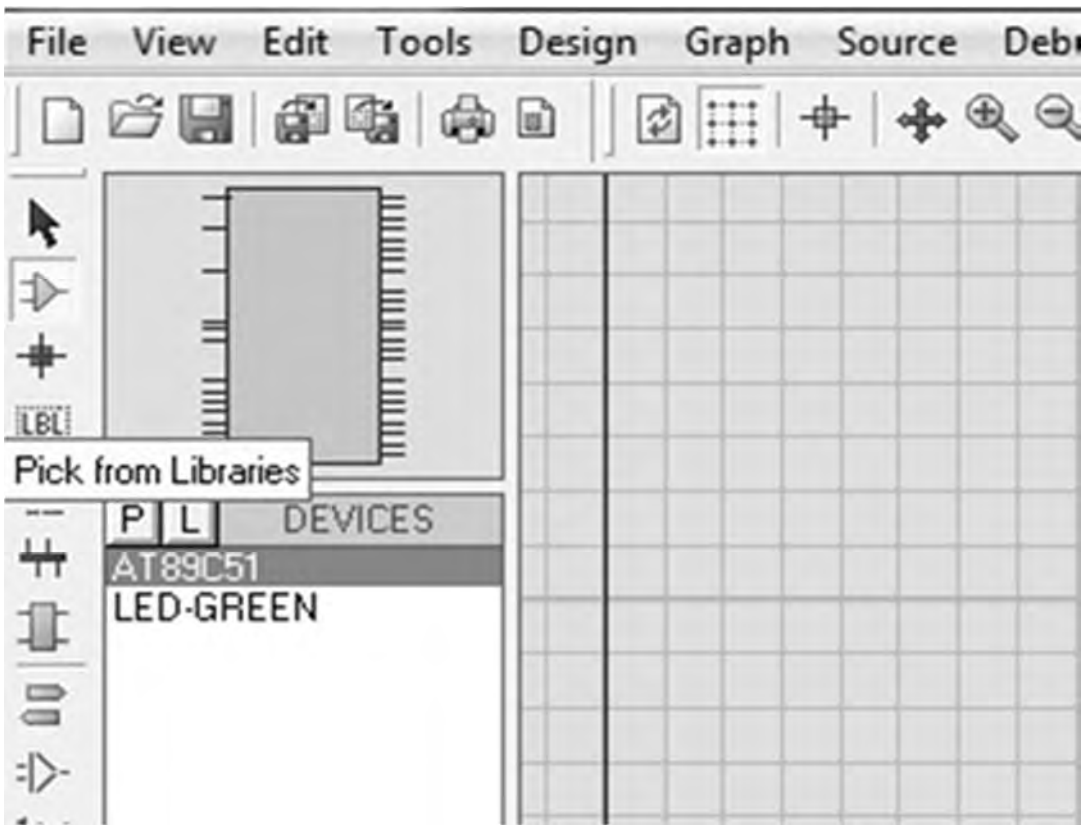


Рисунок 6.2 – Кнопка pick from libraries

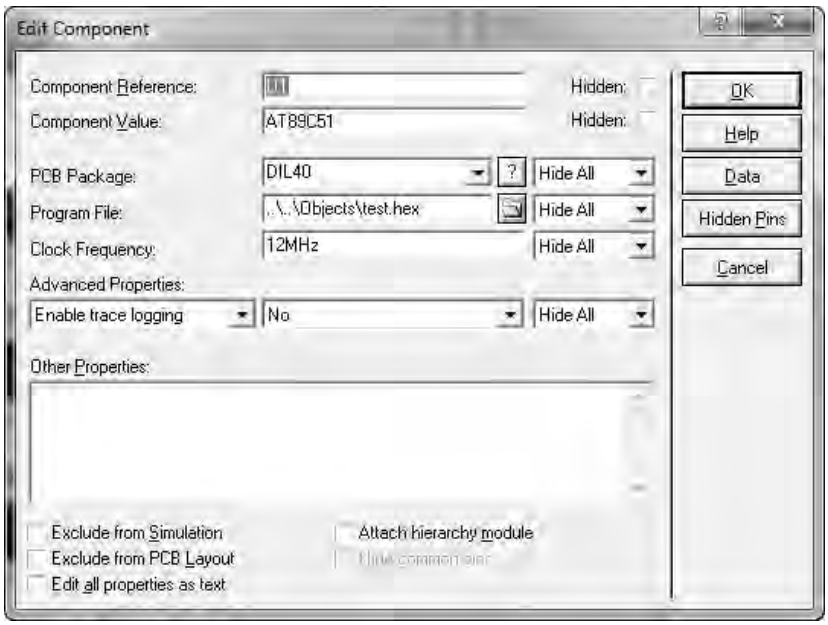


Рисунок 6.3 – Загрузка файла прошивки в микроконтроллер

Порядок выполнения работы

1 Создать файл прошивки для микроконтроллера из кода лабораторной работы № 5.

2 Собрать схему в программе ISIS Professional 7.

3 Проверить работоспособность схемы.

4 Оформить отчет.

7 Лабораторная работа № 7. Системная плата

Цель работы: изучить основные элементы системной платы компьютера с архитектурой PC AT.

Основные теоретические положения

Самым существенным элементом является главная плата, часто называемая основной, материнской или системной платой. Она представляет собой подсистему, содержащую базовые микросхемы ввода/вывода, оперативную память, а также микропроцессор. Далее представлено описание системной платы, базирующейся на чип-сете Intel 430 VX (TRITON VX), которая поддерживает процессоры семейства Pentium (Intel Pentium, Cyrix 6x86, AMD 5K86).

Системная плата с процессором семейства Pentium имеет достаточное количество слотов расширения и слотов для наращивания оперативной памяти, чтобы наращивать возможности системы, добавляя новые устройства и память. Данная плата для установки процессора использует ZIF (Zero Insertion Force)-разъем, который позволяет без особых усилий заменить процессор. Также эта материнская плата снабжена рядом перемычек для изменения типа и ве-

личины напряжения питания процессора, что позволяет использовать процессоры с различной архитектурой питания.

Системная плата, именуемая Intel Triton VX, полностью совместима на аппаратном уровне с системами MS-DOS, OS/2, Unix, Linux, MS Windows 9x/2000/NT, Novell NetWare.

Свойства системной платы Intel Triton VX.

Системная плата Intel Triton VX имеет все свойства высокоинтегрированной системы, приспособленной к взаимодействию с произвольным процессором семейства Intel Pentium, Cyrix 6x86, AMD 5K86. Высокую производительность обеспечивает внешняя кэш-память, имеющая размер 256 или 512 Кбайт в зависимости от модели системной платы. Далее перечислены возможности системной платы Intel Triton VX:

– возможность работ с произвольным процессором семейства Intel Pentium, Cyrix 6x86, AMD 5K86.

– размер кеш-памяти – 256 или 512 Кбайт;

– поддержка стандартов памяти EDO/Fast page DRAM, 72-pin SIMM разъемы памяти с возможностью установки 128 Мб; стандарт памяти SDRAM и один разъем 168-pin DIMM;

– выбор напряжения питания процессора – 2,5 или 3,3 В;

– поддержка USB – Universal Serial Bus;

– на плате имеется три PCI-слота и три ISA-слота (рисунок 7.1).

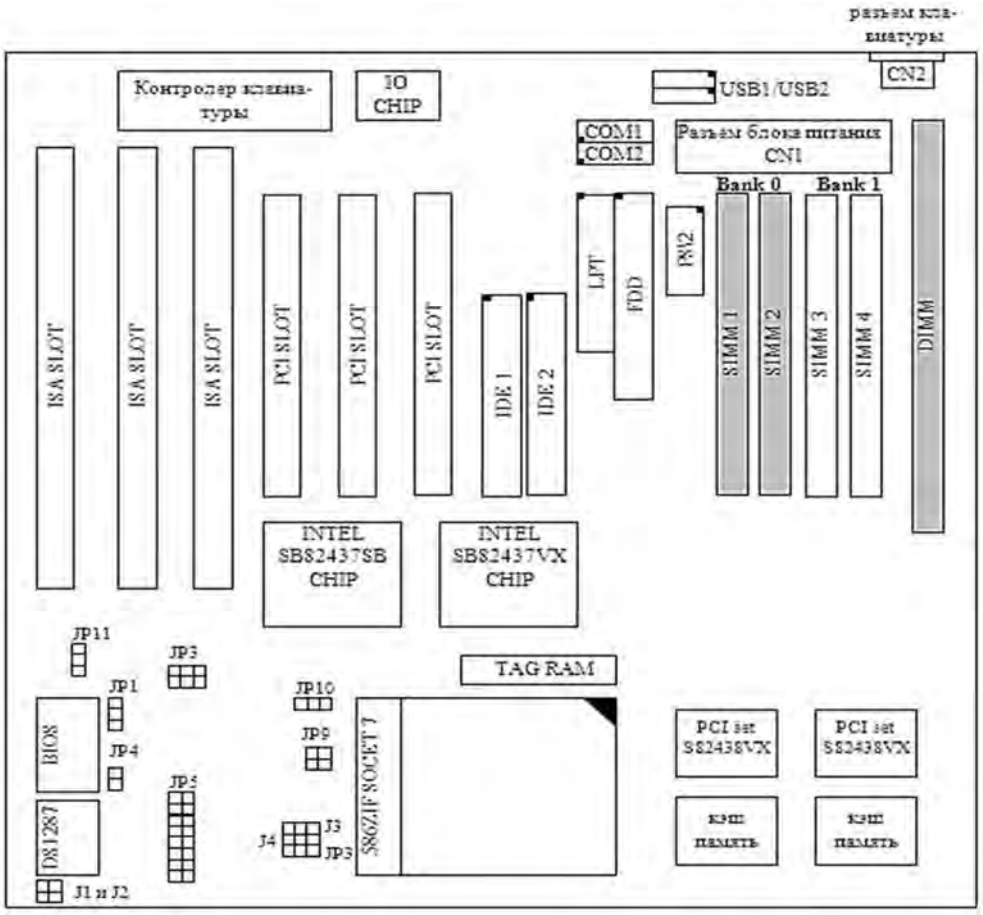


Рисунок 7.1 – Основные элементы системной платы

Свободное аппаратное наращивание системы, заключающиеся в замене процессора и кварцевого генератора, требует на плате достаточно большого количества перемычек, позволяющих подобрать соответствующую конфигурацию, зависящую от типа установленного процессора.

Описание перемычек, используемых для выбора типа процессора

JP1: Выбор частоты тактового генератора для системной AT-шины.

Для установки частоты тактового генератора для AT-шины требуется изменить положение перемычки JP1 соответственно с типом процессора, установленного на системной плате.

При замкнутом втором и третьем контактах устанавливается частота системной шины 60 или 66 МГц (по умолчанию) (рисунок 7.2). В этом случае задающая частота для PCI-устройств равна 15 или 16,5 МГц. При замкнутом первом и втором контактах устанавливается частота системной шины 50; 55 или 75 МГц.

Рисунок 7.2 – Положение перемычек для установки соответствующих частот системной AT-шины

JP3, JP11: Выбор частоты тактового генератора шины процессора.

Для установки частоты тактового генератора для шины процессора требуется изменить положение перемычки JP3 и JP11 (таблица 7.1) соответственно с типом процессора, установленного на системной плате (рисунок 7.3).

Таблица 7.1 – Варианты установки перемычек для выбора частоты шины процессора

JP3	Частота шины	Частоты процессора
2-3, 5-6	50 МГц, вариант ***а*** на рисунке 7.3	75 МГц
1-2, 4-5	55 МГц или 75 МГц, вариант ***б*** на рисунке 7.3	110, 200 (Cyrix) МГц
2-3, 4-5	60 МГц, вариант ***в*** на рисунке 7.3	90, 120, 150, 180 МГц
1-2, 5-6	66 МГц, вариант ***г*** на рисунке 7.3	100, 133, 166, 200 МГц

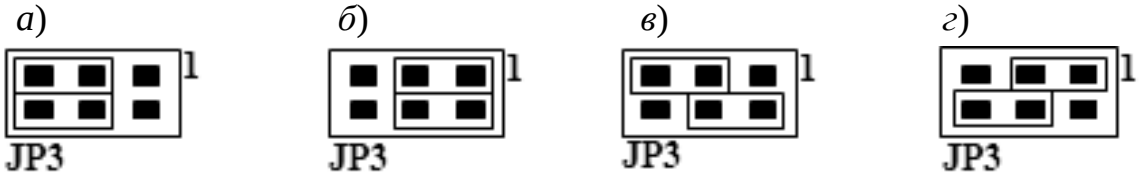


Рисунок 7.3 – Варианты установки перемычек JP3 для выбора частоты шины

Рисунок 7.4 – Варианты установки перемычек JP11 для выбора частоты шины процессора

J1/J2: Выбор множителя частот для процессора.

Для установки частоты тактового генератора для шины процессора требуется изменить положение перемычек J1/J2 (таблица 7.2).

Таблица 7.2 – Варианты установки перемычек для выбора множителя

J1	J2	Множитель	Частоты процессора
OFF	OFF	1,5, вариант **а** на рисунке 7.5	75, 90, 100 МГц
ON	OFF	2,0, вариант ***б*** на рисунке 7.5	120, 133 МГц
ON	ON	2,5, вариант ***в*** на рисунке 7.5	150, 166 МГц
OFF	ON	3,0, вариант *г* на рисунке 7.5	180, 200 МГц

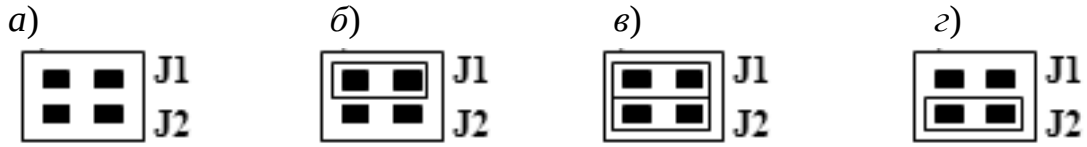


Рисунок 7.5 – Варианты установки перемычек для выбора множителя

JP9, JP2, J4: Выбор питания процессора.

Для нормальной работы процессора требуется перемычками JP9, JP2 и J4 выставить нужное для процессора напряжение (рисунок 7.6). Обычно напряжение и тип питания указывается на процессоре.

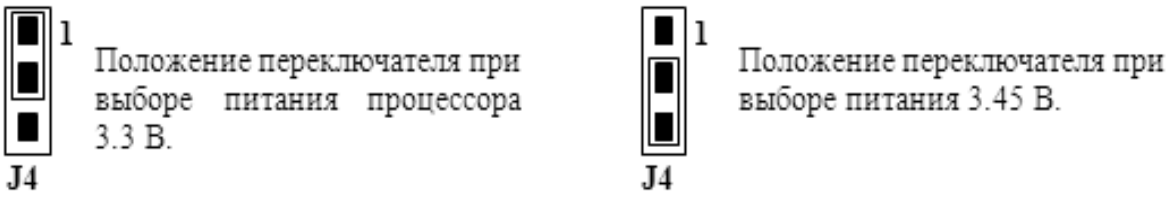


Рисунок 7.6 – Выбор напряжения одиночного питания процессора

При изменении архитектуры процессора изменяется и тип питания. Это потребовало введения двойного питания процессора. Для изменения типа питания процессора служит перемычка JP9 (рисунок 7.7). Двойное питание процессора применяется в процессорах Intel Pentium с частотой 180 и 200 МГц.

Для увеличения частоты процессора требуется поднять напряжения питания. Для этого служит перемычка JP2 (см. рисунок 7.2).

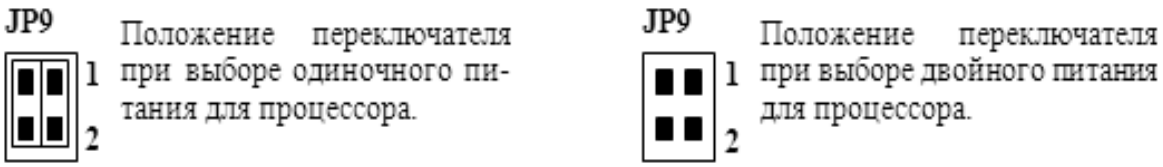


Рисунок 7.7 – Выбор типа питания процессора

Конфигурация кеш-памяти.

В оборудовании класса IBM PC оперативная память размещается на системной плате. Старые модели плат имели большие размеры и включали в себя контактные панельки под дискретные микросхемы (DIP). Благодаря техническому прогрессу стало возможным применение в качестве оперативной памяти модулей типа SIMM, DIMM. Модули SIMM базируются на микросхемах динамической памяти типа Fast page DRAM, EDO DRAM, а модули DIMM базируются на микросхемах динамической памяти типа SDRAM, Direct RDRAM (Rambus). У этих микросхем время обращения к памяти является достаточно большим. При этом в системах с большой вычислительной мощностью, а также с высокой тактовой частотой процессора возникает необходимость продления времени пересылки сигналов управления доступом к памяти путём подачи тактов ожидания.

Возрастающие запросы потребителей компьютеров и требования прикладных программ вынуждают увеличивать вычислительную мощность микропроцессора и системы в целом. Одним из методов увеличения вычислительной мощности является оборудование системы быстродействующей вспомогательной кеш-памятью.

Запрос процессором данных из памяти «перехватывается» системой управления кеш-памятью, которая проверяет, находятся ли данные, требуемые процессором, во вспомогательной памяти. Если данные, требуемые процессором, находятся во вспомогательной памяти, то контроллер кеш-памятью пересылает данные в процессор без введения циклов готовности. В противном случае контроллер считывает данные из оперативной памяти, записывает их в кеш-память и посылает в процессор. Если процессор потребует записи данных, то контроллер проверяет, находятся ли уже эти данные во вспомогательной памяти. Если этот так, то данные записываются во вспомогательную и в основную память одновременно. Выбор размера кеш-памяти выполняется при помощи перемычки JP10 (см. рисунок 7.1).

Конфигурация оперативной памяти системы.

Системная плата дает возможность расширения оперативной памяти путем установки дополнительных модулей SIMM (Single In-Line Memory Modules) или установкой модуля DIMM. Системная плата содержит четыре 72-pin контактных гнезда, в которые устанавливается определенное количество модулей SIMM. В зависимости от типа модулей SIMM и их количества возможно получение соответствующей емкости оперативной памяти. Модули SIMM должны иметь время доступа 70 нс для fast page SIMM или 60 нс для EDO SIMM.

Память размещается в двух банках: Банк 0 и Банк 1. Каждый банк содер-

жит по два гнезда для SIMM-модулей. В каждом гнезде можно устанавливать модули типов FP SIMM и EDO SIMM размером 4; 8; 16; 32; 64 Мб, при этом размер памяти может быть от 8 до 64 Мбайт в одном банке и от 8 до 128 Мбайт оперативной памяти в системе. Все гнезда в данном банке должны быть полностью заполнены, а модули SIMM в пределах банка должны быть одного типа.

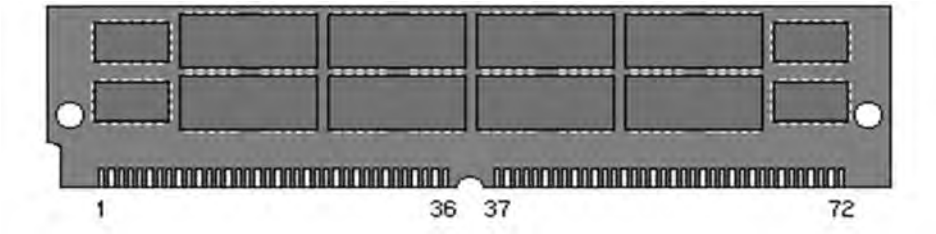


Рисунок 7.8 – Внешний вид модуля памяти SIMM

Порядок установки памяти типа SIMM следующий:

1) модуль SIMM необходимо установить таким образом, чтобы контактные площадки на плате модуля совпали с входными гнездами SIMM на системной плате;

2) далее следует разместить модуль SIMM в гнезде под углом 45° и легко дожать так, чтобы контакты модуля надежно вошли в гнездо (рисунок 7.9);

3) следующим шагом является установка модуля в вертикальное положение таким образом, чтобы модуль SIMM надежно защелкнулся в гнезде;

4) следует повторить описанные выше действия до полного заполнения гнезд SIMM в данном банке.

Рисунок 7.9 – Установка модуля памяти

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Получить задание у преподавателя, выполнить типовые задания.

3 Сделать выводы по результатам исследований.

4 Оформить отчет.

Контрольные вопросы

1 Перечислите основные компоненты системной платы.

2 Назовите основные возможности системной платы.

3 Каковы особенности выбора питания процессора?
4 Укажите порядок установки модулей памяти.

8 Лабораторная работа № 8. Изучение микропроцессорного ядра и контроллера шины в архитектуре PC AT

Цель работы: изучить принципиальную схему и функции микропроцессорного ядра и контроллера шины в архитектуре PC AT.

Основные теоретические положения

На рисунке 8.1 представлена принципиальная схема микропроцессорного ядра, которое состоит из микропроцессора 80286, арифметического сопроцессора 80287 и синхронизатора 80284. Вход -ERROR микропроцессора (МП) в PC AT не используется. Контроль ошибок сопроцессора выполняется через прерывание по каналу IRQ 13, сброс микропроцессора RESET происходит по включению питания или от кнопки «СБРОС».

Содержимое разряда адреса A20 микропроцессора перед выходом на локальную шину проходит через специальный элемент. Его присутствие обусловлено тем, что микропроцессор 80286 имеет отличие от 8086 при работе в реальном режиме времени, которое заключается в следующем. Например, если содержимое регистра CS равно FF00h, а регистра IP – 1000h, то в микропроцессоре 8086 выборка команды будет выполняться по адресу 00000h, при этом перенос из 19-го разряда адреса просто пропадает. В микропроцессоре 80286 физический адрес будет 100000h. Поэтому, чтобы обеспечить совместимость с программами, написанными для микропроцессора 8086, выход 20-го разряда адреса микропроцессора не подается на локальную шину в реальном режиме. Если выполняется переключение в защищенный режим или необходимо обратиться к памяти по адресам выше первого мегабайта, выполняется запись в порт 64 нужной константы, сигнал A20GATE становится активным и содержимое A20 на локальной шине совпадает со значением 20-го разряда адреса микропроцессора.

На рисунке 8.2 представлена принципиальная схема контроллера шины, который по состоянию микропроцессора формирует необходимую команду шины и соответствующие сигналы управления передачей данных DEN, DT/-R.

В архитектуре PC AT 286 имеется три шины: адреса, данных и управляющих сигналов. Синхрогенератор генерирует тактовый сигнал CLK для синхронизации внутреннего функционирования процессора и других микросхем. Сигнал RESET производит сброс процессора в начальное состояние. Сигнал READY также формируется с помощью синхрогенератора, предназначен для удлинения циклов шины при работе с медленными периферийными устройствами.

Сигнал AIOW активен в цикле записи в порт ввода-вывода и используется для управления формирователями шины данных. Этот сигнал становится пас-

сивным на один период синхронизации позже, чем команда -IOW, что обеспечивает удержание данных на входах порта по отношению к стробу занесения. На этом же листе представлены формирователи шины данных, которые разделяют локальную и системную шины. Элемент LS646 представляет собой шинный формирователь со встроенными регистрами.

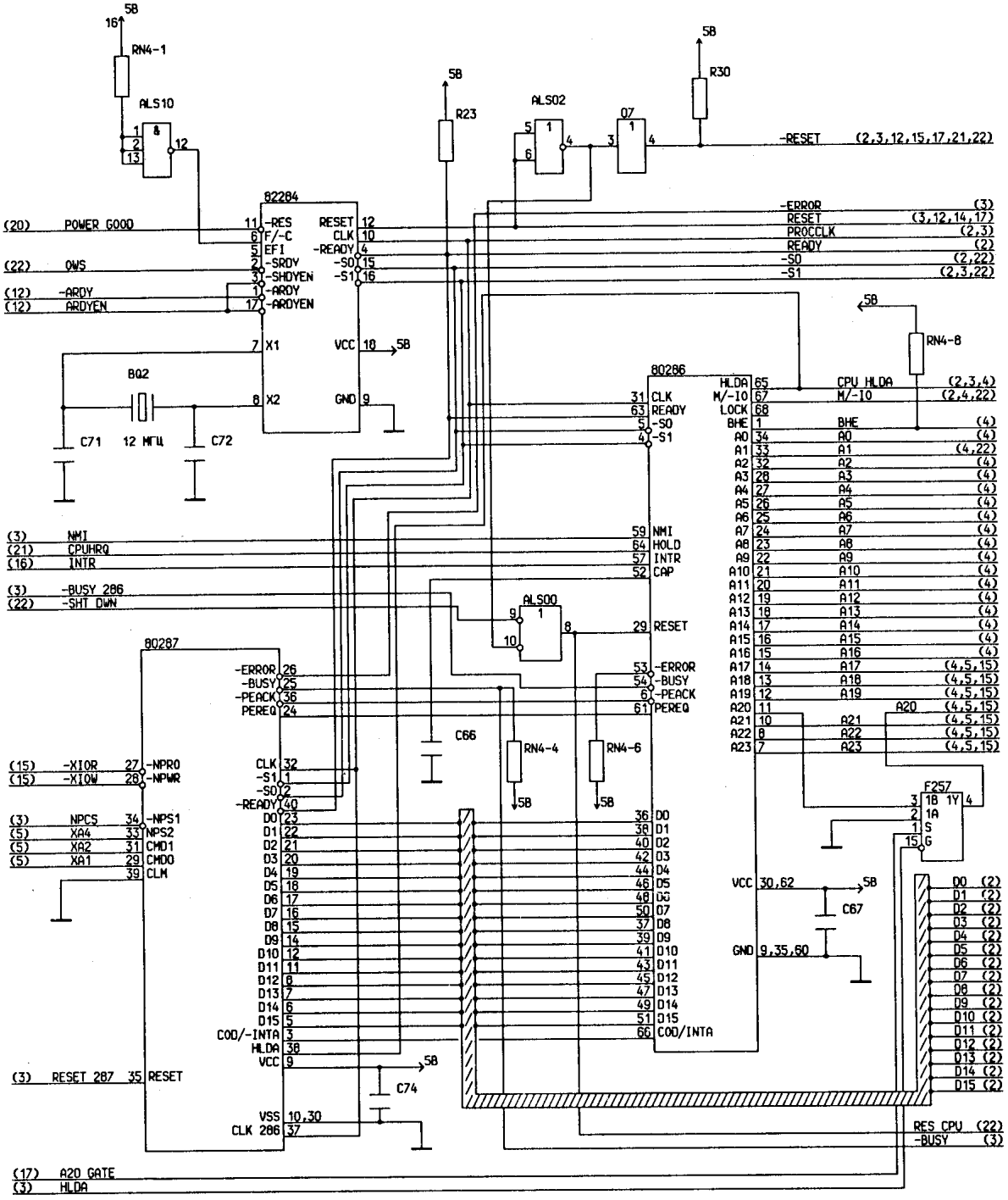


Рисунок 8.1 – Принципиальная схема микропроцессорного ядра

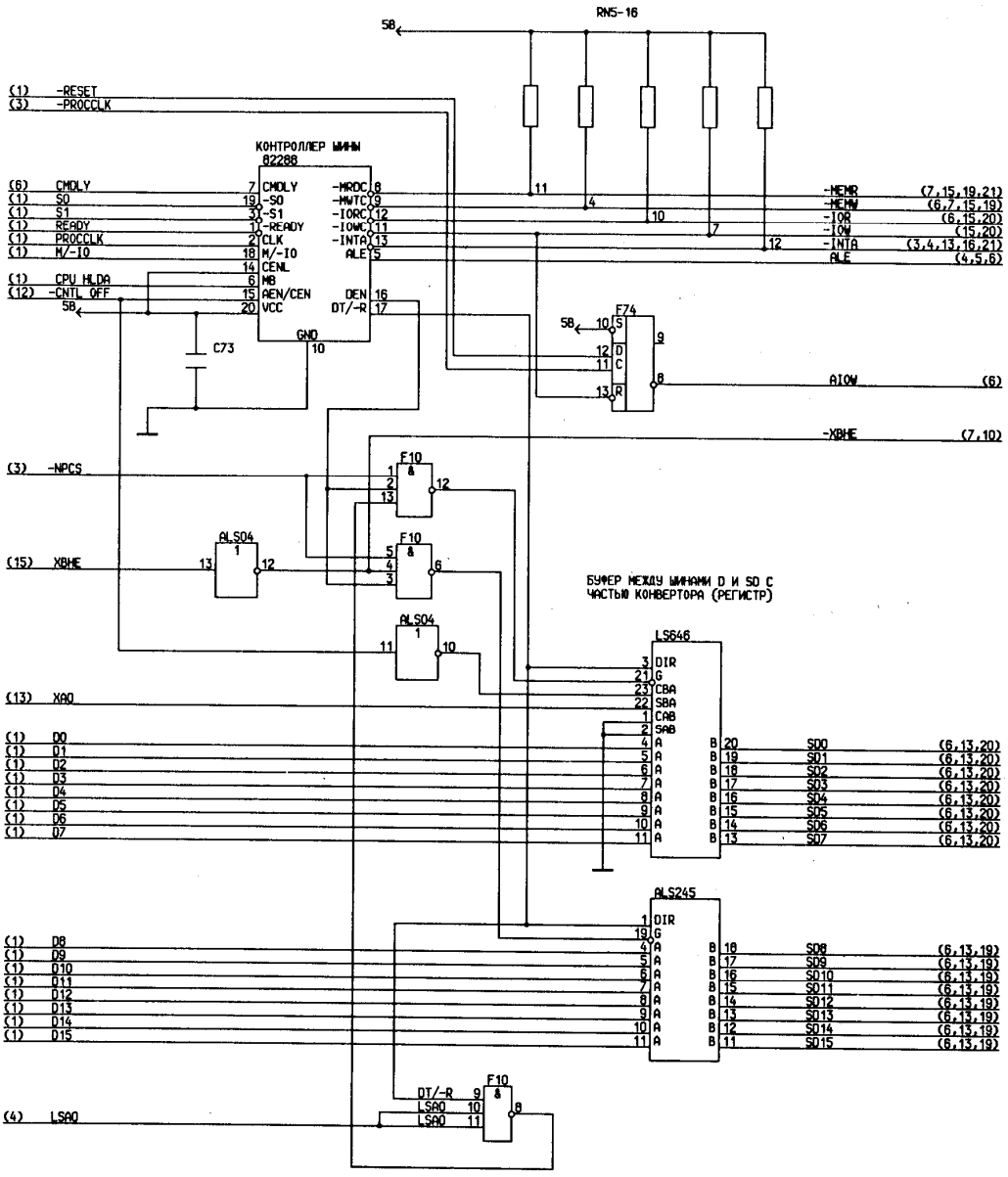


Рисунок 8.2 – Принципиальная схема контроллера шины

На адресную шину, состоящую из 24 линий, микропроцессор i80286 выставляет адрес байта или слова, которые будут пересылаться по шине данных в процессор или из него. Кроме того, шина адреса используется микропроцессором для указания адресов (номеров) периферийных портов, с которыми производится обмен данными.

Шина данных состоит из 16 линий, по которым возможна передача как отдельных байтов, так и двухбайтовых слов. Шина данных двунаправленна, т. к. передача байтов и слов может производиться как в микропроцессор, так и из него.

Шина управления формируется, во-первых, сигналами, поступающими непосредственно от микропроцессора, во-вторых, сигналами, сформированными системным контроллером, и, в-третьих, сигналами, идущими к микропроцессору от других микросхем и периферийных адаптеров. Микропроцессор ис-

пользует системный контроллер для формирования управляющих сигналов, определяющих правила переноса данных по шине.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Получить задание у преподавателя, выполнить типовые задания.

3 Изучить рисунки 8.1 и 8.2.

4 Определить функциональное назначение основных сигналов микропроцессорного ядра и контроллера шины.

5 Оформить отчет.

Контрольные вопросы

1 Элемент B02 на рисунке 8.1 – это:

а) резистор;
б) кварцевый резонатор;
в) конденсатор;
г) триггер.

2 Сигнал разрешения передачи старшего байта – это:

а) BHE;
б) BALE;
в) OE;
г) CE.

3 Схемная земля – это:

а) OSC;
б) CS;
в) GND;
г) DCS.

4 CPU HLDA – это сигнал:

а) на захват шины МП;
б) синхросигнал;
в) ошибка сопроцессора;
г) выбор МП.

5 Сигнал VCC – это:

а) прерывание;
б) питание;
в) сигнал ошибки;
г) сигнал записи.

6 Сигнал NPCS – это:

а) дешифратор строк;
б) немаскируемое прерывание;
в) подтверждение на захват прерывания;
г) выбор сопроцессора.

7 На рисунке 8.1 элемент 80287 – это:

а) сопроцессор;

б) шина;

в) процессор;

г) микросхема.

8 На рисунке 8.2 элемент F74 – это:

а) DC-триггер;

б) RS-триггер;

в) JK-триггер;

г) D-триггер.

9 Лабораторная работа № 9. Изучение топологии компьютерной сети

Цель работы: изучить топологии вычислительных сетей.

Основные теоретические положения

Сетевая топология (от греч. – место) – способ описания конфигурации сети: схема расположения и соединения сетевых устройств, или/и схема прохождения электрических сигналов, или/и описание направления потоков информации, или/и принцип предоставления доступа к сети.

Сетевая топология может быть:

1) физическая – описывает реальное расположение и связи между узлами сети – способ физического соединения компьютеров с помощью среды передачи, например, участками кабеля;

2) логическая – описывает прохождение сигнала в рамках физической топологии. Таким образом, логическая топология описывает пути передачи потоков данных между сетевыми устройствами и определяет направление и способ передачи, а не схему соединения физических проводников;

3) информационная – описывает направление потоков информации, передаваемых по сети;

4) управление обменом – это принцип передачи права на пользование сетью.

Выделяют три базовые топологии и ряд дополнительных производных типовой сетевой топологии, объединяющих компьютеры в единую сеть (таблица 9.1)

Дополнительные способы являются комбинациями базовых. В общем случае такие топологии называются смешанными или гибридными, но некоторые из них имеют собственные названия, например, «Дерево».

Рассмотрим базовые типы топологий.

Таблица 9.1 – Типы сетевых топологий

Базовый тип сетевых топологий	Производные (дополнительные) типы сетевых топологий
Шина Звезда Кольцо	Дерево Ячеистая топология Полносвязная Двойное кольцо Fat Tree Решётка

Топология Шина

Топология Шина относится к наиболее простым и широко распространенным топологиям в 1980-х гг. В ней используется один кабель, именуемый магистралью или сегментом, вдоль которого подключены все компьютеры сети (рисунок 9.1).

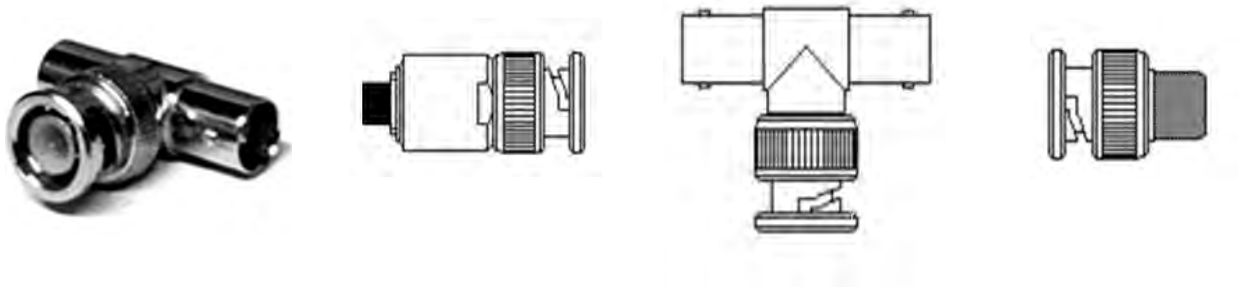
Рисунок 9.1 – Сетевая топология Шина

В настоящее время топология Шина считается устаревшей, но до сих пор используется. В Шине всегда реализуется режим так называемого полудуплексного (half duplex) обмена (в обоих направлениях, но по очереди, а не одновременно).

Топология Шина применяется в локальных сетях с архитектурой Ethernet (классы 10Base-5 и 10Base-2 для толстого и тонкого коаксиального кабеля соответственно). В настоящее время толстый коаксиальный кабель для Ethernet нигде не применяется. Что касается Ethernet 10Base-2, применяется тонкий коаксиальный кабель (0,2 дюйма). Каждый компьютер подключается к коаксиальному кабелю с помощью Т-разъема (Т-коннектор), т. е. Т-коннектор включается в сетевую карту компьютера, а к свободным концам коннектора подключаются куски кабеля от предыдущего и последующего компьютера. К первому компьютеру и к последнему, на свободные концы Т-коннектора устанавливаются терминаторы (рисунок 9.2).

Терминаторы используются для гашения сигналов, которые достигают концов канала передачи данных. Таким образом, информация поступает на все узлы одновременно, но принимается только тем узлом, у которого MAC-адрес совпадает с адресом в информационном кадре. Описанная выше топология относится к физической и логической топологии Шина.

а) *б*) *в*) *г*)



а – внешний вид Т-коннектора; *б* – терминатор; *в* – Т-коннектор; *г* – BNC-разъем

Рисунок 9.2 – Типы подключений

Топология Звезда

Звезда – это топология сети с явно выделенным центром, к которому подключаются все остальные абоненты (рисунок 9.3). Обмен информацией идет исключительно через центральный узел: компьютер, коммутатор, маршрутизатор. Никакие конфликты в сети с топологией Звезда в принципе невозможны, т. к. управление полностью централизовано.

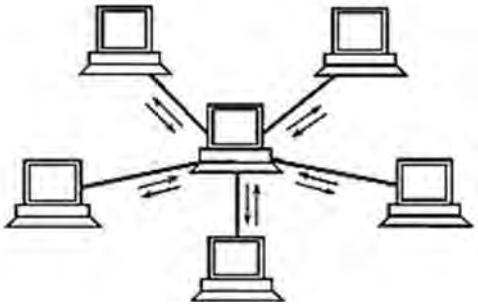


Рисунок 9.3 – Топология Звезда (активная Звезда)

Если говорить об устойчивости Звезды к отказам компьютеров, то выход из строя периферийного компьютера или его сетевого оборудования никак не отражается на функционировании оставшейся части сети, зато любой отказ центрального компьютера делает сеть полностью неработоспособной.

Обрыв кабеля или короткое замыкание в нем при топологии Звезда нарушает обмен только с одним компьютером, а все остальные компьютеры могут нормально продолжать работу.

В отличие от Шины, в Звезде на каждой линии связи находятся только два абонента: центральный и один из периферийных. Для их соединения используется две линии связи (в одном кабеле), каждая из которых передает информацию в одном направлении, т. е. на каждой линии связи имеется только один приемник и один передатчик. Это так называемая дуплексная передача точка-точка.

Серьезный недостаток топологии Звезда состоит в жестком ограниче-

нии количества абонентов. Обычно центральный узел может обслуживать не более 8-48 периферийных абонентов. В Звезде допустимо подключение вместо периферийного еще одного центрального абонента (в результате получается топология из нескольких соединенных между собой Звёзд).

Звезда, показанная на рисунке 9.3, называется активной или истинной Звездой. Существует также топология, называемая пассивной Звездой, которая внешне похожа на Звезду (рисунок 9.4), но логика работы у неё другая.

В центре сети с данной топологией помещается не компьютер (коммутатор, маршрутизатор), а специальное устройство – концентратор или, как его еще называют, хаб (hub), которое выполняет ту же функцию, что и повторитель, т. е. восстанавливает приходящие сигналы и пересылает их на все линии связи всем компьютерам одновременно (см. рисунок 9.4).

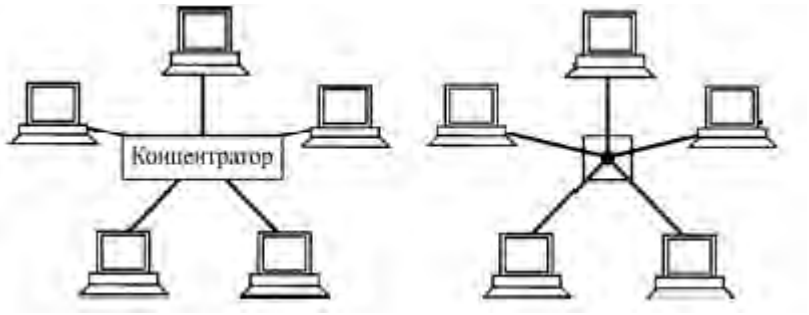


Рисунок 9.4 – Топология пассивная Звезда и ее эквивалентная схема

Информация поступает на все рабочие станции, но обрабатывается только теми станциями, которым она предназначается. Так как сигналы распространяются одновременно во все направления, то логическая топология данной локальной сети является логической Шиной.

Данная топология применяется в локальных сетях с 10 Base-T Ethernet (рисунок 9.5).

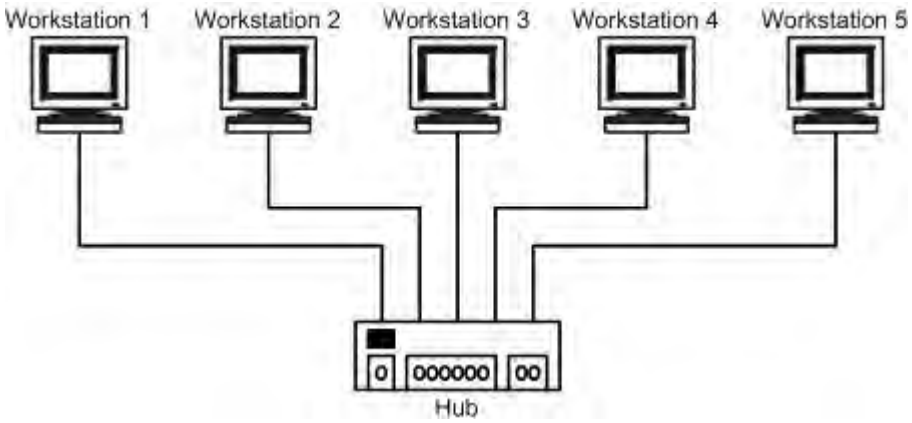


Рисунок 9.5 – Подключение ПК к концентратору

Топология Кольцо

Кольцо – это топология, в которой каждый компьютер соединен линиями связи с двумя другими: от одного он получает информацию, а другому переда-

ет. На каждой линии связи, как и в случае Звезды, работают только один передатчик и один приемник (связь типа точка-точка) (рисунок 9.6).

Важная особенность Кольца состоит в том, что каждый компьютер ретранслирует (восстанавливает, усиливает) приходящий к нему сигнал, т. е. выступает в роли репитера. Поэтому предельная длина Кольца может достигать NLпр, где N – количество компьютеров в Кольце; Lпр – предельная длина кабеля, ограниченная затуханием. Кольцо в этом отношении существенно превосходит любые другие топологии. В топологии Кольцо право на передачу (или, как еще говорят, на захват сети) переходит последовательно от одного компьютера к следующему по кругу.

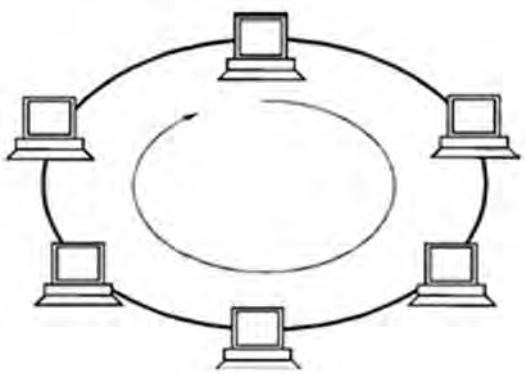


Рисунок 9.6 – Сетевая топология Кольцо

Подключение новых абонентов в Кольцо требует обязательной остановки работы всей сети на время подключения. Максимальное количество абонентов в Кольце может быть довольно велико (до тысячи и больше). Кольцевая топология обычно обладает высокой устойчивостью к перегрузкам, обеспечивает уверенную работу с большими потоками передаваемой по сети информации.

Выход из строя хотя бы одного из компьютеров (или же его сетевого оборудования) нарушает работу сети в целом. Это существенный недостаток Кольца.

Точно так же обрыв или короткое замыкание в любом из кабелей Кольца делает работу всей сети невозможной. Из трех рассмотренных топологий Кольцо наиболее уязвимо к повреждениям кабеля, поэтому в случае топологии Кольцо обычно предусматривают прокладку двух (или более) параллельных линий связи, одна из которых находится в резерве (рисунок 9.7).

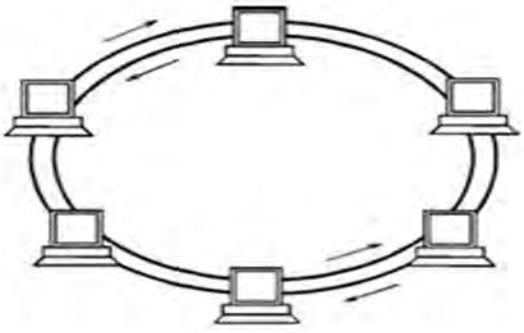


Рисунок 9.7 – Сеть с двумя кольцами

Иногда сеть с топологией Кольцо выполняется на основе двух параллельных кольцевых линий связи, передающих информацию в противоположных направлениях (см. рисунок 9.7). Цель подобного решения – увеличение (в идеале – вдвое) скорости передачи информации по сети и повышение надежности сети.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Зарисовать расположение компьютеров в классе и определить тип топологии локальной сети.

3 Описать преимущества и недостатки каждого из базовых типов топологии.

4 Оформить отчет.

10 Лабораторная работа № 10. Построение LAN различной топологии

Цель работы: изучить принципы построения компьютерных сетей базовой топологии базовым набором аппаратных средств с применением программного обеспечения – симулятора сетей Packet Tracer.

Основные теоретические положения

Для построения моделей сетей различной топологии с разной степенью сложности используют симуляторы. На рисунке 10.1 представлен симулятор Cisco Packet Tracer фирмы Cisco Systems.

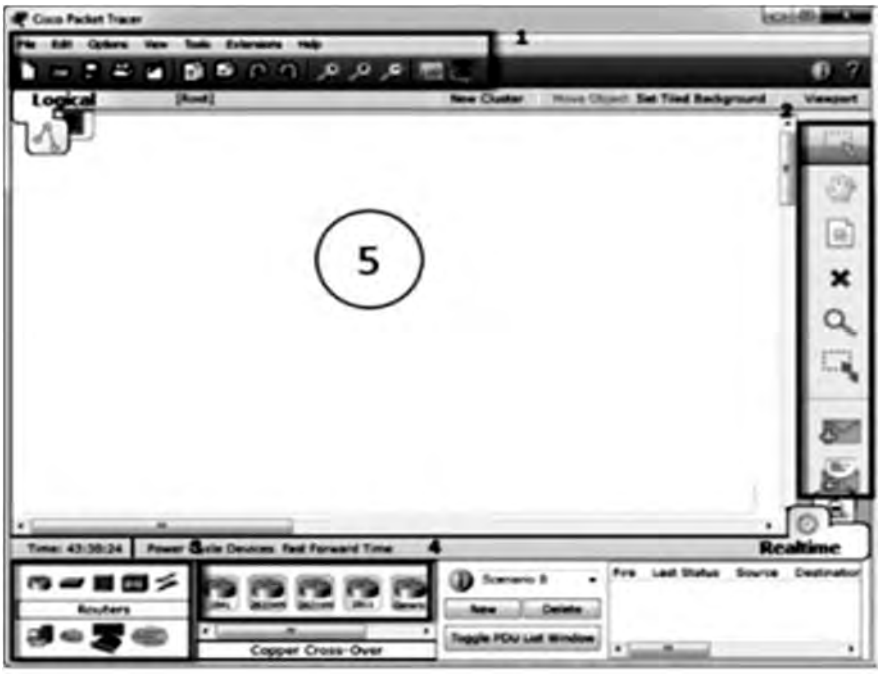


Рисунок 10.1 – Интерфейс программы Cisco Packet Tracert с основными инструментами

Программный продукт **Packet Tracer (PT)** является симулятором сети передачи данных, разработанным фирмой Cisco Systems. ***Packet Tracer*** позволяет имитировать работу различных сетевых устройств: маршрутизаторов, коммутаторов, точек беспроводного доступа, персональных компьютеров, сетевых принтеров, IP-телефонов и т. д. С помощью ***Packet Tracer*** можно создавать работоспособные модели сети.

На рисунке 10.1 отмечены наиболее часто используемые меню программы Cisco Packet Tracer.

1 Стандартное программное меню.

2 Правое графическое меню.

3 Меню выбора типа сетевых устройств.

4 Меню выбора конкретного сетевого устройства.

5 Рабочее пространство («Рабочий стол PT»).

Порядок выполнения работы

1 Изучить основные теоретические положения.

2 Необходимо смоделировать локальную сеть (LAN) с использованием коммутатора в качестве центрального коммутирующего сетевого устройства и концентратора, как устройства для расширения сети (рисунок 10.2).

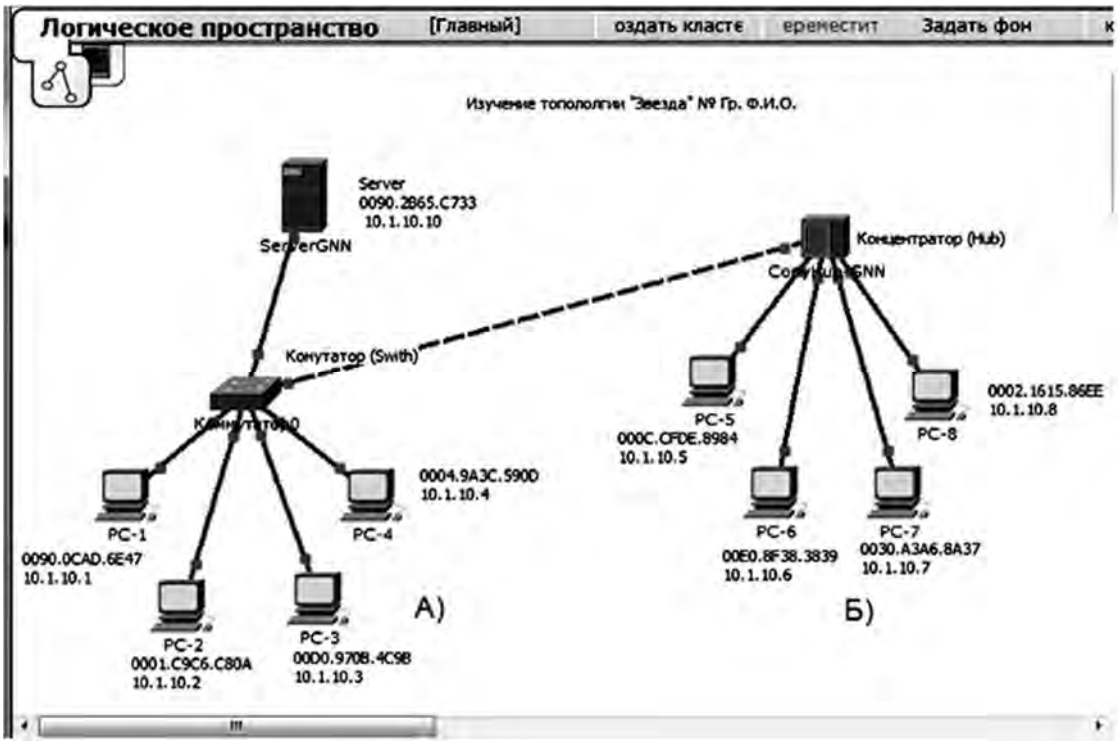


Рисунок 10.2 – Модель сети с использованием коммутатора и концентратора

3 В качестве вариантов следует выполнять следующее правило: при вводе IP-адресов для компьютеров PC-1…PC-N – 192.GGG.NN.1…192.GGG.NN.N, для сервера ввести IP-адрес: 192.GGG.NN.10, где GGG – номер группы (только цифры), NN – порядковый номер по журналу группы, маска 255.255.255.0.

4 Запустить программу Packet Tracer с созданной моделью сети в режиме симуляции, отправить пакеты с компьютеров на сервер. Сравнить трассы прохождения, объяснить причину.

5 Оформить отчет.

11 Лабораторная работа № 11. Кодирование информации в LAN на физическом уровне OSI

Цель работы: изучить способы физического и логического цифрового кодирования на физическом уровне при передаче данных в сетях.

Основные теоретические положения

При цифровом кодировании дискретной двоичной информации применяют потенциальные и импульсные коды. Потенциальные коды для представления двоичных данных используют только значение потенциала сигнала. Импульсные коды используют либо импульсы определенной полярности, либо часть импульса – перепад потенциала определенного направления.

Рассмотрим наиболее известные способы физического и логического цифрового кодирования.

Потенциальный код с инверсией при единице NRZI (Non Return to Zero with ones Inverted (NRZI))

При потенциальном кодировании с инверсией при единице (Non Return to Zero with ones Inverted (NRZI)) используется только два уровня сигнала (рисунок 11.1). При передаче нуля он передает потенциал, который был установлен в предыдущем такте (то есть не меняет его), а при передаче единицы потенциал инвертируется на противоположный. Часто используются в оптических кабелях, где устойчиво распознаются два состояния сигнала – свет и темнота, например, Base-FX.

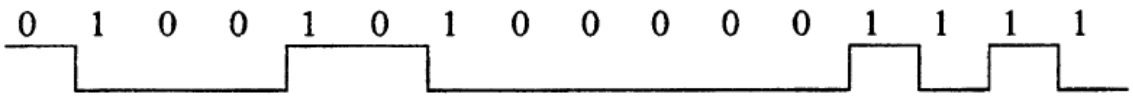


Рисунок 11.1 – Потенциальный код NRZI с инверсией при единице

Метод биполярного кодирования с альтернативной инверсией (AMI)

При биполярном кодировании с альтернативной инверсией (Bipolar Alternate Mark Inversion, AMI) используются три уровня потенциала – отрицательный, нулевой и положительный (рисунок 11.2).

Для кодирования логического нуля используется нулевой потенциал.

Логическая единица кодируется либо положительным потенциалом, либо

отрицательным, при этом потенциал каждой новой единицы противоположен потенциалу предыдущей.

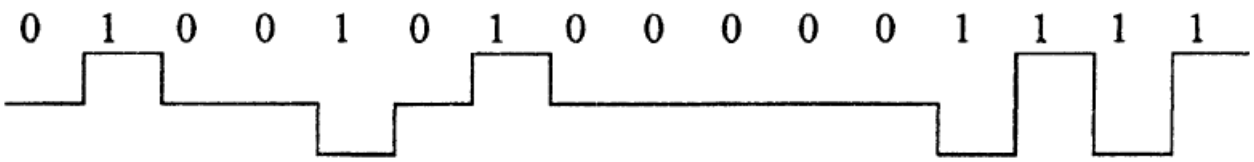


Рисунок 11.2 – Квазитроичное кодирование (AMI)

Код AMI частично ликвидирует проблемы постоянной составляющей и отсутствия самосинхронизации, присущие другим потенциальным кодам.

Потенциальный код 2B1Q

Название кода – **2B1Q** – отражает его суть – каждые два бита (2B) передаются за один такт (1) сигналом, имеющим четыре состояния (Q – Quadra). Это потенциальный код с четырьмя уровнями сигнала для кодирования данных. Каждые два бита (2B) передаются за один такт сигналом, имеющим четыре состояния (1Q):

1) паре бит 00 соответствует потенциал –2,5 В;
2) паре бит 01 соответствует потенциал –0,833 В;
3) паре бит 11 соответствует потенциал +0,833 В;
4) паре бит 10 соответствует потенциал +2,5 В.

Таким образом, при данном методе кодирования кодируется не каждый бит информационного исходного сигнала, а два бита (рисунок 11.3).

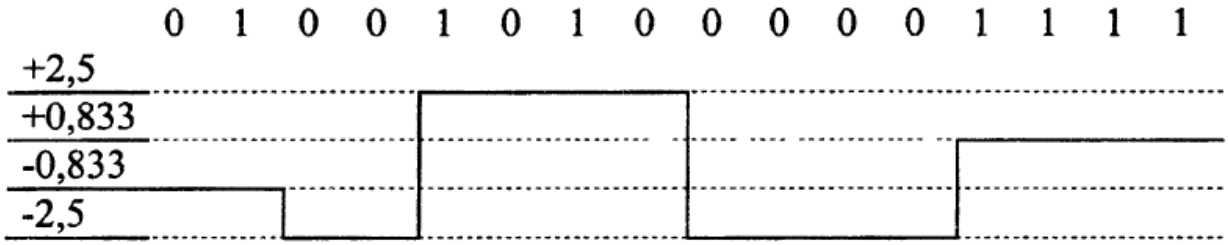


Рисунок 11.3 – Потенциальный код 2B1Q

С помощью кода 2B1Q можно по одной и той же линии передавать данные в 2 раза быстрее, чем с помощью кода AMI или NRZI.

Длительность такта при **2B1Q** увеличивается в 2 раза, что позволяет уменьшить спектр сигнала.

***Код MLT3* (*Multi Level Transmissi-3*)**

Используются три уровня линейного сигнала: «-1», «0», «+1». Логической единице соответствует обязательный переход с одного уровня сигнала на другой. При передаче логического нуля изменение уровня линейного сигнала не

происходит. Код MLT3, на первый взгляд, идентичен коду AMI. Разница состоит в том, что при кодировании «1» в AMI используются два уровня «-1» и «+1», в MLT3 – три уровня кодирования «-1», «0», «+1», при кодировании «0» в AMI используется «0», а при MLT3 – любой из трех «-1», «0», «+1», а при «0» в следующем такте уровень не меняется.

При передаче последовательности единиц период изменения уровня сигнала включает четыре бита. В этом случае $f_0 = N/4$ Гц. Это максимальная основная частота сигнала в коде MLT3. В случае чередующейся последовательности нулей и единиц основная гармоника сигнала находится на частоте $f_0 = N/8$ Гц (рисунок 11.4).

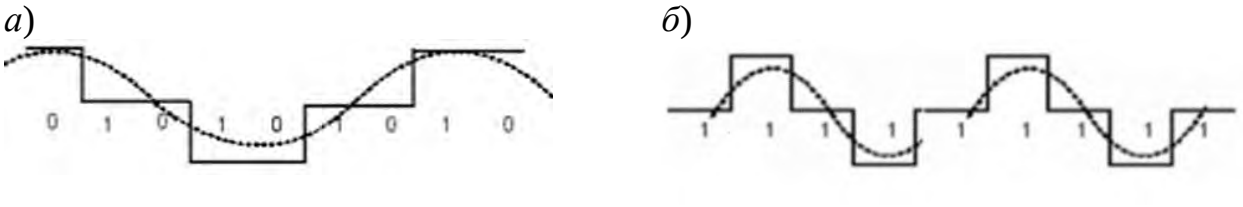


Рисунок 11.4 – Передача чередующихся единиц и нулей (*а*) и передача длинной последовательности единиц (*б*)

Импульсные коды. Манчестерский код

В локальных сетях до недавнего времени самым распространенным методом кодирования был *манчестерский код*. Он применяется в технологиях Ethernet и Token Ring. В манчестерском коде для кодирования единиц и нулей используется перепад потенциала, т. е. фронт импульса (рисунок 11.5).

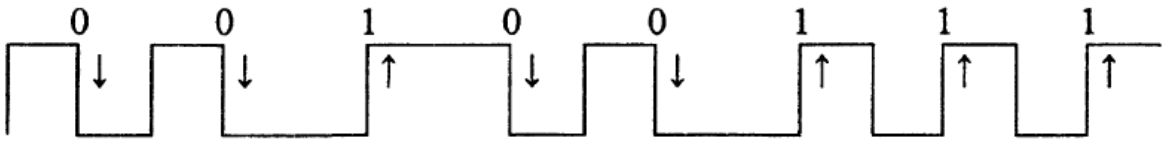


Рисунок 11.5 – Манчестерский код

Каждый такт делится на две части. Информация кодируется перепадами потенциала, происходящими в середине каждого такта.

Манчестерский код обладает хорошими самосинхронизирующими свойствами. У него также нет постоянной составляющей, а основная гармоника в худшем случае (при передаче последовательности единиц или нулей) имеет частоту N Гц, а в лучшем (при передаче чередующихся единиц и нулей) – $N/2$ Гц.

Логическое кодирование. Избыточные коды

Избыточные коды основаны на разбиении исходной последовательности бит на части, которые часто называют символами. Затем каждый исходный символ заменяется на новый, который имеет большее количество бит, чем исходный.

Логический код 4B/5B заменяет исходные символы длиной в 4 бита на символы длиной в 5 бит. Так как результирующие символы содержат избыточные биты, то общее количество битовых комбинаций в них больше, чем в исходных. Результирующие символы содержат 32 битовые комбинации, а исходные символы только 16. Поэтому в результирующем коде можно отобрать 16 таких комбинаций, которые не содержат большого количества нулей, а остальные считать запрещенными кодами (code violation) (таблица 11.1). Кроме устранения постоянной составляющей и придания коду свойства самосинхронизации, избыточные коды позволяют приемнику распознавать искаженные биты.

Таблица 11.1 – Соответствие исходных и результирующих кодов 4B/5B

Исходный код	Результативный код	Исходный код	Результативный код
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

Код 4B/5B затем передается по линии с помощью физического кодирования по одному из методов потенциального кодирования (потенциальных кодов типа AMI, NRZI или 2Q1B).

Логическое кодирование. Скремблирование.

Скремблирование заключается в побитном вычислении результирующего кода на основании битов исходного кода и полученных в предыдущих тактах битов результирующего кода. Например, скремблер может реализовывать следующее соотношение:

$$B_i = A_i B_{i-3} B_{i-5}.$$

В данном примере функции:

– B_i – бит, полученный на i-м такте работы скремблера;

– A_i – бит, поступающий на i-м такте на вход скремблера;

– $B_{i\text{-}3}$ и $B_{i\text{-}5}$ – биты результирующего кода, полученные на предыдущих тактах работы скремблера и объединенные операцией исключающего «ИЛИ».

После получения результирующей последовательности приемник передает ее дескремблеру, который восстанавливает исходную последовательность на основании обратного соотношения:

$$C_i = (A_i \oplus B_{i-3} \oplus B_{i-5}) \oplus B_{i-3} \oplus B_{i-5} = A_i \oplus B_{i-3} \oplus B_{i-5} \oplus B_{i-3} \oplus B_{i-5}.$$

Скремблирование кодами B8ZS и HDB3

Логическое кодирование B8ZS (Bipolar with 8-Zeros Substitution) и HDB3 (High-Density Bipolar 3-Zeros) используются для улучшения кода Bipolar AMI. Они основаны на искусственном искажении последовательности нулей запрещенными символами (рисунок 11.6).

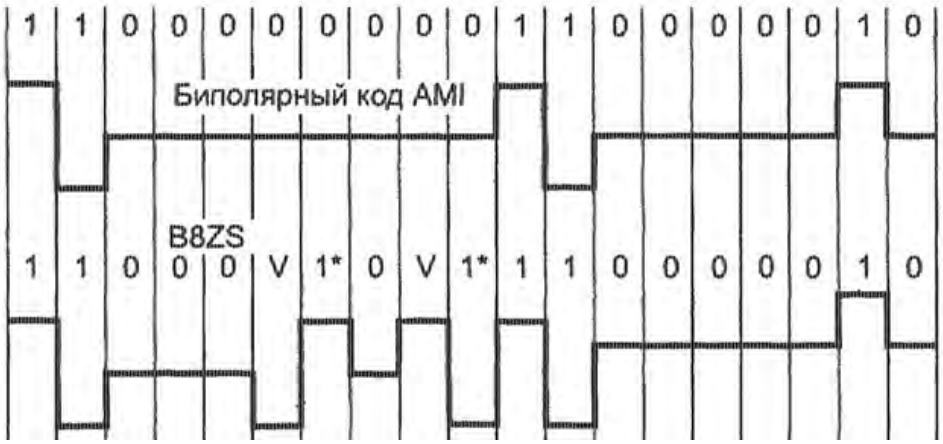


Рисунок 11.6 – Логическое кодирование (скремблирование) кодом B8ZS

Код HDB3 исправляет любые четыре смежных нуля в исходной последовательности. Правила формирования кода HDB3 более сложные, чем кода B8ZS. Каждые четыре нуля заменяются четырьмя сигналами, в которых имеется один сигнал V. Для подавления постоянной составляющей полярность сигнала V чередуется при последовательных заменах. Кроме того, для замены используются два образца четырех тактовых кодов. Если перед заменой исходный код содержал нечетное число единиц, то задействуется последовательность 000V, а если число единиц было четным – последовательность 1*00V (рисунок 11.7).

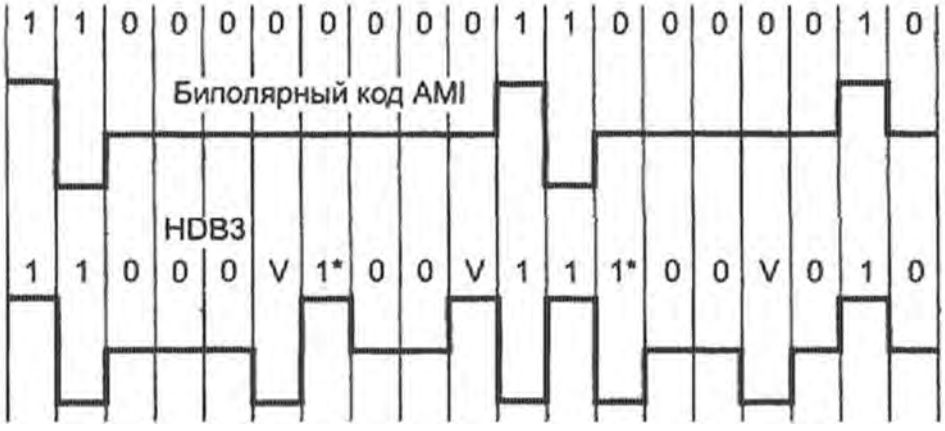


Рисунок 11.7 – Логическое кодирование (скремблирование) кодом HDB3

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Выполнить задание: создать программу, написанную на языке программирования высокого уровня C#, C++, VBA и т. п., позволяющую конвертировать последовательность 8–16 символов в физический или логический код в соответствии с вариантом задания (таблица 11.2).

3 Сделать выводы по результатам исследований.

4 Оформить отчет.

Таблица 11.2 – Варианты для выполнения заданий

Номер варианта	Тип кодирования	Номер варианта	Тип кодирования	Номер варианта	Тип кодирования	Номер варианта	Тип кодирования
1	NRZ и 2B1Q	5	4B/5B и AMI	9	NRZI и MLT3	13	2B1Q и AMI
2	AMI и NRZI	6	B8ZS и AMI	10	Скрм и NRZI	14	NRZ и NRZI
3	NRZ и AMI	7	HDB3 и AMI	11	NRZI и Скрм	15	2B1Q и NRZI
4	2B1Q; MLT3	8	4B/5B и MLT3	12	AMI и MLT3	16	Манч и MLT3

12 Лабораторная работа № 12. Изучение протоколов доступа к среде передачи

Цель работы: изучить протоколы доступа к среде передачи LAN, метод случайного доступа, детерминированный доступ, канальный уровень, подуровни LLC и MAC, MAC-адрес, формат кадра Ethernet.

Основные теоретические положения

Существуют различные методы, способы и порядок доступа к физической среде, обеспечивающей передачу дискретной информации между компьютерами. По **порядку доступа** среда передачи может быть **индивидуальной** или **разделяемой**.

Разделяемой средой передачи (shared media) называется линия, которая используется попеременно несколькими (более чем двумя) устройствами, подключенными к ней. Индивидуальной средой передачи является линия связи, к каждому окончанию которой подключено только одно устройство.

Примерами связи компьютеров посредством разделяемой среды являются системы с топологиями общая Шина и Кольцо, а также сети Wi-Fi. Примером связи компьютеров с индивидуальной средой является топология сетей LAN – Звезда, а также соединение двух компьютеров точка-точка.

Существуют два основных метода доступа к разделяемой физической среде:

1) метод случайного доступа;

2) детерминированный доступ.

Метод случайного доступа является одним из основных методов захвата разделяемой среды. Он основан на том, что узел, у которого есть кадр для передачи, пытается его отправить без какой бы то ни было предварительной процедуры согласования времени использования разделяемой среды с другими узлами сети.

Детерминированный доступ – это другой популярный подход к обеспечению доступа к разделяемой среде. Он получил свое название благодаря тому, что максимальное время ожидания доступа к среде всегда известно. Алгоритмы детерминированного доступа используют два механизма – **передачу токена и опрос**.

Для случайного доступа применяются два типа алгоритма – **CSMA/CA** и **CSMA/CD.**

CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) – множественный доступ с прослушиванием несущей и избеганием коллизий. Узел, готовый послать кадр, прослушивает линию. При отсутствии несущей он посылает короткий сигнал запроса на передачу (RTS) и определенное время ожидает ответа (CTS) от адресата назначения. При отсутствии ответа (подразумевается возможность коллизии) попытка передачи откладывается, при получении ответа в линию посылается кадр. Метод не позволяет полностью избежать коллизий, но они обрабатываются на вышестоящих уровнях протокола.

CSMA/CD (Carrier Sense Multiple Access/Collision Detect) – множественный доступ с прослушиванием несущей и обнаружением коллизий. Узел, готовый послать кадр, прослушивает линию. При отсутствии несущей он начинает передачу кадра, одновременно контролируя состояние линии. При обнаружении коллизии передача прекращается, и повторная попытка откладывается на случайное время. Коллизии – нормальное, хотя и не очень частое явление для CSMA/CD. Их частота связана с количеством и активностью подключенных узлов.

В современных компьютерных локальных сетях (LAN) применяются стандарты Ethernet, которые широко используют **метод случайного доступа – CSMA/CD (Carrier Sense Multiple Access/Collision Detect)**.

Упрощенный алгоритм метода **CSMA/CD** представлен на рисунке 12.1.

1 Для передачи кадра интерфейс-отправитель должен убедиться, что разделяемая среда свободна.

2 Если канал занят – рабочая станция ждет.

3 Если канал свободен – рабочая станция начинает передачу и одновременно прослушивает канал.

4 При возникновении коллизии (столкновении сигналов соседних станций) рабочая станция прекращает передачу и ждет случайно выбранный промежуток времени $T = L$ (интервал отсрочки). Интервал отсрочки равен 512 битовым интервалам (для скорости 10 Мбит/с – 0,1 мкс). L – целое число, выбранное с рав-

ной вероятностью из диапазона [0, 2*N*], где *N* – номер повторной попытки передачи данного кадра: 1, 2, …, 10.

5 При отсутствии коллизии рабочая станция продолжает передачу до конца кадра.

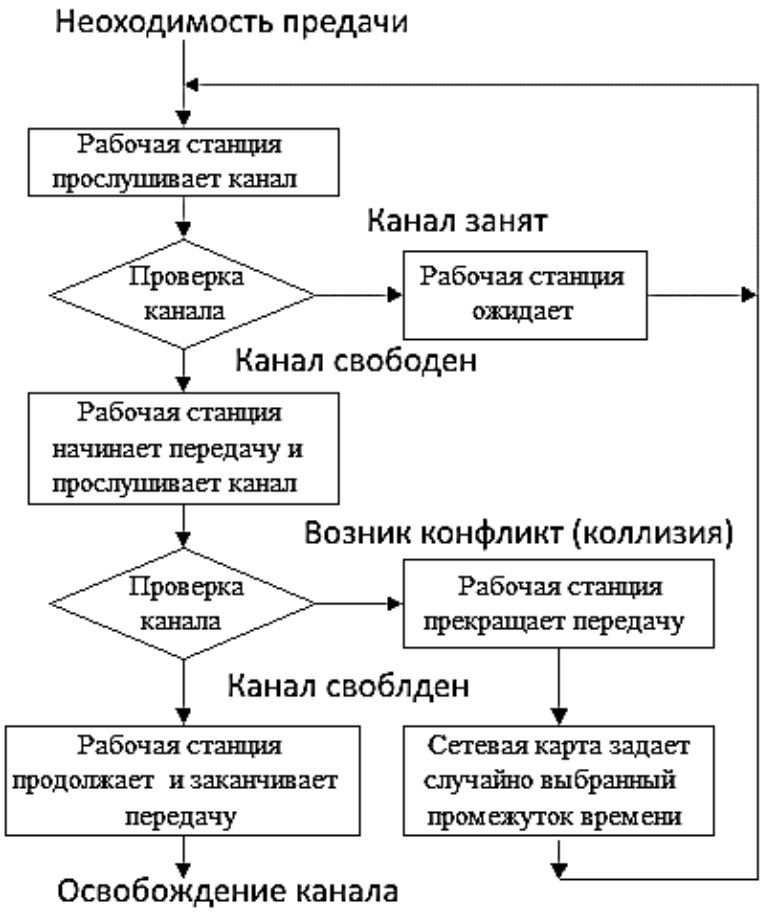


Рисунок 12.1 – Упрощенный алгоритм метода доступа **CSMA/CD**

Метод управления обменом **CSMA/CD** используется как в обычных сетях типа Ethernet, так и в высокоскоростных сетях (Fast Ethernet, Gigabit Ethernet). Но с применением коммутаторов в сети, дуплексного режима и особенно построения сегментов сети Ethernet высоких скоростей по топологии звезда необходимость в данном методе отпала. В стандарте 10 Gigsbit Ethernet (10 GE) он уже не применяется.

Форматы кадров технологии Ethernet

В сетях Ethernet на канальном уровне используются кадры 4-х различных форматов (типов). На практике чаще всего применяется кадр, который называется Ethernet DIX, или Ethernet II (рисунок 12.2).

Кадр – Ethernet DIX (Ethernet II) – представляет собой структуру из пяти полей: DA (Destination Address) – MAC-адрес назначения, SA (Source Address) –

MAC-адрес источника, Данные – поле данных, FCS (Frame Check Sequence) – поле контрольной последовательности кадра.

Кадр Ethernet DIX (II)				
6	6	2	46-1500	4
DA	SA	T	Данные	FCS

Рисунок 12.2 – Формат кадра Ethernet DIX или Ethernet II

Кадр Ethernet формируется подуровнем MAC канального уровня и широко применяется в современных локальных вычислительных сетях.

Детерминированные методы доступа

TPMA (Token Passing Multiple Access) – множественный доступ с передачей полномочия, или метод с передачей маркера.

Метод с передачей маркера – это метод доступа к среде, в котором от рабочей станции к рабочей станции передается маркер, дающий разрешение на передачу сообщения. При получении маркера рабочая станция может передавать сообщение, присоединяя его к маркеру, который переносит это сообщение по сети. Каждая станция видит это сообщение, но только станция-адресат принимает его. При этом она создает новый маркер. Широко применяется в сетях с топологией кольцо.

TDMA (Time Division Multiple Access) – множественный доступ с разделением во времени, основанный на распределении времени работы канала между системами (рисунок 12.3).

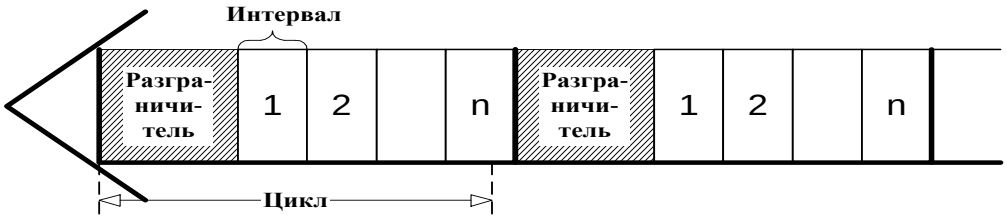


Рисунок 12.3 – Метод доступа с разделением во времени

Доступ TDMA основан на использовании специального устройства, называемого тактовым генератором. Этот генератор делит время канала на повторяющиеся циклы. Каждый из циклов начинается сигналом-разграничителем (сигнал синхронизации).

Цикл включает *n* (обычно 30) пронумерованных временных интервалов, называемых ячейками. Интервалы предоставляются для загрузки в них блоков данных. Данный способ позволяет организовать передачу данных с коммутацией пакетов и с коммутацией каналов.

FDMA (Frequency Division Multiple Access) или множественный доступ с разделением длины волны **(Wavelength Division Multiple Access – WDMA)**. Доступ *FDMA* основан на разделении полосы пропускания канала на группу полос частот (рисунок 12.4), образующих *логические каналы*. Широкая полоса канала делится на ряд узких полос, разделенных защитными полосами.

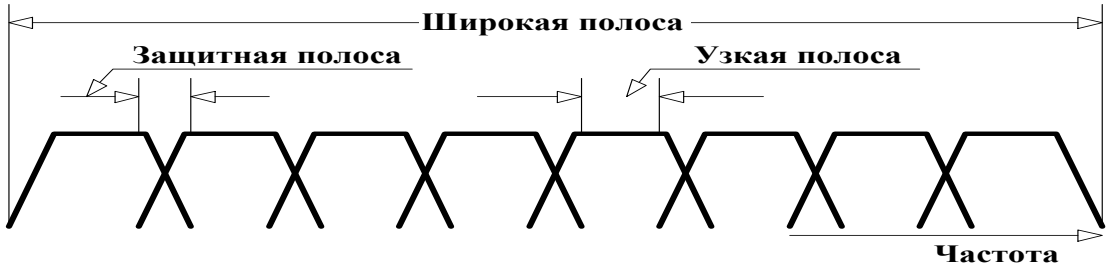


Рисунок 12.4 – Метод доступа **FDMA.** Схема выделения логических каналов

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Определить максимальное количество кадров минимальной и максимальной длины, проходящих по стандартной сети Ethernet –10 Base-5.

3 Определить максимальную пропускную способность сегмента 10 Base 5 в Мбит/с для кадров максимальной, усредненной и минимальной длины.

4 Сделать выводы по результатам исследований.

5 Оформить отчет.

Список литературы

1 **Бройдо, В. Л.** Архитектура ЭВМ и систем: учебник для вузов / В. Л. Бройдо, О. П. Ильина. – Санкт-Петербург: Питер, 2009. – 720 с.

2 **Новиков, В. А**. Информационные системы и сети: учебное пособие с электронным приложением / В. А. Новиков, А. В. Новиков, В. В. Матвеенко. – Минск: Изд-во Гревцова, 2014. – 448 с.

3 **Олифер, В. Г.** Компьютерные сети. Принципы, технологии, протоколы: учебник / В. Г. Олифер, Н. А. Олифер. – 5-е изд. – Санкт-Петербург: Питер, 2016. – 992 с.

4 **Пескова, С. А.** Сети и телекоммуникации : учебное пособие / С. А. Пескова, А. В. Кузин, А. Н. Волков. – 2-е изд., стер. – Москва: Академия, 2007. – 352 с.

5 **Бройдо, В. Л**. Вычислительные системы, сети и телекоммуникации: учебное пособие / В. Л. Бройдо, О. П. Ильина. – 4-е изд. – Санкт-Петербург: Питер, 2011. – 560 с.