

Министерство образования и науки Российской Федерации
**Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Тамбовский государственный технический университет»**

П.В. БАЛАБАНОВ, А.Е. БОЯРИНОВ, А.П. САВЕНКОВ

**ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И
СЕТИ В ЗАДАЧАХ
УПРАВЛЕНИЯ КАЧЕСТВОМ**

*Рекомендовано Учёным советом университета
в качестве практикума по дисциплине
«Сети электронно-вычислительных машин и средства коммуникаций»
для студентов очной формы обучения направления подготовки
бакалавров 221400 «Управление качеством»*



Тамбов
Издательство ФГБОУ ВПО «ТГТУ»
2012

УДК 004(076)
ББК *з*973я73-5
Б20

Рецензенты:

Доктор технических наук, профессор кафедры
«Управление качеством и сертификация» ФГБОУ ВПО «ТГТУ»
М.М. Мордасов

Кандидат технических наук, доцент кафедры общей физики
ФГБОУ ВПО «ТГУ им. Г.Р. Державина»
В.В. Штейнбрехер

Балабанов, П.В.

Б20 Вычислительная техника и сети в задачах управления качеством : практикум / П.В. Балабанов, А.Е. Бояринов, А.П. Савенков. – Тамбов : Изд-во ФГБОУ ВПО «ТГТУ», 2012. – 92 с. – 70 экз. – ISBN 978-5-8265-1116-9.

Даны методические указания для практических занятий по дисциплине «Сети электронно-вычислительных машин и средства коммуникаций». Рассмотрены задачи, связанные с применением вычислительной техники и сетей в системах контроля и управления качеством продукции, процессов и услуг.

Предназначен для студентов очной формы обучения направления подготовки бакалавров 221400 «Управление качеством». Будет полезен студентам специальности 200503 «Стандартизация и сертификация» при изучении дисциплины «Микропроцессоры в системах контроля», а также студентам других специальностей и направлений подготовки при изучении дисциплины «Вычислительные машины и сети». Разработан с учётом федеральных государственных образовательных стандартов.

УДК 004(076)
ББК *з*973я73-5

ISBN 978-5-8265-1116-9

© Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Тамбовский государственный технический университет» (ФГБОУ ВПО «ТГТУ»), 2012

ВВЕДЕНИЕ

Область профессиональной деятельности бакалавров по направлению подготовки 221400 «Управление качеством» включает разработку, исследование, внедрение и сопровождение в организациях всех видов деятельности и всех форм собственности систем управления качеством, охватывающих все процессы организации и направленных на достижение долговременного успеха и стабильности её функционирования.

В современных условиях любая деятельность сопровождается использованием вычислительной техники. Задания настоящего практикума затрагивают следующие аспекты практического использования вычислительной техники бакалаврами:

- грамотное использование офисной техники и локальной сети организации;
- оценку целесообразности внедрения и разработки, надёжности, стоимости, рациональности и сроков использования офисной техники и сетей организации;
- разработку приборов контроля параметров технологических процессов и качества продукции, автоматизированных систем контроля и управления технологическими процессами с использованием современной вычислительной техники.

Рассмотрено 16 практических работ. Приведены цели, задания, методические указания по выполнению работ и списки контрольных вопросов. Дан список рекомендуемой литературы.

Первая часть практических работ выполняется с целью повышения компьютерной грамотности студентов бакалавриата. Изучаются аппаратные и программные средства современных компьютеров, сетевые технологии. Предусмотрено выполнение заданий по поиску неисправностей в сетях, их проектированию и защите информации.

Вторая часть работ посвящена программированию промышленных контроллеров по последовательному интерфейсу. Контроллеры применяются в автоматизированных системах контроля и управления технологическими процессами. Изучаются интегрированная среда программирования, порты ввода-вывода, таймеры/счётчики, система прерываний и блок аналого-цифрового преобразователя.

ПРАКТИЧЕСКИЕ РАБОТЫ

Практическая работа 1

ТЕСТИРОВАНИЕ АППАРАТНЫХ СРЕДСТВ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ

Цель работы: научиться определять основные технические характеристики аппаратных средств современного персонального компьютера.

Задание

1. Изучите методические указания и рекомендуемую литературу.
2. При помощи программы «Everest» соберите данные о технических характеристиках персонального компьютера.

Методические указания

К основным составным частям персонального компьютера относят: системный блок, монитор, клавиатуру, мышь. В свою очередь, системный блок состоит из материнской (системной) платы, на которой установлены центральный процессор, оперативная память, платы расширения (звуковой адаптер, видеоадаптер, сетевой адаптер и т.д.); блока питания, жёсткого диска, оптического привода и др.

Рассмотрим состав, назначение и характеристики некоторых основных компонентов системного блока.

Системная плата (рис. 1.1) – это сложная многослойная печатная плата, на которой устанавливаются основные компоненты персонального компьютера (центральный процессор ЦП, контроллер оперативного запоминающего устройства (ОЗУ) и собственно ОЗУ, загрузочное постоянное запоминающее устройство (ПЗУ), контроллеры базовых интерфейсов ввода-вывода).

Быстродействие различных компонентов компьютера (процессора, оперативной памяти и контроллеров периферийных устройств) может существенно различаться. Для согласования быстродействия на системной плате устанавливаются специальные микросхемы (чипсеты), включающие в себя контроллер оперативной памяти (так называемый северный мост) и контроллер периферийных устройств (южный мост).

Северный мост обеспечивает обмен информацией между процессором и оперативной памятью по системной шине. В процессоре используется внутреннее умножение частоты, поэтому частота процессора в несколько раз больше, чем частота системной шины. В современных компьютерах частота процессора может превышать частоту

системной шины в 10 раз (например, частота процессора 1 ГГц, а частота шины – 100 МГц).

К северному мосту подключается шина PCI (Peripheral Component Interconnect Bus – шина взаимодействия периферийных устройств), которая обеспечивает обмен информацией с контроллерами периферийных устройств. Частота контроллеров меньше частоты системной шины, например, если частота системной шины составляет 100 МГц, то частота шины PCI обычно в три раза меньше – 33 МГц. Контроллеры периферийных устройств (звуковая плата, сетевая плата, SCSI-контроллер, внутренний модем) устанавливаются в слоты расширения системной платы.

По мере увеличения разрешающей способности монитора и глубины цвета требования к быстродействию шины, связывающей видеоплату с процессором и оперативной памятью, возрастают. На рисунке 1.1 для подключения видеоплаты используется специальная шина AGP (Accelerated Graphic Port – ускоренный графический порт), соединённая с северным мостом и имеющая частоту, в несколько раз большую, чем шина PCI.

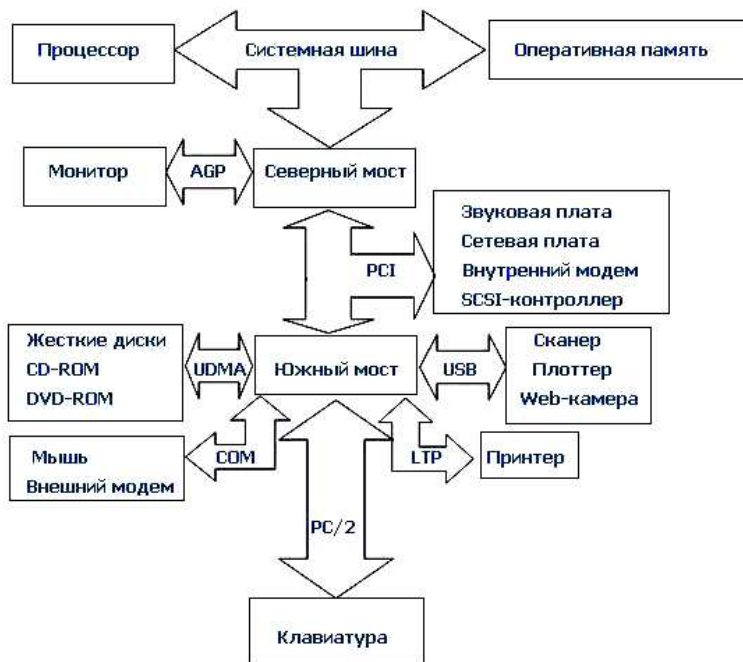


Рис. 1.1. Логическая схема системной платы

Южный мост обеспечивает обмен информацией между северным мостом и портами для подключения периферийного оборудования.

Устройства хранения информации (жёсткие диски, CD-ROM, DVD-ROM) подключаются к южному мосту по шине UDMA (Ultra Direct Memory Access – прямое подключение к памяти).

Мышь и внешний модем подключаются к южному мосту с помощью последовательных портов, которые передают электрические импульсы, несущие информацию в машинном коде, последовательно один за другим. Обозначаются последовательные порты как COM1 и COM2, а аппаратно реализуются с помощью 25-контактного и 9-контактного разъёмов, которые выведены на заднюю панель системного блока.

Принтер подключается к параллельному порту, который обеспечивает более высокую скорость передачи информации, чем последовательные порты, так как передаёт одновременно 8 электрических импульсов, несущих информацию в машинном коде. Обозначается параллельный порт как LTP, а аппаратно реализуется в виде 25-контактного разъёма на задней панели системного блока.

Для подключения сканеров и цифровых камер обычно используется порт USB (Universal Serial Bus – универсальная последовательная шина), который обеспечивает высокоскоростное подключение к компьютеру сразу нескольких периферийных устройств. Клавиатура подключается обычно с помощью порта PS/2.

Системные платы классифицируются по форм-фактору.

Форм-фактор материнской платы – стандарт, определяющий размеры материнской платы для персонального компьютера, места её крепления к корпусу, расположение на ней интерфейсов шин, портов ввода-вывода, сокета центрального процессора и слотов для оперативной памяти, а также тип разъёма для подключения блока питания.

Примерами обозначения форм-факторов могут быть:

- Устаревшие: Baby-AT; Mini-ATX; полноразмерная плата AT; LPX.
- Современные: ATX; microATX; Flex-ATX; NLX; WTX, СЕВ.
- Внедряемые: Mini-ITX и Nano-ITX; Pico-ITX; ВТХ, MicroВТХ и PicoВТХ.

Центральный процессор – центральное вычислительное устройство, исполнитель машинных инструкций, часть аппаратного обеспечения компьютера или программируемого логического контроллера, отвечающая за выполнение операций, заданных программой.

В настоящее время используются многоядерные процессоры на одном или нескольких кристаллах.

Для повышения быстродействия используют так называемое кэширование. Кэширование – это использование дополнительной быст-

родействующей памяти (кэш-памяти) для хранения копий блоков информации из основной (оперативной) памяти, вероятность обращения к которым в ближайшее время велика.

Оперативная память – это рабочая область для процессора компьютера. В ней во время работы хранятся программы и данные. Оперативная память часто рассматривается как временное хранилище, потому что данные и программы в ней сохраняются только при включённом компьютере или до нажатия кнопки сброса (reset). Перед выключением или нажатием кнопки сброса все данные, подвергнутые изменениям во время работы, необходимо сохранить на запоминающем устройстве, которое может хранить информацию постоянно (обычно это жёсткий диск). При новом включении питания сохранённая информация вновь может быть загружена в память.

Устройства оперативной памяти иногда называют запоминающими устройствами с произвольным доступом. Это означает, что обращение к данным, хранящимся в оперативной памяти, не зависит от порядка их расположения в ней.

В современных компьютерах используются запоминающие устройства трёх основных типов.

– **ROM (Read Only Memory)**. Постоянное запоминающее устройство, не способное выполнять операцию записи данных.

– **DRAM (Dynamic Random Access Memory)**. Динамическое запоминающее устройство с произвольным порядком выборки.

– **SRAM (Static RAM)**. Статическое ОЗУ.

В памяти типа ROM данные можно только хранить. Именно поэтому такая память используется только для чтения данных. ROM также часто называется *энергонезависимой памятью*, потому что любые данные, записанные в неё, сохраняются при выключении питания. Поэтому в ROM помещаются команды запуска персонального компьютера, т.е. программное обеспечение, которое загружает систему.

Основной код BIOS содержится в микросхеме ROM на системной плате, но на платах адаптеров также имеются аналогичные микросхемы. Они содержат вспомогательные подпрограммы базовой системы ввода-вывода и драйверы, необходимые для конкретной платы, особенно для тех плат, которые должны быть активизированы на раннем этапе начальной загрузки, например видеоадаптер.

Динамическая оперативная память DRAM используется в большинстве систем оперативной памяти современных персональных компьютеров. Основное преимущество памяти этого типа состоит в том, что её ячейки упакованы очень плотно, т.е. в небольшую микросхему можно упаковать много битов, а значит, на их основе можно построить память большой ёмкости.

Ячейки памяти в микросхеме DRAM – это крошечные конденсаторы, которые удерживают заряды. Именно так (наличием или отсутствием зарядов) и кодируются биты. Проблемы, связанные с памятью этого типа, вызваны тем, что она динамическая, т.е. должна постоянно регенерироваться, так как в противном случае электрические заряды в конденсаторах памяти будут «стекают» и данные будут потеряны.

Статическая оперативная память SRAM названа так потому, что, в отличие от динамической оперативной памяти, для сохранения её содержимого не требуется периодической регенерации. Но это не единственное её преимущество. SRAM имеет более высокое быстродействие, чем динамическая оперативная память, и может работать на той же частоте, что и современные процессоры. Кроме того, SRAM потребляют гораздо меньше энергии, чем DRAM, и могут применяться в устройствах, работающих от автономных источников (аккумуляторов или батарей).

Однако для хранения каждого бита в конструкции SRAM используется кластер из шести транзисторов. Отсутствие необходимости в регенерации обусловлено схемой-триггером, элементом, который может находиться только в одном из двух устойчивых состояний (0 или 1). Почему же тогда микросхемы SRAM не используются для всей системной памяти? Ответ прост. По сравнению с динамической оперативной памятью быстродействие SRAM намного выше, но плотность её гораздо ниже, а цена довольно высока. Более низкая плотность означает, что микросхемы SRAM имеют большие габариты, хотя их информационная ёмкость намного меньше. Всё это не позволяет использовать память типа SRAM в качестве оперативной памяти в персональных компьютерах.

Жёсткий диск – энергонезависимое перезаписываемое компьютерное запоминающее устройство. Является основным накопителем данных практически во всех компьютерах. Жёсткий диск состоит из гермозоны и блока электроники.

Гермозона включает в себя корпус из прочного сплава, собственно диски (пластины) с магнитным покрытием, блок головок с устройством позиционирования, электропривод шпинделя.

Блок головок – пакет рычагов из пружинистой стали (по паре на каждый диск). Одним концом они закреплены на оси рядом с краем диска. На других концах (над дисками) закреплены головки.

Диски (пластины), как правило, изготовлены из металлического сплава. Хотя были попытки делать их из пластика и даже стекла, но такие пластины оказались хрупкими и недолговечными. Обе плоскости пластин, подобно магнитофонной ленте, покрыты тончайшей пылью ферромагнетика – окислов железа, марганца и других металлов. Точный состав и технология нанесения держатся в секрете.

Диски жёстко закреплены на шпинделе. Во время работы шпиндель вращается со скоростью несколько тысяч оборотов в минуту (4200, 5400, 7200, 10 000, 15 000). При такой скорости вблизи поверхности пластины создаётся мощный воздушный поток, который приподнимает головки и заставляет их парить над поверхностью пластины. Форма головок рассчитывается так, чтобы при работе обеспечить оптимальное расстояние от пластины. Пока диски не разогнались до скорости, необходимой для «взлёта» головок, парковочное устройство удерживает головки в зоне парковки. Это предотвращает повреждение головок и рабочей поверхности пластин.

Устройство позиционирования головок состоит из неподвижной пары сильных, как правило неодимовых, постоянных магнитов и катушки на подвижном блоке головок.

Блок электроники обычно содержит: управляющий блок, постоянное запоминающее устройство, буферную память, интерфейсный блок и блок цифровой обработки сигнала.

Интерфейсный блок обеспечивает сопряжение электроники жёсткого диска с остальной системой.

Блок управления представляет собой систему управления, принимающую электрические сигналы позиционирования головок и вырабатывающую управляющие воздействия приводом типа «звуковая катушка», коммутации информационных потоков с различных головок, управления работой всех остальных узлов (к примеру, управление скоростью вращения шпинделя).

Блок ПЗУ хранит управляющие программы для блоков управления и цифровой обработки сигнала, а также служебную информацию винчестера.

Буферная память сглаживает разницу скоростей интерфейсной части и накопителя (используется быстродействующая статическая память). Увеличение размера буферной памяти в некоторых случаях позволяет увеличить скорость работы накопителя.

Блок цифровой обработки сигнала осуществляет очистку считанного аналогового сигнала и его декодирование (извлечение цифровой информации).

Сетевая плата (адаптер) – периферийное устройство, позволяющее компьютеру взаимодействовать с другими устройствами сети.

Звуковая карта – плата, которая позволяет работать со звуком на компьютере. В настоящее время звуковые карты бывают встроенными в материнскую плату, отдельными платами расширения, внешними устройствами.

Видеокарта (видеоадаптер) – устройство, преобразующее изображение, находящееся в памяти компьютера, в видеосигнал для монитора.

Более полную информацию по компонентам компьютера, и в частности системного блока, можно найти на сайтах сети Интернет и в книгах [1 – 4].

Порядок выполнения работы

1. Запустите программу Everest на тестируемом компьютере и с помощью мастера отчётов (меню «Отчёт») сформируйте отчёт об аппаратном обеспечении.

2. Заполните табл. 1.1.

1.1. Результаты выполнения работы

№	Наименование компонента системного блока или характеристика	Найденное обозначение или характеристика
1	Тип ЦП, частота	
2	Тип системной платы, форм-фактор	
3	Чипсет системной платы	
4	Тип жёсткого диска, объём	
5	Тип сетевого адаптера	
6	Тип видеоадаптера	
7	Тип звукового адаптера	
8	Разъёмы ОЗУ	
9	Разъёмы расширения системной платы	
10	Объём кэш-памяти процессора	

Контрольные вопросы

1. Назначение и компоненты системной платы.
2. Что такое северный мост? Его назначение.
3. Что такое южный мост? Его назначение.
4. Что такое форм-фактор материнской платы?
5. Назначение центрального процессора.
6. Что такое многоядерный процессор?
7. Что такое кэширование?
8. Оперативное запоминающее устройство. Его назначение.
9. Что такое энергозависимые и энергонезависимые запоминающие устройства?
10. Универсальная последовательная шина USB.
11. Шина ввода-вывода PCI и PCI-Express.
12. Шина AGP.
13. Видеокарта. Назначение и устройство.
14. Сетевой адаптер. Назначение, типы, параметры и функции.
15. Назначение и типы оптических приводов.
16. Жёсткий диск. Назначение и устройство.

СЕТЕВЫЕ ТОПОЛОГИИ

Цель работы: изучить правила организации физического расположения в пространстве компьютеров, объединённых в сеть.

Задание

1. Подготовьте доклад с презентацией на одну из тем, приведённых ниже. Тему утвердите у преподавателя.
2. Выполните практическое задание.

Список тем доклада:

1. Базовые сетевые топологии. Шина. Преимущества и недостатки.
2. Базовые сетевые топологии. Кольцо. Преимущества и недостатки.
3. Базовые сетевые топологии. Звезда. Преимущества и недостатки.
4. Топология «дерево».
5. Сеть с сетчатой топологией.
6. Доступ к среде передачи.

Практическое задание

Вам поручено установить сеть для небольшой, но развивающейся компании, занимающей половину этажа. В состав компании входят директор, управляющий, администратор и пять сотрудников (рис. 2.1). Планируется принять на работу ещё двух сотрудников. У каждого сотрудника есть компьютер. Если необходимо обменяться информацией, приходится делать это устно или с использованием съёмных носителей, что неудобно. Лазерный принтер имеется у администратора. У каждого сотрудника имеется сканер. Какую топологию Вы предложите для компании? Оцените суммарную длину кабеля в каждом из предложенных случаев и выберите оптимальный вариант.

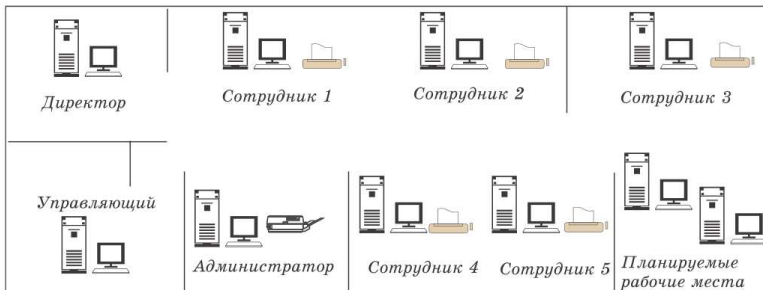


Рис. 2.1. План размещения рабочих мест сотрудников компании

Контрольные вопросы

1. Нарисуйте схему сети, построенной по топологии типа шина. Сеть должна включать 5 компьютеров.
2. Имеется 3 компьютера, расположенных на расстоянии 200 м друг от друга. Какую топологию вы выберете для создания сети?
3. Имеется комната площадью 20 кв. м. В ней необходимо поставить 10 компьютеров, объединённых сетью. Нарисуйте схему сети.
4. Нарисуйте схему сети, построенной по топологии типа звезда. Сеть должна включать 5 компьютеров.
5. В организации имеется 3 отдела. В каждом отделе по 8 компьютеров. Все отделы расположены на одном этаже здания. Нарисуйте схему сети.

Практическая работа 3

ЛИНИИ СВЯЗИ

Цель работы: изучить типовые линии связи, применяемые в компьютерных сетях.

Задание

1. Подготовьте доклад с презентацией на одну из тем, приведённых ниже.
2. Выполните практическое задание.

Список тем доклада:

1. Коаксиальный кабель (тонкий и толстый). Терминаторы. Примеры сетей на тонком и толстом коаксиальном кабеле.
2. Витая пара. Категории. Разводка проводников в коннекторах RJ-45.
3. Оптоволоконные кабели. Коннекторы для оптоволоконных кабелей.
4. Беспроводные линии.

Практическое задание

Для ранее разработанной сети (см. практическую работу 2) составить проект прокладки кабеля витая пара категории 5 в кабельных каналах согласно выбранной Вами сетевой топологии.

Контрольные вопросы

1. В чём отличия тонкого и толстого коаксиального кабеля?
2. Зачем нужны терминаторы?
3. Что такое витая пара?
4. Зачем необходимо экранировать витую пару?
5. В чём преимущество оптоволоконного кабеля по сравнению с витой парой?

АДРЕСАЦИЯ В КОМПЬЮТЕРНОЙ СЕТИ

Цель работы: изучить способы адресации в IP-сетях.

Задание

1. Изучите методические указания и рекомендуемую литературу.
2. Начертите схему локальной сети компьютерного класса.

Методические указания

В компьютерной сети передача информации от одного узла (компьютера) к другому узлу осуществляется посредством пакетов. Пакет можно рассматривать как набор битов служебной информации и информации, подлежащей передаче. К служебной информации относятся адрес узла источника пакета и адрес узла назначения, длина пакета и др., а в качестве примера информации, подлежащей передаче, можно указать часть текста электронного письма.

Пакет должен быть доставлен из пункта отправки в пункт назначения. Для этого используется адресация TCP/IP.

TCP/IP – Transmission Control Protocol/Internet Protocol (протокол управления передачей/межсетевой протокол). TCP/IP часто называют протоколом открытых систем.

TCP/IP представляет собой набор (стек) протоколов, обеспечивающих связь компьютеров (и других устройств) в сети Internet. Семейство TCP/IP состоит из многих протоколов (более двух десятков), и каждый занят своим делом. Каждый переносит сетевые данные в различных форматах и обладает различными возможностями. В зависимости от требований приложения для передачи данных по Internet используется определённый протокол из семейства TCP/IP. Семейство включает в себя протокол контроля транспортировки (TCP), адресный протокол Internet (IP) и множество других протоколов. Все они предназначены для передачи сообщений в сети Internet.

Адрес IP – это 32-разрядное значение, которое используется для правильной идентификации источника и адреса пункта назначения. Адрес IP обычно представляется в виде 204.107.2.100. Адрес можно разбить на четыре позиции по восемь битов каждая, а можно представить в двоичном коде. Эти позиции называют октетами. В приведённом примере IP-адреса число 204 – это значение первого октета, а 100 – четвёртого. Приведённый в примере IP-адрес можно легко записать в двоичном виде (для преобразования десятичной системы счисления в двоичную можно воспользоваться утилитой «Калькулятор» операционной системы Windows)

11001100.01101011.00000010.01100100

IP-адрес состоит из двух частей: номера сети и номера узла. Номер сети может быть выбран администратором сети произвольно либо назначен по рекомендации специального подразделения Internet (Network Information Center, NIC), если сеть должна работать как составная часть Internet. Обычно провайдеры услуг Internet получают диапазоны адресов у подразделений NIC, а затем распределяют их между своими абонентами.

Номер узла в протоколе IP назначается независимо от локального адреса узла. Деление IP-адреса на поле номера сети и номера узла – гибкое, и граница между этими полями может устанавливаться весьма произвольно. Узел может входить в несколько IP-сетей. В этом случае узел должен иметь несколько IP-адресов, по числу сетевых связей. Таким образом IP-адрес характеризует не отдельный компьютер или маршрутизатор, а одно сетевое соединение.

Для того чтобы в IP-адресе различить номер сети и номер узла, используют **маску подсети**. Маска подсети важна для разделения назначения адресов IP на классы. Существуют классы А, В, С, D, Е. В таблице 4.1 приведены стандартные значения маски подсети и соответствующие им классы сети.

В таблице 4.2 обобщены сведения по классам и узлам сети.

В таблице 4.3 приведены допустимые диапазоны частного сетевого адреса. Таблица 4.3 показывает, например, что в сетях класса С в первом октете адреса IP не может быть цифры 191 или 222, т.е. адрес IP: 191.107.1.190 будет ошибочным, если указана маска подсети 255.255.255.0.

4.1. Классы адресов Internet на примере адреса IP 204.107.2.100

Класс	Адрес сети	Адрес узла	Стандартное значение маски подсети
А	204	107.2.100	255.0.0.0
В	204.107	2.100	255.255.0.0
С	204.107.2	100	255.255.255.0

4.2. Обобщение классов А, В, С сетей и узлов

Класс	Действительное количество возможных сетей	Действительное количество возможных узлов в сети
А	126	16777214
В	16384	65534
С	2097151	254

4.3. Допустимые диапазоны частного сетевого адреса

Класс	Начало диапазона адреса в первом октете	Конец диапазона адреса в первом октете
A	001	126
B	128	191
C	192	221

Кроме IP-адреса существует **локальный адрес узла**, определяемый технологией, с помощью которой построена отдельная сеть, в которую входит данный узел. Для узлов, входящих в локальные сети, – это MAC-адрес сетевого адаптера или порта маршрутизатора, например 11-A0-17-3D-BC-01. Эти адреса назначаются производителями оборудования и являются уникальными адресами, так как управляются централизованно. Для всех существующих технологий локальных сетей MAC-адрес имеет формат 6 байтов: старшие 3 байта – идентификатор фирмы-производителя, а младшие 3 байта назначаются уникальным образом самим производителем.

На рисунке 4.1 приведён пример схемы локальной компьютерной сети из четырёх узлов: PC1, PC2, PC3, PC4 и сервера, работающего под управлением операционной системы Windows 2000 Server.

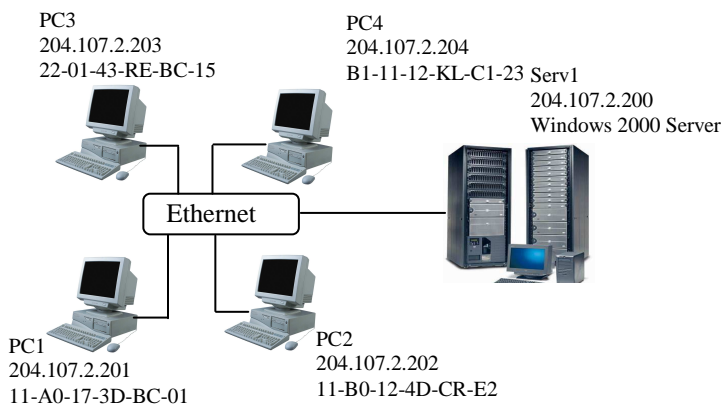


Рис. 4.1. Схема локальной компьютерной сети

Для определения адреса IP любого узла локальной сети достаточно знать его символическое имя и воспользоваться командой *ping*. Например, в результате выполнения команды *net view* было определено, что в компьютерной сети имеется три узла с именами PC1, PC2 и PC3.

```
>net view  
PC1  
PC2  
PC3
```

Теперь достаточно задать команду *ping Имя_узла*, чтобы получить адрес IP заданного узла

```
>ping PCI
```

Достаточно полезной для отображения параметров TCP/IP может оказаться утилита *ipconfig*, которая отображает в том числе имя узла, MAC-адрес и IP-адрес.

Для её использования достаточно ввести команду

```
> ipconfig /all
```

Более полную информацию по адресации узлов в IP-сетях можно найти на сайтах сети Интернет и в книгах [5 – 7].

Порядок выполнения работы

1. Вызвать командную строку для выполнения вводимых с клавиатуры команд (Пуск->Выполнить->*cmd*).

2. В командной строке выполните команду *net view*, в результате выполнения которой определите символьные имена узлов локальной сети, а также имя сервера.

3. Выполнив команды *ping* и *ipconfig*, определите адрес IP и MAC-адрес каждого узла.

4. Определите версию операционной системы, под управлением которой работает сервер.

5. Начертите схему локальной сети с указанием для каждого узла и сервера символьного имени, адреса IP, MAC-адреса. Для сервера укажите версию операционной системы.

Контрольные вопросы

1. Что такое адрес IP?
2. Что такое MAC-адрес?
3. Что такое маска подсети?
4. На какие классы делятся сети IP?
5. Даны адрес узла и маска подсети. Что здесь неверно?
Адрес узла в частной сети: 131.107.2.100
Маска подсети: 255.255.255.0
6. Дана маска подсети 255.255.0.0. К какому классу относится сеть? Каково максимальное количество узлов в сети?
7. Дана маска подсети 255.255.255.0. Число узлов в сети 255. Что здесь неверно?
8. Дан IP-адрес узла в частной сети: 221.101.2.150. Задайте правильную маску подсети.

СРЕДСТВА УСТРАНЕНИЯ НЕИСПРАВНОСТЕЙ В ТСП/IP

Цель работы: ознакомиться со средствами поиска неисправностей ТСП/IP.

Задание

1. Изучите методические указания и рекомендуемую литературу.
2. Примените утилиты *ipconfig* и *ping* для поиска неисправностей в настройке ТСП/IP.

Методические указания

Целью устранения неисправностей в настройке ТСП/IP является восстановление нормальной работы сети. Для поиска неисправностей можно использовать специальные диагностические утилиты, список некоторых из которых приведён в табл. 5.1.

Рассмотрим более подробно применение двух утилит из табл. 5.1. Утилита *IPConfig* обеспечивает отображение информации о ТСП/IP. Эту утилиту хорошо применять в самом начале тестирования системы, так как она даёт полную информацию о конфигурации ТСП/IP. Существуют различные варианты команды *IPConfig*. Они задаются с помощью переключателей командной строки. Например, команда

>ipconfig /?

позволяет вызвать справку о команде.

5.1. Средства и утилиты поиска неисправностей в ТСП/IP

Утилита или средство	Описание
IPConfig	Выводит текущую информацию о ТСП/IP. Переключатели командной строки позволяют изменять IP-адрес узла
Ping	Проводит тестирование соединений и проверяет настройки
Hostname	Введение этой команды в командной строке приводит к возвращению имени текущего узла
Route	Отображает или изменяет таблицу маршрутизации
ARP	Позволяет просмотреть таблицы ARP локального компьютера, чтобы обнаружить повреждённые записи

Ниже приведены некоторые возможные варианты переключателей и их описание.

<i>/all</i>	Вызов полных сведений о конфигурации
<i>/release</i>	Отображение адреса IP для указанного адаптера
<i>/renew</i>	Обновление адреса IP указанного адаптера

Чаще всего используется команда *ipconfig /all*. По этой команде отображается информация о каждом физически присутствующем сетевом адаптере, соединениях модема и виртуальных соединениях.

Команда *ping* передаёт пакеты протокола контроля сообщений в Internet между двумя узлами TCP/IP. Пример вызова справки о команде приведён ниже.

```
>ping /?
```

Рассмотрим пример поиска неисправностей в настройках TCP/IP. Пусть известна схема сети (см. рис. 2.1). Специалист, выполняющий поиск неисправностей в настройках TCP/IP, работает на рабочей станции (узле) PC3. Тогда при выполнении тестирования на первом шаге на локальном рабочем узле (узел PC3) выполняется команда *IPConfig* для просмотра настроек TCP/IP.

```
> IPConfig /all
```

Результат выполнения данной команды показан ниже.

```
Настройка Windows 2000 IP
Имя узла.....PC3
Имя основного домена.....Main.local
Тип узла.....Broadcast
Включение маршрутизации IP.....No
Физический адрес.....22-01-43-RE-BC-15
Адрес IP.....204.107.2.203
Маска подсети.....255.255.255.0
Шлюз по умолчанию.....204.107.2.200
```

Просмотрев выведенную информацию, определяем IP-адрес шлюза по умолчанию 204.107.2.200 (это сервер).

На втором этапе выполняется команда *ping* для внутреннего адреса обратной связи, чтобы проверить, что TCP/IP установлен и сконфигурирован правильно на локальном компьютере узла. Этот адрес – 127.0.0.1, выделенный адрес, который не может использоваться как реальный IP-адрес

```
>ping 127.0.0.1
```

На третьем этапе выполняется команда *ping* для локального удалённого узла (например, узел PC1), чтобы гарантировать, что TCP/IP работает правильно

```
> ping 204.107.2.201
```

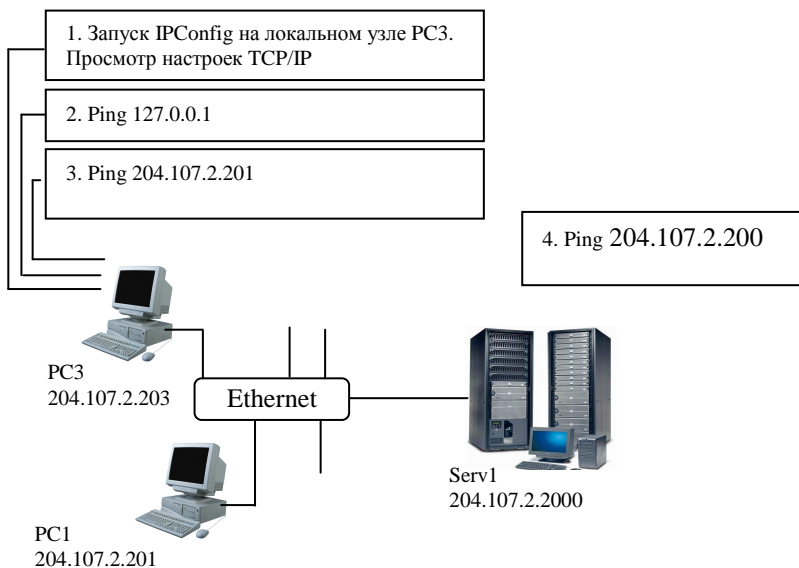


Рис. 5.1. Порядок совместного использования утилит IPConfig и Ping

На последнем этапе выполняется команда *ping* для адреса IP маршрутизатора или шлюза, используемого по умолчанию. Это позволит убедиться в правильном функционировании маршрутизатора или шлюза по умолчанию.

Для шлюза по умолчанию
> *ping* 204.107.2.200

Схема, отражающая порядок тестирования в приведённом примере, показана на рис. 5.1.

Более полную информацию по способам поиска неисправностей в настройках TCP/IP можно найти в литературе [8].

Порядок выполнения работы

1. Выполните команду *IPConfig* на локальном рабочем узле и просмотрите информацию о конфигурации TCP/IP.
2. Выполните команду *ping* для внутреннего адреса обратной связи.
3. Выполните команду *ping* для локального удалённого узла.
4. Выполните команду *ping* для адреса IP маршрутизатора или шлюза, используемого по умолчанию.
5. Разработайте схему, отражающую порядок выполнения тестирования TCP/IP утилитами *IPConfig* и *ping*.

Контрольные вопросы

1. Перечислите утилиты, которые можно использовать для поиска неисправностей в настройках TCP/IP. Каковы их возможности?
2. Утилита IPConfig. Назначение, параметры, результаты применения.
3. Утилита Ping. Назначение, параметры, результаты применения.
4. Порядок совместного применения утилит IPConfig и Ping.
5. Назначение утилиты ARP.
6. Какую утилиту можно применить для получения имени узла?
7. Для чего используется команда Route?
8. Поясните полученные в практической работе результаты.

Практическая работа 6

УСТРОЙСТВА СВЯЗИ. ПРОЕКТИРОВАНИЕ СЕТИ КРУПНОЙ ФИРМЫ

Цель работы: научиться осуществлять подбор сетевого оборудования, требуемого для создания сети организации.

Задание

Спроектируйте в виде примерной структурной схемы сеть крупной фирмы, состоящей из трёх подразделений:

- офис администрации (отдельный этаж здания в центре Москвы, десять рабочих мест);
- склад (отдельное здание за пределами МКАД) – оснащён пятью стационарными рабочими станциями;
- торговый центр (рынок стройматериалов большой площади и автостоянка для покупателей), персонал которого при работе с клиентами использует КПК, свободно перемещаясь по территории торгового центра и стоянок на расстоянии до 2 км.

В пределах офиса и склада подсети должны иметь звездообразную структуру. Для офиса администрации необходимо обеспечить возможность выхода в Интернет по каналу ADSL, а связь между подразделениями фирмы осуществляется по оптоволоконному кабелю. Считать определяющими показатели надёжности и скорости, пренебрегая стоимостью оборудования.

Результаты выполнения работы: схема сети, перечень необходимого оборудования.

ИСПОЛЬЗОВАНИЕ УДАЛЁННЫХ СЕТЕВЫХ РЕСУРСОВ

Цель работы: изучить способы подключения удалённых ресурсов общего доступа.

Задание

1. Изучите методические указания.
2. Выполните лабораторную работу в соответствии с порядком, изложенным в методических указаниях.

Методические указания

Одним из ощутимых преимуществ компьютерной сети является совместное использование таких ресурсов, как диски, папки, принтеры, физически находящиеся на различных узлах. Для реализации этого преимущества с компьютера, на котором установлен ресурс, необходимо разрешить к нему доступ, а на компьютере клиенте необходимо подключить этот ресурс. Пусть на компьютере с именем Serv 1 (рис. 7.1) на диске D содержится папка Inform. Предположим, что пользователю узла PC2 во время работы требуется получить доступ к содержимому папки Inform. В этом случае для пользователя PC2 папка Inform будет являться удалённым ресурсом.

Для подключения удалённого ресурса можно использовать три основных подхода: метод «выбрать и подключить», метод с использованием графического пользовательского интерфейса GUI, метод командной строки.

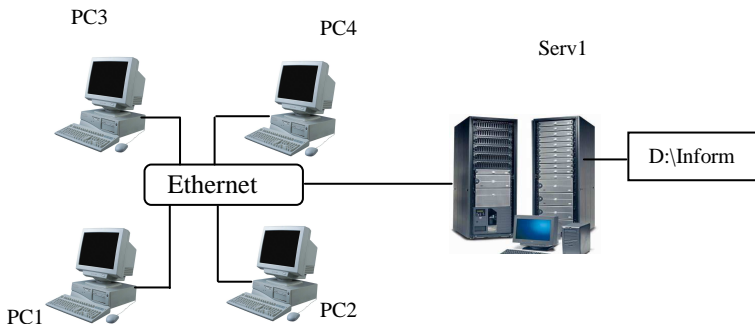


Рис. 7.1. К понятию об удалённом ресурсе

Подключение сетевого диска с помощью метода «выбрать и подключить» осуществляется в следующей последовательности:

- открыть диалоговое окно «Сетевое окружение»;
- раскрыть элемент «Вся сеть»;
- открыть компьютер, с которым требуется установить связь;
- выбрать совместно используемый ресурс, которому требуется поставить в соответствие сетевой диск;
- выбрать свободную букву сетевого диска.

Подключение сетевого диска методом GUI осуществляется в следующей последовательности:

- вызвать меню «Подключить сетевой диск...» щелчком правой кнопки мыши на ярлыке «Мой компьютер» или ярлыке «Сетевое окружение» (рис. 7.2);
- выбрать команду «Подключить сетевой диск»;
- выбрать букву диска и имя подключаемого ресурса (имя папки), указав путь к этому ресурсу.

Рассмотрим третий способ подключения удалённых ресурсов – метод командной строки. Описание некоторых полезных команд приведено ниже.

Для просмотра удалённых компьютеров и доступных на них ресурсов в командной строке используется команда NET VIEW.

Например, следующая команда показывает символьные имена всех доступных в данный момент компьютеров локальной сети

```
> NET VIEW
```

Для отображения всех доступных ресурсов компьютера с именем, например PC1, используется команда

```
> NET VIEW \\PC1
```

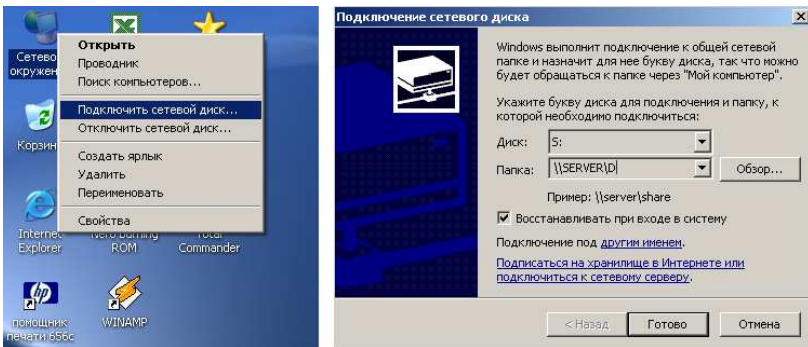


Рис. 7.2. Подключение удалённого ресурса методом GUI

Для подключения ресурсов через командную строку используется команда

```
>NET USE ДИСК: \\ИМЯ_КОМПЬЮТЕРА\РЕСУРС
```

где *ДИСК* – буква диска, к которой подключается удалённый ресурс, *ИМЯ_КОМПЬЮТЕРА* – имя удалённого компьютера (либо символическое, либо IP-адрес),

РЕСУРС – подключаемый ресурс удалённого компьютера.

Например, команда

```
>NET USE H: \\Serv1\Inform
```

позволит подключить ресурс Inform сервера Serv1 (см. рис. 7.1), используя букву диска H, команда

```
> NET USE
```

отображает все подключённые к данному компьютеру удалённые ресурсы, команда

```
> NET USE ДИСК: /D
```

производит отключение подключённого ранее ресурса.

Подключение сетевого принтера производится аналогичным образом, только вместо буквы диска пишется порт (например, LPT1):

```
>NET USE LPT1 \\ИМЯ_КОМПЬЮТЕРА\ИМЯ_ПРИНТЕРА
```

Чтобы обеспечить возможность создания общих (общим называется ресурс, к которому разрешён доступ с других компьютеров сети) ресурсов в операционной системе Windows XP, необходимо отключить режим **простого общего доступа к папкам**. Для этого необходимо выбрать в окне «Мой компьютер» меню «Сервис» – «Свойства папки...».

Затем в открывшемся окне выбрать вкладку «Вид» и снять отметку в пункте «Использовать простой общий доступ к файлам».

Для открытия общего доступа к папке необходимо на выбранной папке щёлкнуть правой кнопкой мыши и выбрать пункт меню «Общий доступ и безопасность...».

В открывшемся окне во вкладке «Доступ» необходимо выбрать кнопку «Открыть общий доступ к этой папке» и определить «Разрешения».

Под разрешениями понимается перечень пользователей, которым разрешён доступ к данной папке, а также определённые им права доступа (чтение, запись и др.).

Чтобы добавить в список разрешений того или иного пользователя, необходимо щёлкнуть на кнопке «Добавить», затем в открывшемся окне ввести имя пользователя.

Порядок выполнения работы

1. Просмотрите через командную строку список всех доступных компьютеров в локальной сети.
2. Просмотрите через командную строку список всех доступных ресурсов сервера.

3. Подключите ресурс сервера «NetShare» (тремя способами).
4. Просмотрите через командную строку список всех подключённых ресурсов к данному компьютеру.
5. Отключите все подключённые ранее ресурсы.
6. Создайте в папке C:\TEMP\ папку с именем DISK №Компьютера, например DISK1 или DISK12.
7. Откройте общий доступ к этой папке.
8. Для пользователей группы «Все» запретите все виды доступа (на чтение, на запись и др.).
9. Добавьте в список разрешений одного пользователя (например, ah08-01). Определите ему права полного доступа.

Контрольные вопросы

1. Что представляет собой удалённый ресурс? Примеры.
2. Какие способы подключения удалённых ресурсов вам известны?
3. Что такое общий ресурс? Приведите примеры.
4. Как создать общий ресурс?
5. Как подключить удалённый принтер, используя командную строку?
6. Как просмотреть список удалённых ресурсов узла?
7. Как удалить подключённый ранее сетевой диск?
8. Как добавить в список разрешений заданного пользователя?

Практическая работа 8

МЕТОДЫ ЗАЩИТЫ КОМПЬЮТЕРА ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА ИЗ ВНЕШНЕЙ СЕТИ И ПОИСК УЯЗВИМОСТЕЙ В СИСТЕМЕ ЗАЩИТЫ

Цель работы:

- 1) изучить возможности брандмауэра Windows и обозревателя Internet Explorer в защите от несанкционированного доступа;
- 2) изучить методы анализа защищённости информационных ресурсов.

Задание

1. Определите все доступные узлы в локальной сети.
2. Просканируйте порты сервера компьютерной сети.

Методы защиты компьютера от несанкционированного доступа из внешней сети

Брандмауэр – сочетание программного и аппаратного обеспечения, образующее систему защиты от несанкционированного доступа из внешней глобальной сети во внутреннюю сеть (интрасеть). Брандмауэр предотвращает прямую связь между внутренней сетью и внеш-

ними компьютерами, пропуская сетевой трафик через прокси-сервер, находящийся снаружи сети. Прокси-сервер определяет, следует ли разрешить файлу попасть во внутреннюю сеть. Брандмауэр называется также шлюзом безопасности.

Можно считать брандмауэр пограничным постом, на котором проверяется информация (часто называемая *трафик*), приходящая из Интернета или локальной сети. В ходе этой проверки брандмауэр отклоняет или пропускает информацию на компьютер в соответствии с установленными параметрами.

Когда к компьютеру пытается подключиться кто-то из Интернета или локальной сети, такие попытки называют «непредусмотренными запросами». Когда на компьютер поступает непредусмотренный запрос, брандмауэр Windows блокирует подключение. Если на компьютере используются такие программы, как программа передачи мгновенных сообщений или сетевые игры, которым требуется принимать информацию из Интернета или локальной сети, брандмауэр запрашивает пользователя о блокировании или разрешении подключения. Если пользователь разрешает подключение, брандмауэр Windows создаёт *исключение*, чтобы в будущем не тревожить пользователя запросами по поводу поступления информации для этой программы.

Если идёт обмен мгновенными сообщениями с собеседником, который собирается прислать файл (например, фотографию), брандмауэр Windows запросит подтверждение о снятии блокировки подключения и разрешении передачи фотографии на компьютер. А при желании участвовать в сетевой игре через Интернет с друзьями пользователь может добавить эту игру как исключение, чтобы брандмауэр пропускал игровую информацию на компьютер.

Хотя имеется возможность отключать брандмауэр Windows для отдельных подключений к Интернету или локальной сети, это повышает вероятность нарушения безопасности компьютера.

Чтобы открыть компонент «Брандмауэр Windows», нажмите кнопку **Пуск**, выберите пункты **Настройка, Панель управления, Сеть и подключения к Интернету и Брандмауэр Windows**.

В обозревателе Internet Explorer имеется несколько возможностей, позволяющих обеспечить защиту конфиденциальности, а также повысить безопасность личных данных пользователя.

Параметры конфиденциальности позволяют защитить личные данные пользователя – с помощью этих параметров можно понять, как просматриваемые веб-узлы используют эти данные, а также задать значения параметров конфиденциальности, которые будут определять, разрешено ли веб-узлам сохранять файлы «cookie» на компьютере.

В число параметров конфиденциальности Internet Explorer входят следующие.

- Параметры конфиденциальности, определяющие обработку на компьютере файлов «cookie». Файлы «cookie» – это созданные веб-узлом объекты, которые сохраняют на компьютере определённые сведения, например о предпочтениях пользователя при посещении данного узла. Кроме того, эти файлы могут также сохранять личные данные пользователя, такие как имя и адрес электронной почты.

- Оповещение безопасности, выдаваемое пользователю при попытке получить доступ к веб-узлу, не соответствующему заданным параметрам конфиденциальности.

- Возможность просмотра политики конфиденциальности РЗР для веб-узла.

Средства безопасности позволяют предотвратить доступ других пользователей к таким сведениям, на доступ к которым у них нет разрешения. Это, например, сведения о кредитной карточке, вводимые при покупках в Интернете. Эти средства безопасности могут также защитить компьютер от небезопасного программного обеспечения.

В число параметров безопасности Internet Explorer входят следующие.

- Возможность блокирования большинства всплывающих окон.

- Возможность обновления, отключения или повторного включения надстроек для веб-обозревателя.

- Средства повышения безопасности, предупреждающие пользователя о попытке веб-узла загрузить файлы или программы на компьютер.

- Цифровые подписи, которые подтверждают, что файл поступил действительно от указанного лица или издателя и с момента включения цифровой подписи в этот файл никем не внесены изменения.

- Безопасное подключение с использованием 128-разрядного ключа, которое применяется для связи с безопасными веб-узлами.

Поиск уязвимостей в системе защиты

Противостояние атакам – важное свойство защиты. Казалось бы, если в сети установлен межсетевой экран (firewall), то безопасность гарантирована, но это распространённое заблуждение может привести к серьёзным последствиям.

Например, межсетевой экран (МЭ) не способен защитить от пользователей, прошедших аутентификацию. А квалифицированному хакеру не составляет труда украсть идентификатор и пароль авторизованного пользователя. Кроме того, межсетевой экран не только не защищает от проникновения в сеть через модем или иные удалённые точки доступа, но и не может обнаружить такого злоумышленника.

При этом система защиты, созданная на основе модели адаптивного управления безопасностью сети (Adaptive Network Security, ANS),

способна решить все или почти все перечисленные проблемы. Она позволяет обнаруживать атаки и реагировать на них в режиме реального времени, используя правильно спроектированные, хорошо управляемые процессы и средства защиты.

Компания Yankee Group опубликовала в июне 1998 г. отчёт, содержащий описание процесса обеспечения адаптивной безопасности сети. Этот процесс должен включать в себя анализ защищённости, т.е. поиск уязвимостей, обнаружение атак, а также использовать адаптивный (настраиваемый) компонент, расширяющий возможности двух первых функций, и управляющий компонент.

Анализ защищённости осуществляется на основе поиска уязвимых мест во всей сети, состоящей из соединений, узлов (например, коммуникационного оборудования), хостов, рабочих станций, приложений и баз данных. Эти элементы нуждаются как в оценке эффективности их защиты, так и в поиске в них неизвестных уязвимостей. Процесс анализа защищённости предполагает исследование сети для выявления в ней слабых мест и обобщение полученных сведений, в том числе в виде отчёта. Если система, реализующая данную технологию, содержит адаптивный компонент, то устранение найденной уязвимости будет осуществляться автоматически. При анализе защищённости обычно идентифицируются:

- люки в системах (back door) и программы типа «троянский конь»;
- слабые пароли;
- восприимчивость к проникновению из внешних систем и к атакам типа «отказ в обслуживании»;
- отсутствие необходимых обновлений (patch, hotfix) операционных систем;
- неправильная настройка межсетевых экранов, WEB-серверов и баз данных.

Обнаружение атак – это процесс оценки подозрительных действий в корпоративной сети, реализуемый посредством анализа журналов регистрации операционной системы и приложения (log-файлов) либо сетевого трафика. Компоненты ПО обнаружения атак размещаются на узлах или в сегментах сети и оценивают различные операции, в том числе с учётом известных уязвимостей.

Адаптивный компонент ANS позволяет модифицировать процесс анализа защищённости, предоставляя самую последнюю информацию о новых уязвимостях. Он также модифицирует компонент обнаружения атак, дополняя его последней информацией о подозрительных действиях и атаках. Примером адаптивного компонента может служить механизм обновления баз данных антивирусных программ, которые являются частным случаем систем обнаружения атак.

Управляющий компонент предназначен для анализа тенденций, связанных с формированием системы защиты организации и генерацией отчётов.

К сожалению, эффективно реализовать все описанные технологии в одной системе пока не удаётся, поэтому пользователям приходится применять совокупность систем защиты, объединённых единой концепцией безопасности. Пример таких систем – семейство продуктов SAFEsuite, разработанных американской компанией Internet Security Systems (ISS). В настоящее время комплект ПО SAFEsuite поставляется в новой версии SAFEsuite Enterprise, в которую входит также ПО SAFEsuite Decisions, обеспечивающее принятие решений по проблемам безопасности.

Система анализа защищённости Internet Scanner предназначена для проведения регулярных всесторонних или выборочных тестов сетевых служб, операционных систем, используемого прикладного ПО, маршрутизаторов, межсетевых экранов, WEB-серверов и т.п.

Другим примером системы анализа защищённости является программа SuperScan, позволяющая сканировать открытые порты узлов с известными IP-адресами.

Для начала сканирования достаточно в поле Start указать IP-адрес сканируемого узла. Для более глубокого сканирования необходимо указать минимальную скорость сканирования, передвинув движок на отметку Min.

Результаты выполнения практической работы

Результаты поиска уязвимостей в системе защиты

узла _____ IP: ____ . ____ . ____ . ____

Символьное имя узла IP-адрес узла

№ п.п	№ порта	Наименование	№ п.п	№ порта	Наименование
1			7		
2			8		
3			9		
4			10		
5			11		
6			12		

Контрольные вопросы

1. Может ли брандмауэр блокировать компьютерным вирусам и «червям» доступ на компьютер?
2. Может ли брандмауэр обнаружить или обезвредить компьютерных вирусов и «червей», если они уже попали на компьютер?

3. Может ли брандмауэр запретить пользователю открывать сообщения электронной почты с опасными вложениями?
4. Может ли брандмауэр блокировать спам или несанкционированные почтовые рассылки?
5. Может ли брандмауэр запросить пользователя о выборе блокировки или разрешения для определённых запросов на подключение?
6. Для чего нужен журнал безопасности брандмауэра?
7. Сколько параметров определяют работу брандмауэра?
8. Перечислите параметры, определяющие работу брандмауэра.
9. Какой параметр брандмауэра обеспечивает наивысшую защиту компьютера?
10. Что такое брандмауэр? Его назначение.
11. Как работает брандмауэр?
12. Какими возможностями обладает интернет-обозреватель для защиты личных данных пользователей?
13. Перечислите параметры безопасности Internet Explorer.
14. Перечислите параметры конфиденциальности Internet Explorer.
15. Для чего нужно проводить анализ защищённости сети?
16. Перечислите наиболее уязвимые «места» сети.
17. Приведите примеры систем анализа уязвимостей.

Практическая работа 9

ИЗУЧЕНИЕ ЗАПОМИНАЮЩИХ УСТРОЙСТВ ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН

Цель работы: изучить устройство постоянных и оперативных запоминающих устройств и получить навыки работы с программаторами ПЗУ.

Задание

1. При помощи лабораторного стенда произвести запись и чтение информации в ОЗУ.
2. При помощи лабораторного стенда произвести запись и чтение информации в ПЗУ.
3. Произвести программирование ПЗУ с использованием программатора, встроенного в персональный компьютер.

Методические указания

Память электронных вычислительных машин (ЭВМ) часто разделяют на так называемую внутреннюю, которая выполняется на основе полупроводниковых микросхем, и внешнюю – в виде магнитных дис-

ков, лент или оптических носителей, обеспечивающих долговременное хранение большого объёма информации. ЭВМ для работы с внешней памятью требуются дополнительные аппаратные (например, жёсткий диск и контроллер жёсткого диска) и программные средства (программы-драйверы). Центральный процессор (ЦП) не может самостоятельно пользоваться внешней памятью. Внутренняя память системы может быть непосредственно доступна для ЦП.

Запоминающее устройство (ЗУ) состоит из огромного числа **элементов памяти** (ЭП), каждый из которых может находиться в одном из двух состояний, кодируемых двоичной цифрой 0 или 1. Элемент памяти представляет собой область, где хранится бит информации. Элементы памяти ЗУ группируются в **слова информации**, т.е. такие порции информации, которые могут одновременно пересылаться между ЗУ и ЦП и обрабатываться последним. Область ЗУ, где хранится слово информации, называется **ячейкой памяти**. Структура ЗУ, состоящего из n ячеек, каждая из которых хранит слово из m битов, представлена на рис. 9.1.

Поиск нужного слова ЗУ можно производить либо по его адресу (адресные ЗУ), либо по его частичному содержанию (ассоциативные ЗУ). В адресных ЗУ обращение к ячейкам памяти производится по их физическим координатам, задаваемым двоичным кодом – **адресом**. Они бывают с произвольным обращением (выборкой), т.е. допускают любой порядок следования адресов, и с последовательным обращением, где выборка ячеек памяти возможна только в определённом порядке возрастания или убывания адресов. Архитектура ЭВМ ориентирована, в первую очередь, на использование адресных ЗУ с произвольной выборкой.

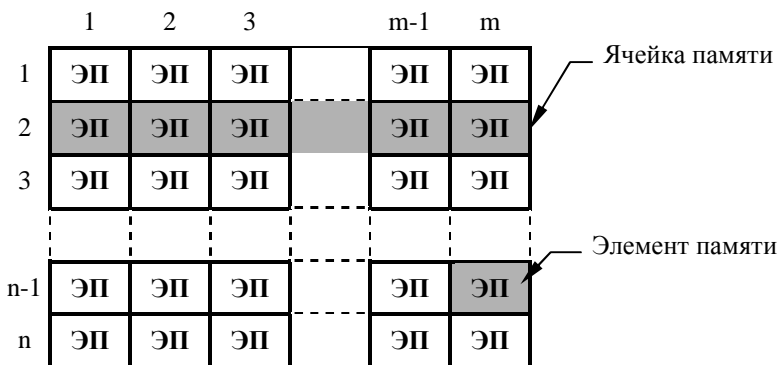


Рис. 9.1. Структура запоминающего устройства

Информационная ёмкость (или просто ёмкость) ЗУ выражается в количестве битов (б), байтов (Б) или слов, состоящих из определённого числа битов. Так как эта ёмкость может быть очень велика (до 10^{12} бит), то обычно используют более крупные единицы, образованные присоединением приставок к вышеперечисленным единицам: кило- (к), мега- (М) или гига- (Г). При этом надо учитывать, что в системах передачи и обработки информации приставки к, М и Г соответственно равны $2^{10} = 1024$, $2^{20} = 1\,048\,676$ и $2^{30} = 1\,073\,741\,824$.

Ячейки памяти обычно содержат 1, 4 или 8 ЭП. Поэтому ЗУ, содержащее, например, 2048 элементов памяти и имеющее ёмкость 2048 бит, 2 кбит или 256 байт, можно изготовить для хранения 2048 1-битовых слов, 512 4-битовых слов или 256 8-битовых слов ($2\text{к} \times 1$, 512×4 или 256×8).

На рисунке 9.2 представлена классификация полупроводниковых ЗУ, применяемых в ЭВМ.

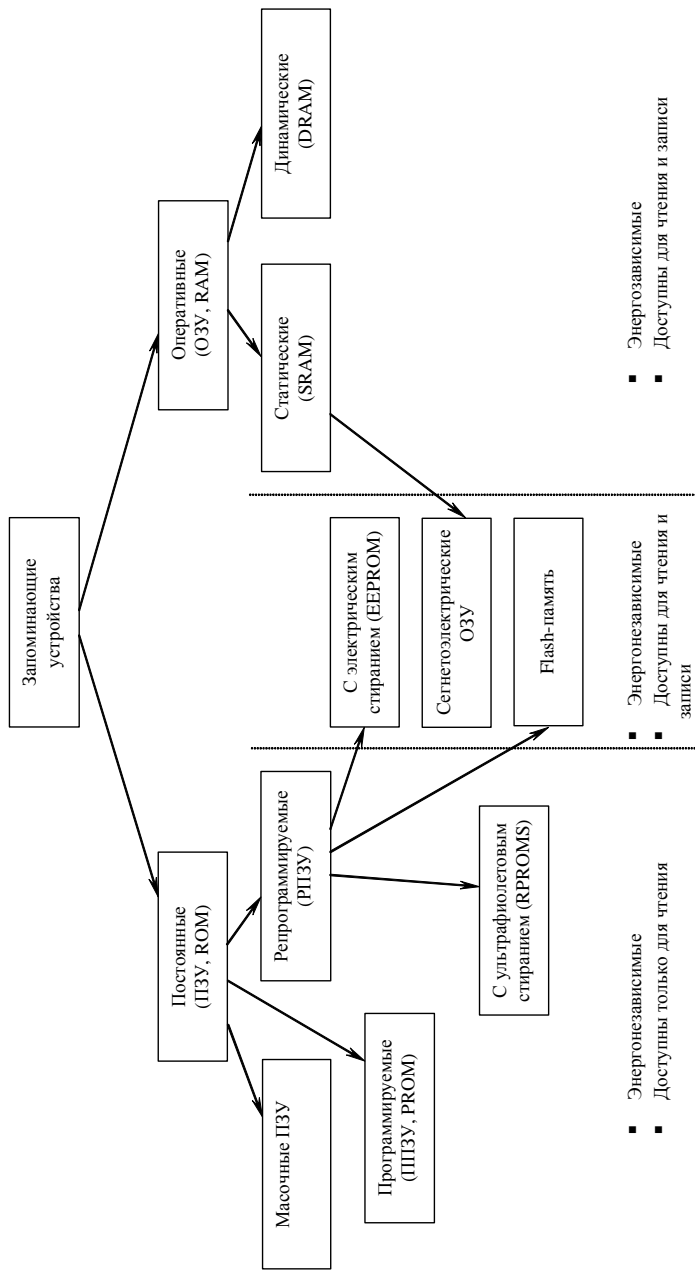
Постоянные запоминающие устройства

Благодаря энергонезависимости ПЗУ (ROM, Read Only Memory – память только для чтения) применяются для хранения инициализирующих и управляющих программ, различных таблиц констант и т.д.

Процесс занесения информации в ПЗУ называется программированием. Для этого служат программаторы, выполняемые в виде автономных или периферийных устройств ЭВМ, в которых производится подготовка и хранение на внешних носителях записываемой в ПЗУ информации.

Исключением являются репрограммируемые ПЗУ (ППЗУ) с электрическим стиранием и записью информации (EEPROM), частным случаем которых являются ЗУ, выполненные по Flash-технологии. Эти типы ЗУ допускают общее стирание и запись информации непосредственно микропроцессором. Это свойство приближает их к ОЗУ, но в отличие от последних EEPROM и Flash ПЗУ имеют ограниченное число циклов стирание/запись и обладают энергонезависимостью.

На рисунке 9.3 приведён пример внутренней структуры ПЗУ с организацией 256 ячеек по 8 бит (256×8). Блоки и сигналы, предназначенные для программирования ПЗУ, на схеме условно не показаны. Для адресации ячеек памяти служат 8 входных линий А0-А7. По ним можно задавать до $2^n = 2^8 = 256$ различных адресов в двоичном коде. Младший разряд обозначен цифрой 0, старший – 7. Дешифратор адреса преобразует двоичный адрес в позиционный код для выбора одной из 256 строк матрицы элементов памяти. С выхода выбранного ЭП на линиях считывания вырабатываются двоичные сигналы 0 или 1. На выходы данных D0-D7, соединённые с вертикальными линиями считывания через усилители ВА0-ВА7, поступит код, соответствующий хранящейся в адресуемой ячейке памяти информации.



- Энергонезависимые
- Доступны только для чтения

- Энергонезависимые
- Доступны для чтения и записи

- Энергозависимые
- Доступны для чтения и записи

Рис. 9.2. Классификация полупроводниковых запоминающих устройств

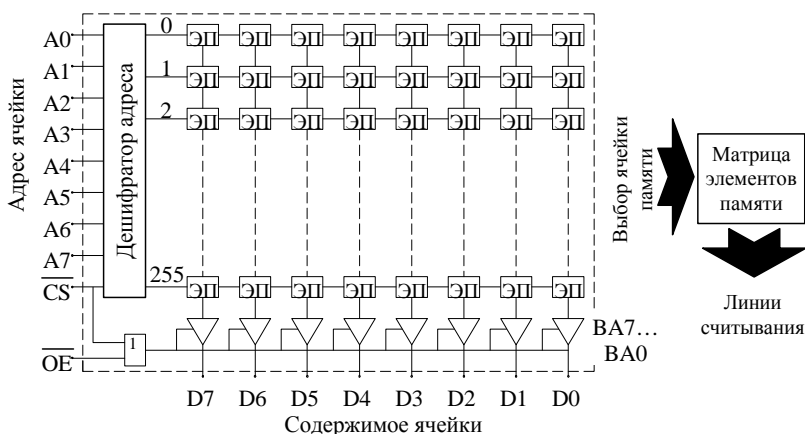


Рис. 9.3. Структура ПЗУ с произвольной выборкой

Управляющие входы CS (Crystal Select – выбор кристалла) и OE (Out Enable – разрешение по выходу) являются инверсными, т.е. активный уровень – логический ноль. Вход CS управляет общим выбором микросхемы, т.е. при подаче нуля (единицы) разрешается (запрещается) дешифрация адреса и выбор ячейки памяти. У выбранного ПЗУ с помощью входа OE производится активизация выходных буферов-усилителей BA0-BA7.

При отсутствии сигнала CS и OE выходы D0-D7 находятся в третьем (высокоимпедансном) состоянии. В третьем состоянии (в отличие от состояний логического нуля или единицы) выходы D0-D7 фактически являются отключёнными от его внутренней схемы. Такое состояние необходимо для возможности организации двунаправленной шины обмена данными. В **шину данных** параллельно включается необходимое число устройств (центральный процессор, ОЗУ, ПЗУ, порты ввода-вывода). В любой момент времени активными могут быть выходы только одного из устройств. Остальные устройства переходят в режим чтения или третье состояние. Информация передаётся от устройства с активными выходами к устройству, которое находится в режиме чтения. После завершения передачи информации в качестве передающего и приёмного могут выступить любые другие устройства системы. Выбор передающего и приёмного устройств осуществляет центральный процессор при помощи **шины управления**, состоящей из индивидуальных линий типа CS и OE, соединённых с каждым устройством системы.

Существует несколько разновидностей ПЗУ, которые различаются принципом программирования и технологией изготовления.

Масочно-программируемые ПЗУ. Информация заносится в них в процессе изготовления, обычно на финишном его участке, и не может быть впоследствии изменена. В серийном производстве эти микросхемы относительно дешёвы. Однако каждая «прошивка», т.е. заносимый в ПЗУ массив информации, требует соответствующей дорогостоящей технологической подготовки производства – индивидуальной маски (фотошаблона). Поэтому данный тип ПЗУ рентабельно применять в уже отлаженных изделиях, выпускаемых большими партиями.

Программируемые ПЗУ (ППЗУ, PROM) с плавкими перемычками поступают к потребителю в первоначальном незапрограммированном состоянии, соответствующем нулям или единицам во всех ЭП. В режиме программирования нужную информацию записывают в ППЗУ путём пережигания плавких перемычек, играющих роль ЭП, электрическим током с помощью программатора. В дальнейшем изменение информации, занесённой в ППЗУ, возможно только путём пережигания перемычек, оставшихся после предыдущего программирования.

Репрограммируемые ПЗУ (РПЗУ) с ультрафиолетовым стиранием (RPROMS) информации в настоящее время редко используются в ЭВМ. В этих ЗУ каждый бит хранимой информации отображается состоянием соответствующего МОП-транзистора с плавающим затвором, представляющим собой конденсатор. Заряжая и разряжая его, производят запись и стирание информации. Заряд в таких конденсаторах может сохраняться очень долго (более 10 лет) за счёт высокого качества изолирующего слоя.

Незапрограммированная микросхема РПЗУ с ультрафиолетовым стиранием имеет на выходах по всем адресам уровень логической единицы.

Для записи в требуемые разряды логического нуля на соответствующие выводы данных подаётся уровень 0, а на остальные – 1. Можно производить коррекцию ранее записанной информации, изменяя состояние 1 на 0 (но не наоборот).

Для стирания информации в течение 30 – 60 минут облучают кристалл микросхемы сквозь прозрачное окно в её корпусе ультрафиолетовым излучением люминесцентной лампы, которое увеличивает ток утечки в изолирующем слое, приводя к рассасыванию хранимого на плавающих затворах заряда.

Число циклов перезаписи лежит обычно в пределах 10...100 (для различных типов), так как по мере перепрограммирования постепенно ухудшаются диэлектрические свойства изолирующего слоя.

РПЗУ с ультрафиолетовым стиранием применялись на начальном этапе развития микропроцессорной техники. В настоящее время они вытеснены РПЗУ с электрическим стиранием различных видов.

ПЗУ с электрическим стиранием и электрической записью (EEPROM) позволяет производить как запись, так и стирание информации с помощью электрических сигналов. Благодаря этому появляется возможность изменять содержимое постоянной памяти непосредственно в ЭВМ, если там предусмотрены устройства формирования сигналов стирания и программирования.

Достоинством ПЗУ с электрическим стиранием является не только удобство и высокая скорость перезаписи информации, но и значительное допустимое число циклов перезаписи. Современные технологии гарантируют не менее 1 000 000 циклов.

Flash-память является разновидностью ПЗУ с электрическим стиранием. В настоящее время Flash ПЗУ широко используются для организации программной памяти микроконтроллеров и в качестве мобильных устройств для хранения больших объёмов информации. Возможность создания ЗУ большой ёмкости обусловлена малым размером ЭП по сравнению с ЭП EEPROM. Однако при этом уменьшается ресурс ЗУ. Для Flash-памяти обычно допускается не менее 1000 циклов перезаписи.

На рисунке 9.4 представлены примеры графических обозначений ПЗУ на принципиальных схемах. Функциональное назначение микросхемы указывается в середине. Слева расположены выходы входных сигналов адреса и управления, справа – выходы данных, подачи питания (U_{CC} , GND) и напряжения программирования (U_{PR}). Знак инверсии на входе CS показывает, что активация микросхемы производится логическим уровнем 0, а выключение – 1. Запись производится при подаче на вход WR логического уровня 1, чтение – 0.

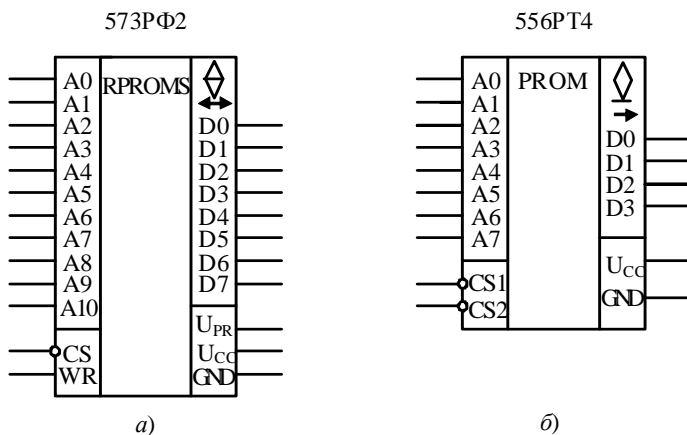


Рис. 9.4. Условные графические обозначения ПЗУ:

а – ПЗУ с ультрафиолетовым стиранием и двунаправленным выходом с тремя состояниями; *б* – программируемое ПЗУ с плавкими перемычками

Для обозначения типов выходов данных используются следующие символы:

- ◊ – выход с тремя состояниями;
- ◊ – выход с открытым коллектором;
- ↔, → – направление передачи информации.

Оперативные запоминающие устройства

ОЗУ (RAM, Random Access Memory – память с произвольной выборкой) служит для хранения временной и изменяемой информации, например отлаживаемых программ и промежуточных данных пользователя. Его главные преимущества перед ПЗУ – высокая скорость записи и чтения информации и неограниченное число циклов перезаписи. Предварительного стирания содержимого ячеек памяти не требуется. После выключения питания информация, находящаяся в ячейках ОЗУ, теряется.

ОЗУ в зависимости от структуры элементов памяти подразделяются на статические и динамические (см. практическую работу 1).

Кроме статических и динамических ОЗУ (SRAM и DRAM) в настоящее время разрабатываются новые типы ОЗУ, обеспечивающие энергонезависимое хранение информации. Примером такого устройства являются статические ОЗУ фирмы FRAMTON, элементы памяти которых выполнены на основе сегнетоэлектриков. Подобные ОЗУ, не уступая ПЗУ во времени хранения информации, превосходят их по быстродействию и количеству операций записи. Однако широкое их применение сдерживают высокая стоимость и недостаточная ёмкость.

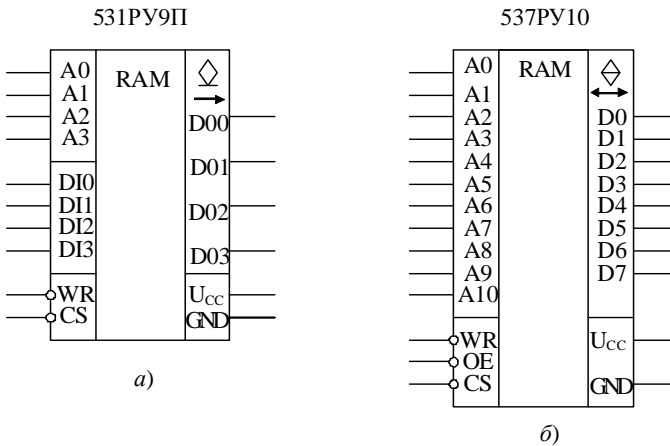


Рис. 9.5. Условные графические обозначения ОЗУ:

а – ОЗУ с отдельными входами и выходами данных;

б – ОЗУ с совмещёнными входами/выходами данных

Внутренняя структура ОЗУ близка к схеме, приведённой на рис. 9.3. Отличие заключается в том, что кроме линий считывания имеются также линии записи.

На рисунке 9.5 представлены примеры графических обозначений ОЗУ на принципиальных схемах.

Существует несколько вариантов организации выводов данных ОЗУ. На рисунке 9.5, *а* показано условное графическое обозначение ОЗУ с отдельными входами (DI0-DI3) и выходами (DO0-DO3) данных, а на рис. 9.5, *б* – ОЗУ с совмещёнными входами/выходами (D0-D7) данных. Знак « \leftrightarrow » свидетельствует о том, что выводы данных являются двунаправленными.

Программаторы

Программаторы служат для занесения информации в программируемые и репрограммируемые ПЗУ. Программаторы выполняют либо в виде автономных устройств, либо на базе компьютеров.

Автономные программаторы имеют ограниченные функциональные возможности и применяются в основном для копирования информации с ПЗУ-оригинала. Более совершенными являются программаторы, построенные на базе компьютеров. Возможны два варианта их подключения: через стандартный интерфейс (COMPORT, LPT, USB) и через системную шину компьютера. В первом случае программатор является внешним блоком компьютера, во втором – его внутренним модулем.

В лаборатории имеются все рассмотренные типы программаторов.

Первый программатор является автономным устройством для программирования ППЗУ с плавкими перемычками типа 155PE3 (32×8), 556PT4 (256×4), 556PT5 (512×8) и др. В состав программатора входит ОЗУ ёмкостью 1кБ и организацией 1024 ячейки по 8 разрядов.

Программатор позволяет в ручном режиме осуществлять предварительную подготовку (ввод и редактирование) данных в ОЗУ ёмкостью 1кБ и организацией 1024 ячейки по 8 разрядов. При наличии ПЗУ-оригинала можно автоматически копировать из неё информацию в ОЗУ. Программирование ПЗУ происходит также автоматически по оригиналу информации, хранящейся в ОЗУ. При этом производится контроль правильности записи по каждому адресу.

Адресация ячеек ОЗУ может осуществляться последовательно и произвольно. При произвольном доступе можно по любому адресу извлечь интересующую информацию и также изменить её. Для этого с помощью тумблеров "A0"... "A9" задаётся необходимый адрес в двоичном коде. При нажатии кнопки "▼" этот адрес отправляется на ОЗУ. Двоичный код текущего адреса можно проконтролировать с помощью

светодиодов, расположенных под тумблерами "A0"... "A9". Данные, хранящиеся по заданному адресу, отображаются светодиодами "D0"... "D7" также в двоичном коде. Погашенный светодиод сигнализирует логический 0, горящий – 1.

При последовательном доступе возможен выбор ячеек, адреса которых задаются только последовательно в порядке увеличения или уменьшения. Клавиша "►" служит для увеличения, а клавиша "◄" – для уменьшения текущего (отображаемого светодиодами "A0"... "A8") адреса. Кнопка «Сброс» позволяет установить начальный адрес.

Второй программатор выполнен на базе компьютера. Он представляет собой встраиваемый модуль и внешний блок с панельками для установки ПЗУ. Имеется специальное программное обеспечение для управления процессом ввода редактирования, хранения данных и программирования ПЗУ. Поэтому данный программатор обеспечивает большие функциональные возможности и широкую номенклатуру программируемых ПЗУ.

Третий программатор также выполнен на базе компьютера, но подключается к нему через COMPORT. Он предназначен для Flash ПЗУ, которые не требуют повышенного напряжения при программировании.

Порядок выполнения работы

Работа с ОЗУ

1. Установите переключатель режима работы автономного программатора в положение «ОЗУ».

2. Просмотрите содержимое ячеек ОЗУ, имеющих следующие адреса: 00h-07h; 0Fh-15h; E8h-EDh¹. Результат представьте в виде табл. 9.1 в шестнадцатеричной (hex) и двоичной (bin) системах счисления.

3. Запишите в ОЗУ информацию в соответствии с табл. 9.2. Для изменения информации новые данные задаются тумблерами "D0"... "D7", а их ввод производится при нажатии кнопки «Запись». Правильность записанных данных можно проконтролировать с помощью светодиодов "D0"... "D7".

9.1. Содержимое ячеек ОЗУ

Адрес ячейки		Содержимое ячейки	
hex-код	bin-код	bin-код	hex-код

¹ Символ «h» обозначает представление числа в шестнадцатеричной системе.

9.2. Данные для записи в ОЗУ

Адрес, hex-код	Данные, hex-код
00	01
01	02
02	04
03	08
04	10
05	20
06	40
07	80

Адрес, hex-код	Данные, hex-код
08	FE
09	FD
0A	FB
0B	F7
0C	EF
0D	DF
0E	BF
0F	7F

4. Проверьте правильность ввода информации путём просмотра данных.

5. Выключите тумблер питания стенда на несколько секунд, а затем включите его. Убедитесь, что вся ранее введённая информация отсутствует.

Работа с ПЗУ

6. Установите переключатель режима работы автономного программатора в положение «ПЗУ».

7. Вставьте в панельку на лицевой панели лабораторного стенда микросхему РПЗУ с УФ-стиранием типа K573РФ2. При этом соблюдайте правильность соединения по «ключу», обозначающему первый вывод микросхемы и панельки.

8. Просмотрите содержимое ячеек ПЗУ в области адресов 00h – 18h.

9. Используя кодировку ASCII, представьте хранящуюся информацию в виде символов. Результаты оформите в виде табл. 9.3.

10. Поместите ПЗУ в программатор на базе персонального компьютера.

11. Читайте содержимое ПЗУ и сравните данные с полученными ранее.

12. Запишите в свободные ячейки ПЗУ какой-либо текст в кодировке ASCII.

9.3. Содержимое ячеек ПЗУ

Адрес, hex-код	Данные, bin-код	Данные, hex-код	Символы ASCII

Контрольные вопросы

1. Что представляет собой внутренняя память ЭВМ?
2. Что называют элементом памяти и словом памяти?
3. Что называют ячейкой памяти?
4. Какие способы поиска данных в ЗУ существуют?
5. Как различают ЗУ по времени хранения данных?
6. Для чего в ЭВМ применяют ПЗУ?
7. Для чего в ЭВМ применяют ОЗУ?
8. Как устроены ЗУ?
9. Как различают ПЗУ по способу программирования?
10. Как различают ПЗУ по способу стирания?
11. Для чего предназначены программаторы?
12. Какие варианты исполнения программаторов Вы знаете?
13. Каковы принципы работы ОЗУ различных типов?
14. Какие варианты организации ввода-вывода данных ЗУ существуют?

Практическая работа 10

ИЗУЧЕНИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ МИКРОКОНТРОЛЛЕРОВ

Цель работы: получить навыки работы в интегрированной среде разработки программного обеспечения для однокристальных микроконтроллеров.

Задание

1. Изучите основные компоненты среды μ Vision2 фирмы Keil Software Inc. и ознакомьтесь с принципами работы в ней.
2. В среде μ Vision2 создайте проект, в основе которого лежит демонстрационная программа.

Методические указания

Одной из важных составляющих системы менеджмента качества в производстве различного рода продукции является автоматизированная система контроля и управления технологическим процессом (АСКиУ ТП). В настоящее время как управление технологическим процессом, так и контроль технологических параметров целесообразно осуществлять с использованием микропроцессорной техники. Это позволяет устранить погрешности, связанные с передачей аналоговых сигналов на большие расстояния, обеспечить экономию кабельной продукции, повысить надёжность и оперативность работы системы,

получить возможность реализации алгоритмов управления любой сложности и самодиагностики.

В АСКУ применяют микропроцессорные системы на основе **промышленных компьютеров и контроллеров**. Контроллер представляет собой ЭВМ с ограниченным набором функций, оптимизированные под решение конкретной задачи. По сравнению с компьютерами они отличаются существенно меньшими габаритами, значительно упрощённым интерфейсом пользователя (или его полным отсутствием), меньшей стоимостью, высокой надёжностью и возможностью приспособления под любые условия эксплуатации.

Контроллер, как правило, строится на основе **микроконтроллера**, т.е. простейшей ЭВМ, размещённой в корпусе одной микросхемы (на одном кремниевом кристалле). Поэтому микроконтроллеры часто называют однокристалльными. В состав микроконтроллера входят микропроцессор, ОЗУ, ПЗУ, порты ввода-вывода и ряд других устройств в зависимости от области применения, например аналого-цифровые (АЦП) и цифро-аналоговые преобразователи (ЦАП).

В лабораторном контроллере используется универсальная отладочная плата EB-552 фирмы КТЦ-МК, расположенная в корпусе, снабжённом разъёмами для подключения внешних устройств и напряжения питания. На плате установлены жидкокристаллический дисплей, простейшая клавиатура и однокристалльный микроконтроллер РСВ80С552 фирмы PHILIPS, принадлежащий к семейству MCS-51, базовой моделью которого является микроконтроллер I-8051 фирмы INTEL [16 – 18].

В основе работы микропроцессорной вычислительной техники лежит принцип программного управления. Для создания программного обеспечения микропроцессорных систем широко используются средства вычислительной техники, в том числе персональные компьютеры, и специальные программы, позволяющие разработчику выполнить весь цикл проектирования, включая отладку целевой программы. Рассмотрим технологию разработки программного обеспечения микроконтроллеров на примере использования продукта американской фирмы Keil Software Inc., называемого μ Vision2.

μ Vision2 фирмы Keil Software Inc. – интегрированная среда разработки программного обеспечения для однокристалльных микроконтроллеров семейства MCS-51. Keil μ Vision2 имеет стандартный интерфейс Windows и включает в себя всё, что нужно для создания, редактирования, компиляции, трансляции, компоновки, загрузки и отладки программ:

- организатор проекта;
- полнофункциональный редактор исходных текстов с выделением синтаксических элементов цветом;

- компилятор языка Си;
- макроассемблер;
- библиотеку стандартных функций;
- компоновщик;
- отладчик;
- операционную систему реального времени.

Первый этап разработки программного обеспечения – создание и настройка проекта под конкретный тип микроконтроллера.

На следующем этапе осуществляется запись исходного текста программы на каком-либо языке программирования. После этого производится его трансляция в коды команд микроконтроллера с использованием компилятора C51 или ассемблера A51. Компиляторы и ассемблеры – прикладные программы, которые преобразуют исходный текст программы в объектный модуль, представляющий собой перемещаемый программный код с относительной адресацией. Объектные и библиотечные модули с помощью программы компоновщика L51 объединяются в исполняемый программный код, размещаемый по абсолютным адресам.

После компоновки объектных модулей наступает **этап отладки** программы, устранения ошибок, оптимизации и тестирования программы. В составе среды Keil μ Vision2 имеются мощные средства отладки, позволяющие симулировать работу микроконтроллера в режиме выполнения программы, наблюдать за содержимым регистров, памяти и контролировать работу всех устройств. Это позволяет исправить практически все ошибки и получить рабочую версию программы до создания самого устройства. Для загрузки готовой программы в память микроконтроллера обычно используют выходной файл в формате HEX, получаемый с помощью программы-конвертора OH51.

На рисунке 10.1 схематически представлен процесс создания программного обеспечения для микроконтроллеров. В составе Keil μ Vision2 имеются следующие основные компоненты.

Макроассемблер A51. Ассемблер A51 транслирует символическую мнемонику в перемещаемый объектный код, имеющий высокое быстродействие и малый размер. Макросредства ускоряют разработку и экономят время, поскольку общие последовательности могут быть разработаны только один раз. Ассемблер поддерживает символический доступ ко всем элементам микроконтроллера и перестраивает конфигурацию для каждой разновидности MCS-51.

Оптимизирующий кросс-компилятор C51. Язык Си – универсальный язык программирования, который обеспечивает эффективность кода, элементы структурного программирования и имеет богатый набор операторов.

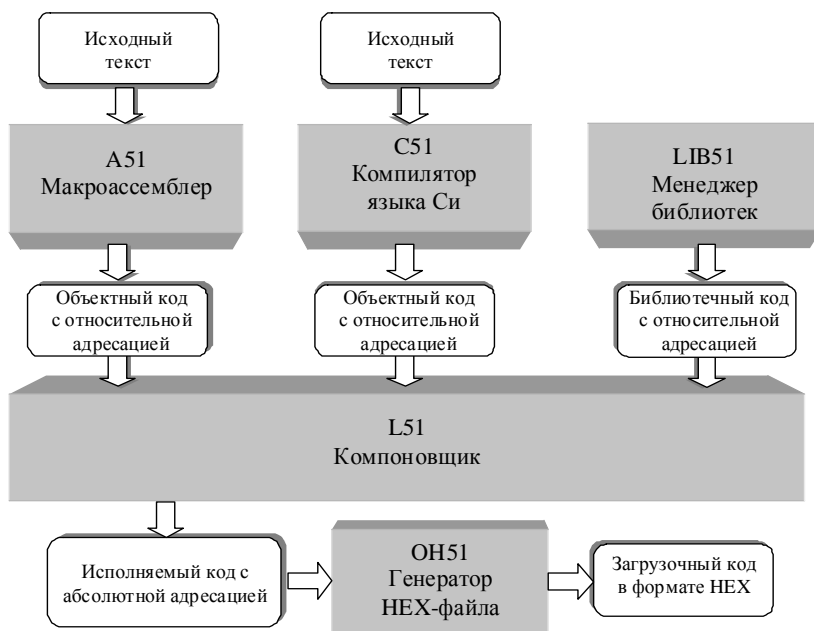


Рис. 10.1. Схема процесса создания программ для микроконтроллеров

Универсальность, отсутствие ограничений реализации делают язык Си удобным и эффективным средством программирования для широкого разнообразия задач. Множество прикладных программ может быть написано легче и эффективнее на языке Си, чем на других более специализированных языках [14, 15].

C51 – полная реализация стандарта ANSI (Американского национального института стандартов), насколько это возможно для архитектуры MCS-51, генерирует код для всего семейства этих микроконтроллеров. Компилятор сочетает гибкость программирования на языке высокого уровня с эффективностью кода и быстродействием ассемблера.

Использование языка высокого уровня Си имеет следующие преимущества над программированием на ассемблере:


- глубокого знания системы команд процессора не требуется, элементарное знание архитектуры микроконтроллера желательно, но не необходимо;
- распределение регистров и способы адресации управляются полностью компилятором;
- лучшая читаемость программы, используются ключевые слова и функции, которые более свойственны человеческой мысли;

- время разработки программ и их отладки значительно короче в сравнении с программированием на ассемблере;
- библиотечные файлы содержат много стандартных подпрограмм, которые могут быть включены в прикладную программу;
- существующие программы могут многократно применяться в новых программах, используя модульные методы программирования.

Компоновщик L51. Компоновщик объединяет один или несколько объектных модулей в одну исполняемую программу. Компоновщик размещает внешние и общие ссылки, назначает абсолютные адреса перемещаемым сегментам программ. Он может обрабатывать объектные модули, созданные компилятором C51 и ассемблером A51. Компоновщик автоматически выбирает соответствующие библиотеки поддержки и связывает только требуемые модули из библиотек. Установки по умолчанию для L51 выбраны так, чтобы они подходили для большинства прикладных программ, но можно определить и заказные установки.

Порядок работы в среде keil μ vision2

1. Запуск Keil μ Vision2

Для запуска Keil μ Vision2 следует открыть файл \keil\UV2\Uv2.exe с ярлыком .

На рисунке 10.2 представлен общий вид окна среды Keil μ Vision2.

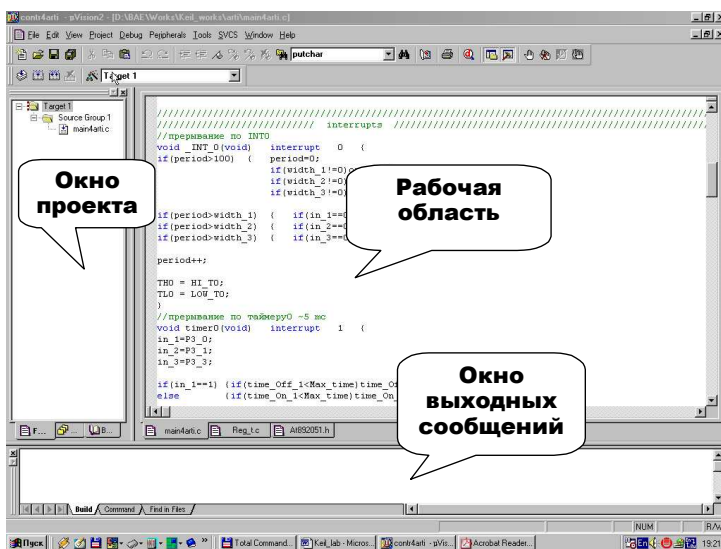


Рис. 10.2. Общий вид окна среды Keil μ Vision2

В верхней части экрана находится панель, включающая следующие меню:

File – для работы с файлами;

Edit – для редактирования файлов;

View – для управления режимом отображения;

Project – для настройки параметров проекта;

Debug – для отладки программ;

Peripherals – для контроля периферийных устройств при отладке программ;

Tools – для подключения дополнительных программ;

SVCS – для контроля версии программы;

Window – для выбора способа расположения окон;

Help – для получения справочной информации.

2. Создание нового проекта

Разработка программного обеспечения начинается с установки параметров интегрированной среды Keil μ Vision2. Совокупность настроек среды, исходных файлов программ, стандартных библиотек и библиотечных модулей для решения конкретной задачи называется Project (проект). Созданный проект может быть сохранён в файле с расширением *.Uv2 и использован для дальнейшей работы. Для разработки программы сразу для нескольких типов микроконтроллеров в проекте можно создавать несколько целевых задач, называемых Target. Каждая из них имеет общие для всех исходные файлы, однако может различаться настройками среды.

Для создания нового проекта выбираем команду New Project... из меню Project. При этом на экран выводится окно выбора устройства (рис. 10.3), в котором следует выбрать соответствующий тип микроконтроллера. В базе поддерживаемые устройства сгруппированы в папки по фирмам-производителям.

Выбрав нужный микроконтроллер, необходимо нажать кнопку ОК. При этом в проекте организуется Target (целевая задача), которой по умолчанию присваивается имя Target 1. В составе целевой задачи автоматически создаётся так называемая Group (группа) – папка с именем Source Group 1. В эту папку включают исходные файлы проекта (см. окно проекта на рис. 10.2).

В группу можно включать исходные файлы в виде текста на языке Ассемблер и(или) Си, а также библиотечные и объектные файлы. При необходимости в проекте создают дополнительные группы, чтобы разделить файлы по видам или установить для них индивидуальные настройки.

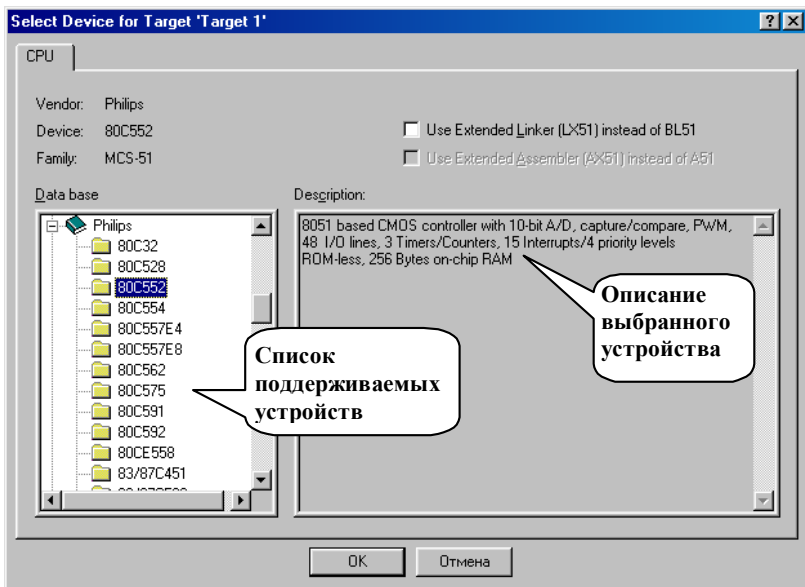



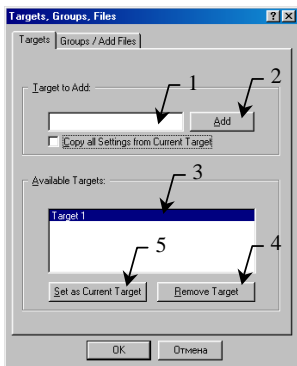
Рис. 10.3. Окно выбора устройства

Для создания нового файла в меню File нужно выбрать пункт New или нажать кнопку  панели File Toolbar. При этом откроется окно нового файла, которому по умолчанию присваивается имя Text1. После ввода текстовой информации файл необходимо сохранить через меню File пункт Save As...

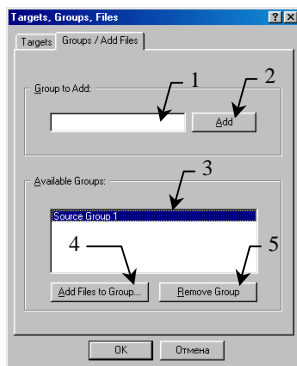
3. Организация структуры проекта

Организация структуры проекта производится через меню Project командой Targets, Groups, Files или через окно проекта Project Window по нажатию правой кнопки мыши на нужном элементе. При этом открывается список команд, одна из которых Targets, Groups, Files. Выбор этой команды открывает окно с двумя закладками: Targets и Groups/Files. Первая закладка (рис. 10.4, а) служит для добавления/удаления и определения текущей целевой задачи. С помощью второй закладки (рис. 10.4, б) для всех целевых задач добавляют/удаляют группы, а также в выбранную группу добавляют файлы.

Для добавления новой целевой задачи нужно ввести её имя в строку (рис. 10.4, а, указатель 1), а затем нажать кнопку Add (рис. 10.4, а, указатель 2). При этом имя отобразится в списке Available Targets (рис. 10.4, а, указатель 3).



а)



б)

Рис. 10.4. Окно организации структуры проекта

В проекте может быть активна только одна Target, которая выбирается из списка и отмечается нажатием кнопки Set as Current Target (рис. 10.4, а, указатель 5). Другой способ активизации целевой задачи заключается в выборе её из списка Select Target панели Build Toolbar (рис. 10.5, указатель 1).

Для удаления ненужных целевых задач нужно выделить левой кнопкой мыши соответствующее имя в списке Available Targets (рис. 10.4, а, указатель 3), а затем нажать кнопку Remove Target (рис. 10.4, а, указатель 4).

Для добавления новой группы нужно ввести её имя в строку (рис. 10.4, б, указатель 1), а затем нажать кнопку Add (рис. 10.4, б, указатель 2). При этом имя новой группы отобразится в списке Available Group (рис. 10.4, б, указатель 3).

Для удаления ненужных групп нужно выделить левой кнопкой мыши имя соответствующей группы в списке Available Group (рис. 10.4, б, указатель 3), а затем нажать кнопку Remove Group (рис. 10.4, б, указатель 5).

Для включения файлов в группу нужно выделить левой кнопкой мыши имя группы (рис. 10.4, б, указатель 3), а затем нажать кнопку Add Files to Group... (рис. 10.4, б, указатель 4). При этом откроется окно для добавления файлов, в котором нужно выбрать тип файла, указать путь к нему и нажать кнопку Add. Добавление не закрывает

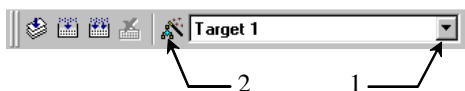


Рис. 10.5. Панель Build Toolbar

окно, что позволяет оперативно подключить несколько файлов. Закрытие производится кнопкой Close. Результат выполнения ваших действий можно проконтролировать в окне проекта (см. рис. 10.2).

Для удаления файла из группы нужно в окне проекта (см. рис. 10.2) нажать на нём правой кнопкой мыши. При этом откроется список команд, в котором следует выбрать команду Remove File.

4. Настройка параметров проекта

При настройке параметров проекта устанавливаются параметры целевой задачи, групп и файлов. Для групп и файлов устанавливаются свойства файлов и параметры компиляции. Настройка параметров групп и файлов может быть произведена индивидуально либо может быть общей и соответствовать параметрам целевой задачи.

Для отдельной целевой задачи можно выбрать свой тип микроконтроллера (см. рис. 10.3). Предварительно необходимо выделить в окне проекта (см. рис. 10.2) целевую задачу. Выбор типа микроконтроллера для активной целевой задачи осуществляется командой Select Device for Target из меню Project либо из окна, вызываемого щелчком правой кнопки мыши на имени целевой задачи в окне проекта (см. рис. 10.2). При этом на экран выводится окно выбора устройства (см. рис. 10.3), в котором следует выбрать соответствующий тип микроконтроллера.

Для открытия окна настройки параметров целевой задачи можно воспользоваться кнопкой Options for Target панели Build Toolbar (см. рис. 10.5, указатель 2). Окно настройки параметров (рис. 10.6 – 10.9) имеет 8 закладок, на каждой из которых можно устанавливать соответствующие значения для выбранной целевой задачи.

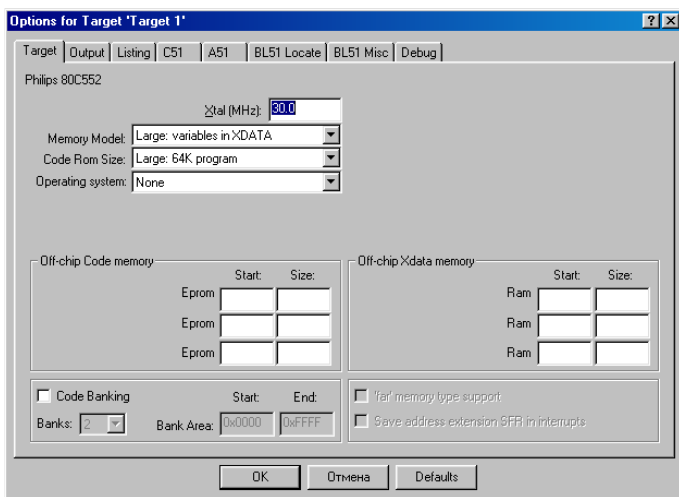


Рис. 10.6. Закладка Target окна Options for Target

Закладка Target (см. рис. 10.6) предназначена для следующего.

– Установка значения тактовой частоты микроконтроллера (Xtal) в МГц. Этот параметр используется при отладке программы в режиме симуляции в среде Keil μ Vision2.

– Выбор модели памяти для хранения данных (Memory Model).
Возможные варианты:

- Small (переменные размещаются во внутренней памяти);

- Compact (переменные размещаются в пределах одной страницы внешней памяти);

- Large (переменные размещаются во внешней памяти).

– Выбор размера программной памяти (Rom Code Size). Возможные варианты:

- Small (объём программы не превышает 2 кБ);

- Compact (объём программы не превышает 64 кБ, а подпрограмм – 2 кБ);

- Large (объём программы и подпрограмм не превышает 64 кБ).

– Использование операционной системы (ОС) и выбор её версии (Operation system). Возможные варианты:

- None – без ОС;

- RTX-51 Full – выбор полной версии многозадачной ОС реального времени для MCS-51;

- RTX-51 Tiny – выбор минимизированной версии ОС для микропроцессорной системы без внешней памяти.

– Определение областей адресов внешнего ПЗУ программ (Off-chip Code memory). Указывается начальный адрес и размер для трёх областей памяти.

– Определение областей адресов внешнего ОЗУ данных (Off-chip Xdata memory). Указывается начальный адрес и размер для трёх областей памяти.

Закладка Output (рис. 10.7) предназначена для выбора папки для размещения выходных файлов, их имени (указатель 1) и типа создаваемых файлов. По умолчанию имена файлов совпадают с названием проекта и размещаются в той же папке, что и сам проект.

Нажатие кнопки Create Executable позволяет получать исполняемые файлы. При этом установка флажка

– Debug Information добавляет в выходной объектный код информацию, необходимую для отладки программы;

– Browse Information делает возможным информационный поиск в программе;

– Create Hex File позволяет получать файл в формате Intel Hex для загрузки в микроконтроллер.

Нажатие кнопки Create Library позволяет получать библиотечные файлы.

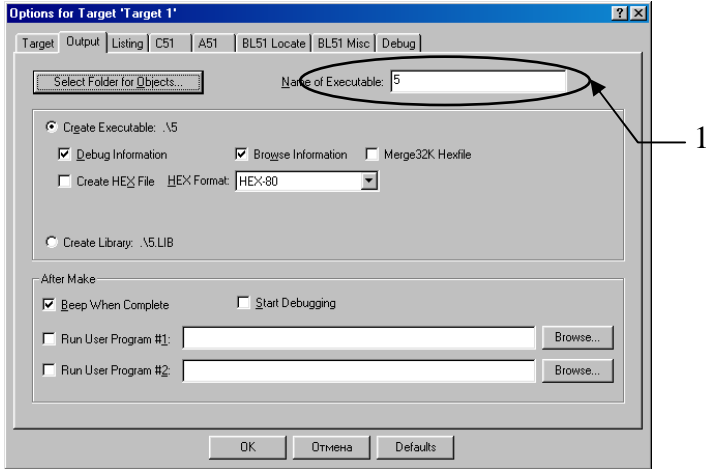


Рис. 10.7. Закладка Output окна Options for Target

После обработки проекта (After Make) можно назначить выполнение следующих действий:

- подача звукового сигнала;
- запуск отладчика;
- запуск программ по желанию пользователя.

Закладка Listing предназначена для настройки создания отчётов о выполнении компиляции и компоновки проекта.

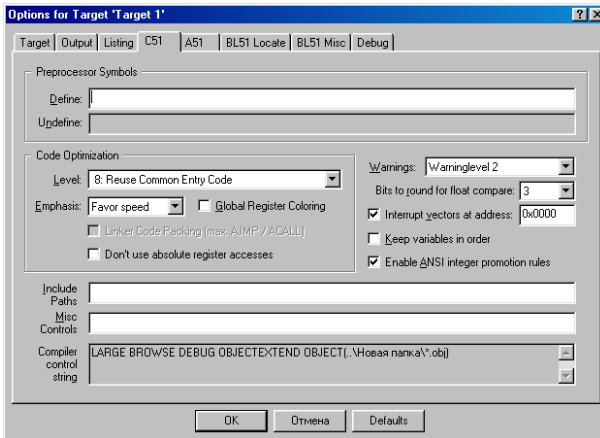


Рис. 10.8. Закладка C51 окна Options for Target

Закладка C51 (рис. 10.8) предназначена для настройки компилятора языка Си и позволяет:

- ввести символы препроцессора языка Си (директивы #define);
- настроить оптимизацию получаемого программного кода по уровню (Level) и выбрать акцент (Emphasis) между скоростью работы и объёмом программы;
- оптимизировать использование регистров микроконтроллера;
- определить тип и степень строгости предупреждений компилятора;
- установить число округляемых бит при сравнении чисел типа float с плавающей точкой;
- установить начальный адрес вектора прерываний;
- установить порядок размещения переменных в соответствии с последовательностью их объявления;
- разрешить приведение к типу integer переменных меньшего размера для обеспечения совместимости со стандартом ANSI;
- ввести дополнительный путь для директив #include;
- ввести специальные директивы для компилятора C51.

Закладки A51, BL51 Locate и BL51 Misc предназначены для настройки компилятора языка ассемблер и компоновщика.

Закладка Debug (рис. 10.9) предназначена для настройки отладчика. Позволяет осуществлять следующее.

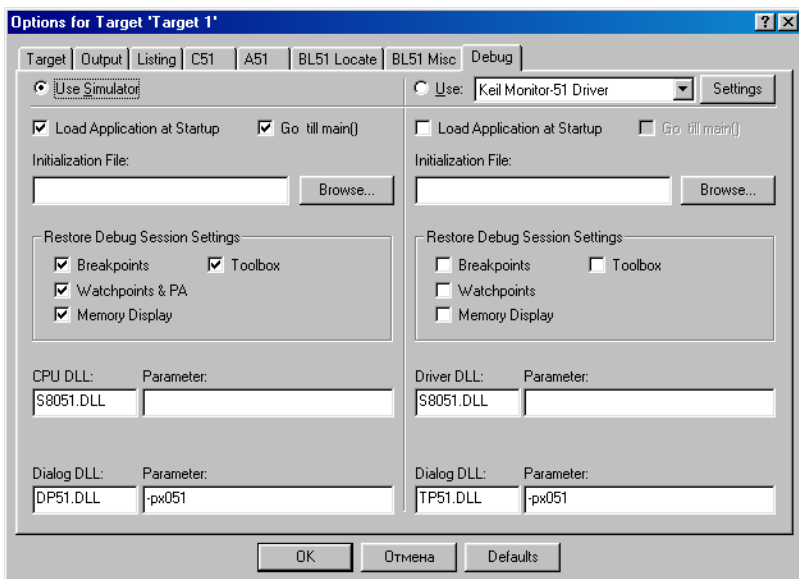


Рис. 10.9. Закладка Debug окна Options for Target

- Выбирать способ отладки:
 - отладка программы производится на компьютере, который имитирует работу микроконтроллера (Use Simulator);
 - отладка программы производится под управлением компьютера, подключённого к микроконтроллеру через последовательный интерфейс (Use) и использующего соответствующий драйвер (Keil Monitor-51 Driver).
 - Устанавливать начало отладки программы с метки main (Go till main ()).
 - Определять в строке Initialization File файл отладочных функций для имитации внешних сигналов.
 - Восстанавливать установки предыдущего сеанса отладки (Restore Debug Session Setting):
 - точки останова программы (Breakpoints);
 - точки наблюдения и анализатор выполнения программы (Watchpoints & PA);
 - дисплей памяти (Memory Display);
 - кнопки управления (Toolbox).

5. Настройка шрифтов проекта

Для корректного отображения букв русского алфавита в комментариях к программе часто требуется произвести настройку шрифтов проекта. Установка стиля, размера и цвета шрифтов, используемых в проекте, производится в окне Options (рис. 10.10), для вызова которого

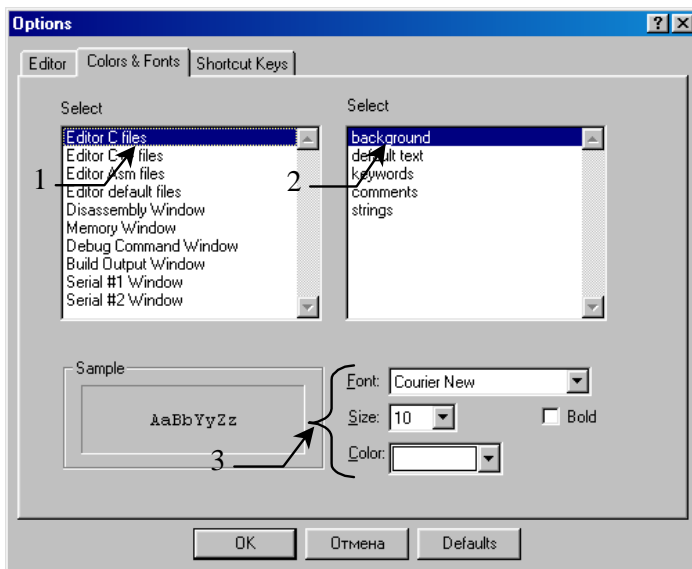


Рис. 10.10. Закладка Colors & Fonts окна Options

следует выбрать в меню View пункт Options. На закладке Colors & Fonts (рис. 10.10) можно указать параметры шрифтов, используемых в различных окнах проекта. Сначала выбирают окно (рис. 10.10, указатель 1), затем вид текста в этом окне (рис. 10.10, указатель 2) и устанавливают тип шрифта Font, его размер Size и цвет Color (рис. 10.10, указатель 3). Для сохранения установленных шрифтов следует нажать кнопку ОК.

6. Обработка проекта

Процесс обработки включает компиляцию (трансляцию) исходных файлов проекта, их компоновку и, если разрешено в установках, создание Hex-файла, предназначенного для загрузки в программную память микропроцессорной системы. Для этого служат команды меню Project, доступные также на панели Build Toolbar (см. рис. 10.5):



Translate – компиляция исходного файла проекта;



Build Target – компиляция только изменённых файлов проекта и их компоновка;




Rebuild All Target Files – компиляция всех исходных файлов проекта и их компоновка.

Результаты обработки проекта отображаются в окне выходных сообщений на закладке Build (см. рис. 10.2). Здесь выводятся сообщения об ошибках компиляции и компоновки, а также предупреждения. Двойной щелчок мыши на сообщении об ошибке компиляции позволяет получить подсказку о её местонахождении в исходном тексте. Возникающие ошибки не позволяют создать загружаемый файл, и выводится сообщение «Target not created». Если ошибки отсутствуют и в установках разрешено создание Hex-файла, то выводится сообщение «creating hex file from».

7. Отладка программы

Прежде чем готовая программа будет загружена в микроконтроллер, вы можете её опробовать и отладить в среде Keil μ Vision2. Один из способов отладки, симулятор (Simulator), удобен тем, что для него не требуется целевая микропроцессорная система, так как отладка программы производится на компьютере, который имитирует работу микроконтроллера.

Для отладки предварительно откомпилированной программы выбирают команду Start/Stop Debug Session из меню Debug или используют кнопку . При этом активизируется панель Debug Toolbar, а в окне проекта открывается закладка Regs для контроля содержимого внутренних регистров микроконтроллера.


При помощи кнопок панели Debug Toolbar оперативно осуществляется сброс микроконтроллера, запуск и остановка выполнения про-

граммы, выполнение программы в пошаговом режиме и реализуется ряд других возможностей среды μ Vision2.

Отладка программы начинается с запуска программы на выполнение. Последовательность выполнения команд можно контролировать в окне исходного текста программы на языке Си или в окне с программой на языке ассемблера. Как и в других интегрированных средах программирования, в μ Vision2 существует возможность устанавливать и снимать точки останова выполнения программы (Breakpoint).

Результат работы программы можно наблюдать в окнах:

- проекта на закладке Regs (содержимое системных регистров микроконтроллера);

- Memory Window (содержимое памяти, кнопка 

- Watch & Call Stack Window (значения переменных, кнопка 

Имеется возможность проверки работы программы с такими устройствами микроконтроллера, как система прерываний, порты ввода-вывода, последовательные порты, таймеры/счётчики, АЦП и др. Для этого нужно из пункта меню Peripherals выбрать соответствующую команду.

Кроме того, можно создавать отладочные функции, которые генерируют внешние прерывания, периодически обновляют сигналы на цифровых и аналоговых входах и подают данные на вход последовательного порта.

Порядок выполнения работы

1. Запустите программу Keil μ Vision2.
2. Создайте новый проект для микроконтроллера 80C552 фирмы Philips.

3. Скопируйте из директории **I:\микропроцессоры\Keil\ Examples for EB552** в директорию, где находится ваш проект, файлы:

Hello.c – исходного текста демонстрационной программы на языке Си;

Startup.a51 – системной программы на языке ассемблер;

Stdio1.h – модифицированной библиотеки функций стандартного ввода-вывода.

Примечание: целесообразно всем файлам и директориям присваивать имена, использующие только латинские буквы, цифры и символ подчёркивания «_» и начинающиеся с латинской буквы.

4. Включите в проект эти файлы.

5. Настройте проект под лабораторный микроконтроллер EB552 с эмулятором ПЗУ.

5.1. На закладке Target установите:

Memory Model – *Large*;

Rom Code Size – *Large*;

Off-chip Code memory: Start *0x8000*, Size *0x8000*;

Off-chip Xdata memory: Start *0x0000*, Size *0x8000*.

5.2. На закладке Output в области Create Executable установите флажок Create Hex File: *Hex-80*.

5.3. На закладке C51:

Level: 8;

Emphasis: *Favor speed*;

Interrupt vectors at address: *0x8000*.

5.4. На закладке Debug в области Use Simulator установите флажки Load Application at Startup и Go till main ().

6. Для настройки шрифтов проекта на вкладке Colors & Fonts окна Options выберите пункт Editor C files и нажмите кнопку ОК (см. рис. 10.10).

7. Ознакомьтесь с текстом демонстрационной программы [14, 15].

8. Обработайте проект с помощью команды Rebuild All Target Files и проконтролируйте результат её выполнения в окне выходных сообщений. При наличии ошибок проведите корректировку текста программы.

9. Ознакомьтесь с содержимым Hex-файла, созданным генератором OH51 в директории проекта.

10. Запустите отладчик и проверьте работу программы в пошаговом режиме.

Контрольные вопросы

1. Какие функции выполняет интегрированная среда μ Vision2?
2. Какие компоненты входят в состав интегрированной среды μ Vision2 и как они взаимодействуют между собой?
3. Поясните процесс разработки программного обеспечения в среде μ Vision2.
4. На каких языках можно писать программы в среде μ Vision2? Сравните эти языки по их возможностям и удобству использования.
5. Что называют проектом в среде μ Vision2?
6. Как организуется структура проекта?
7. Как производится настройка параметров проекта?
8. Как производится обработка проекта?
9. Какие средства отладки предоставляет разработчику программ среда μ Vision2?

ПЕРЕДАЧА ИНФОРМАЦИИ ПО ИНТЕРФЕЙСУ RS-232

Цель работы: получить представление о принципе работы последовательных интерфейсов.

Задание

Записать в эмулятор ПЗУ лабораторного контроллера программу, разработанную при выполнении практической работы 10, с использованием интерфейса RS-232 (COMPORT).

Методические указания

Обзор интерфейса rs-232

В современных сетях широко используются последовательные интерфейсы передачи данных. Эти интерфейсы предполагают поочередную передачу битов информации по одной линии. В отличие от параллельных интерфейсов, использующих для одновременной передачи нескольких битов соответствующее число проводников, последовательные интерфейсы обеспечивают снижение затрат на изготовление и монтаж кабельной продукции.

Изначально последовательный интерфейс RS-232 был известен как V24 и был принят в качестве основного для телеграфии Международным консультативным комитетом по телеграфии и телефонии. В 1962 г. Ассоциация промышленной электроники EIA (Electronics Industries Association) разработала рекомендации для производителей оборудования, назвав их «Рекомендованный стандарт 232».

Интерфейс RS-232 является максимально универсальным. Кодировать символы допускается битами в количестве от пяти до восьми; напряжения сигнала могут составлять от ± 3 до ± 25 В; предусмотрено 16 сервисных сигналов, использование которых не обязательно; допускается работа как в синхронном, так и асинхронном режиме передачи данных. Такая лояльность стандарта обеспечила его широкое распространение. В компьютерных сетях для реализации интерфейса используется COM-порт (Communication port). В промышленных сетях АСУ ТП для передачи измерительной информации и управляющих сигналов используются усовершенствованные интерфейсы RS-422, RS-455 и RS-485, принципы передачи информации в которых соответствуют RS-232. На территории России известна помехоустойчивая токовая реализация интерфейса RS-232, известная под названием С2 (Стык 2).

В персональных компьютерах и большинстве контроллеров интерфейс RS-232 полностью аппаратно реализован за счёт универсального асинхронного приёмопередатчика UART (Universal Asynchronous Receiver-Transmitter). UART одинаков для всех подобных интерфейсов (RS-485 и RS-422). В результате аппаратной реализации на приём и передачу данных тратится минимум программных ресурсов [19].

Приёмопередатчики UART разрабатываются по редакции RS-232C стандарта RS-232, выпущенной в 1969 г. EIA с учётом семилетнего опыта применения стандарта RS-232A/B. Так как интерфейс RS-232C универсален, то подразумевается его использование в различных приложениях и режимах. Для каждого приложения могут использоваться различные сочетания сигналов интерфейса, которые называются конфигурациями. Стандарт RS-232C предусматривает 13 стандартных конфигураций A–M и одну пользовательскую конфигурацию Z. Блоки UART работают по стандарту RS-232C в конфигурации D.

Для COM-порта компьютера используется 25-штырьковый разъём DB25r согласно стандарту RS-232C или 9-штырьковый разъём DE9r согласно стандарту TIA-574, разработанному Телекоммуникационной ассоциацией TIA (Telecommunications Industry Association) (рис. 11.1). Первоначально в COM-портах использовали разъёмы DB25r. В связи с тем, что большая часть сервисных сигналов RS-232C в UART не использовалась, фирма IBM стала использовать в своих компьютерах разъёмы DE9r с шестью сервисными сигналами.

В COM-порте реализованы следующие сигналы:

- информационные
 - RxD (Received Data) – приём данных;
 - TxD (Transmitted Data) – передача данных;
- сервисные
 - DTR (Data Terminal Ready) – готовность терминала к работе;
 - DSR (Data Set Ready) – готовность оконечного устройства к работе;
 - CTS (Clear to Send) – готовность оконечного устройства к передаче;
 - RTS (Request to Send) – запрос на передачу;
 - DCD (Data Carrier Detect) – обнаружен носитель информации;
 - RI (Ring Indicator) – сигнал звонка;
- общие
 - GND или SG (Signal Ground) – сигнальная земля (общий сигнальный провод порта, изолированный от корпуса);
 - PG (Protective Ground) – защитное заземление.

На рисунке 11.2 представлена временная диаграмма передачи по интерфейсу RS-232 двух символов, каждый из которых кодируется восьмью битами 0...7. Временная диаграмма представляет собой изменение информационных сигналов TxD и RxD во времени t .

Как видно из рис. 11.2, в исходном состоянии на линии передачи присутствует логическая единица ($-3...-25$ В). Передача символа начинается стартовым битом start, который всегда равен нулю, за ним через равные промежутки времени следуют восемь битов информации, начиная с младшего (0) и заканчивая старшим (7). Посылка заканчивается стоповым битом stop, который всегда равен единице.

Передача информации может сопровождаться **синхронизацией** отдельных битов специальными синхронизирующими импульсами. Для этого могут быть использованы специальные синхронизирующие линии или одна из информационных. Фронт синхронизирующего импульса (переход из 0 в 1) соответствует середине интервала времени от начала до окончания передачи бита. Принимающее устройство воспринимает состояние линии только в моменты изменения сигнала на синхронизирующей линии. Это позволяет с высокой точностью распознавать принимаемую информацию и обязательно используется в **синхронных** интерфейсах.

При передаче информации по интерфейсу RS-232 внешние синхронизирующие импульсы используются редко. Асинхронный приёмопередатчик UART в режиме передачи посылает данные, заботясь только о равенстве интервалов времени, а в режиме приёма сам подстраивается под поступающий сигнал. Для определения моментов времени, в которые необходимо считать состояние входной линии RxD, используется внутренний синхронизирующий сигнал, вырабатываемый схемой блока UART.

С момента поступления бита start, который всегда может быть определён, выдерживается интервал времени, зависящий от скорости передачи данных, и производится чтение бита start. Затем через равные интервалы времени по фронтам импульсов внутренней синхронизации, схема которой запускается битом start, производится чтение остальных битов.

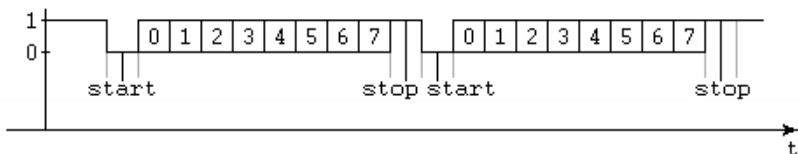


Рис. 11.2. Временная диаграмма передачи двух символов

Бит stop необходим для того, чтобы при любой комбинации нулей и единиц в передаваемых данных однозначно распознавался стартовый бит start. Если непосредственно после нуля седьмого бита передать бит start, то он не может быть распознан, так как в линии не произойдёт никаких изменений. Переход от единицы stop-бита к нулю start-бита, напротив, присутствует всегда, независимо от передаваемой информации (см. рис. 11.2). Схема внутренней синхронизации запускается каждый раз при обнаружении бита start. При этом она не реагирует на изменение состояния принимаемого сигнала при передаче битов 0...7, так как обнаружение стартового бита начинается только после десятого синхронизирующего импульса (приёма бита stop).

Метод, которым синхронизируются данные по стандарту RS-232, стал общепотребительным для всех асинхронных протоколов обмена данными.

Символы, передаваемые по интерфейсу RS-232, могут иметь размер от 5 до 8 бит. После передачи этих информационных битов может следовать бит **паритета**, служащий для обнаружения ошибки передачи битов данных. При необходимости может быть передано два stop-бита.

Использование бита паритета (Parity Control Bit) является простейшим способом обнаружения ошибок. Он позволяет определить возникновение ошибок в нечётном числе битов. При наличии ошибок в чётном числе битов их обнаружение будет невозможно. Режимы использования контрольного бита:

- N (None) – проверка на паритет не используется (бит отсутствует);
- E (Even) – проверка на чётность (передаваемый символ дополняется битом паритета так, чтобы количество единиц было чётным);
- O (Odd) – проверка на нечётность (передаваемый символ дополняется битом паритета так, чтобы количество единиц было нечётным);
- M (Mark) – бит всегда равен единице;
- S (Space) – бит всегда равен нулю.

Кроме синхронизации отдельных битов информации интерфейс RS-232 предусматривает синхронизацию обмена данными. В этом случае передающее и принимающее устройства учитывают текущие состояния друг друга и используют сервисные сигналы (DTR, DSR, CTS, RTS, DCD, RI). Такой режим называется режимом синхронизации обмена данными или Handshaking (рукопожатия). Если этот режим не применяется, линии сервисных сигналов могут быть использованы для целей, не связанных с передачей данных.

Для совместной работы приёмника и передатчика линии RxD и TxD соединяются перекрёстно: TxD одного устройства с RxD другого и наоборот (рис. 11.3). То же относится и к парам сигналов RTS-CTS и DTR-DSR. Кабели для интерфейса RS-232, которые устроены таким образом, называются **нуль-модемными**.

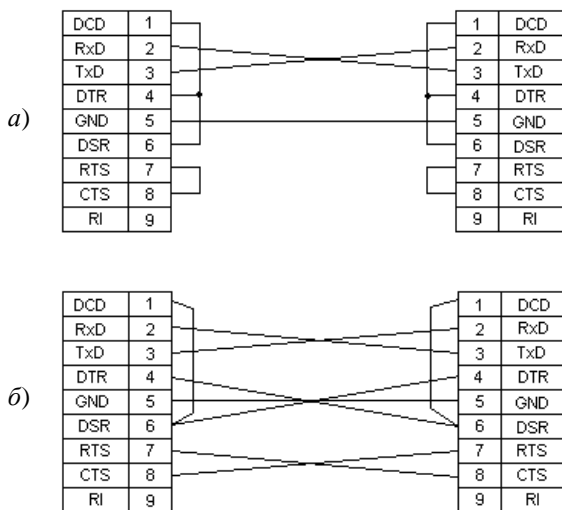


Рис. 11.3. Схемы нуль-модемных кабелей для режимов обмена данными без синхронизации (*a*) и с синхронизацией (*б*)

При таком соединении устройства соединяются между собой непосредственно, без использования модемов (модуляторов-демодуляторов), устройств, обеспечивающих передачу сигналов на большие расстояния [19].

Схема кабеля, предназначенного для использования интерфейса RS-232 без режима Handshaking, представлена на рис. 11.3, *a*. В этом случае сервисные сигналы не влияют на обмен данными. Так как режим синхронизации обмена может быть включён, часто сервисные сигналы замыкают внутри разъёма кабеля, обеспечивая возможность работы устройств (рис. 11.3, *a*). Схема нуль-модемного кабеля для использования аппаратного режима синхронизации представлена на рис. 11.3, *б*.

Стандартом RS-232 гарантируется передача информации на расстоянии до 15 м, но на практике это могут быть много большие величины. Максимальное расстояние зависит от скорости передачи, типа и сечения проводников кабеля и других факторов. Скорость обмена данными выбирается из ряда (50, 75, 110, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, 14 400, 19 200, 28 800, 38 400, 56 000, 57 600, 115 200, 128 000) бит/с.

Для передачи информации на большие расстояния ассоциации EIA и TIA разработали асинхронный интерфейс RS-485, принципы передачи данных в котором соответствуют интерфейсу RS-232. В ин-

терфейсе RS-485 для передачи и приёма данных используется одна витая пара проводов. Поэтому обеспечивается возможность реализации только полудуплексного режима. В одном сегменте сети на базе интерфейса RS-485 может быть установлено до 32 приёмопередатчиков. Максимальная длина одного сегмента составляет 1200 м. Стандарт приобрёл большую популярность и стал основой промышленных сетей, используемых для решения задач автоматизации технологических процессов и производств.

Программирование контроллера через comport

Выходной HEX-файл в формате Intel HEX, получаемый при обработке проекта в среде μ Vision2, содержит информацию, которая предназначена для загрузки в микроконтроллер (см. практическую работу 10). В структуре HEX-файла имеются данные (команды) и адреса их размещения в памяти.

Обычно программа для микроконтроллера полностью записывается в ПЗУ. Однако при использовании микроконтроллера в лабораторной практике загрузка программы в ПЗУ нецелесообразна в связи с ограниченным числом циклов стирание/запись. Лабораторный контроллер на основе отладочной платы EB552 снабжён эмулятором ПЗУ на основе ОЗУ объёмом 32 кБ.

Загрузка HEX-файла в эмулятор ПЗУ осуществляется с компьютера через последовательный интерфейс RS-232. Напряжения сигналов блока UART COM-порта компьютера находятся в диапазоне от минус 12 В до плюс 12 В. Напряжения входных и выходных сигналов микроконтроллеров семейства MCS-51 не должны выходить за пределы 0...5 В. Для согласования уровней в контроллере на плате EB552 установлена микросхема ADM202.

Загружаемый HEX-файл принимается контроллером посредством программы-загрузчика, которая записана в микросхему ПЗУ, установленную на плате контроллера.

После подачи напряжения питания на лабораторный контроллер или нажатия кнопки «RESET» на его верхней панели на дисплее отображается e-mail адрес фирмы-производителя КТЦ-МК и сообщение «> Boot waits for hex» (Загрузчик ожидает HEX-файл). Контроллер находится в режиме ожидания HEX-файл. После приёма файла в нижней строке дисплея появляется соответствующее сообщение и происходит автоматический переход на выполнение загруженной программы. Для выхода из программы в режим ожидания загрузки следует нажать кнопку «RESET».

Если в режиме ожидания загрузки, не загружая файл, нажать кнопку «SELECT» («INT0») или «ENTER» («T0»), открывается меню. Кнопка «SELECT» («INT0») позволяет просмотреть возможные альтернативы, а «ENTER» («T0») – выбрать необходимую. Возможны следующие режимы работы:

> Load Hex to Buffer – загрузка файла без автоматического выполнения;

> Start from 8000h – запуск на выполнение с адреса 8000h;

> View RAM, from 0000 – просмотр содержимого ОЗУ;

> View eROM, from 8000 – просмотр содержимого эмулятора ПЗУ;

> Check hardware – контроль ресурсов контроллера.

Для передачи HEX-файла через COM-порт удобно использовать командный файл с расширением *.bat следующего содержания:

mode COM1 96,N,8,1

copy NAME.hex COM1:

Первая строка настраивает следующие параметры работы порта COM1 компьютера:



– 96 – устанавливает скорость обмена данными 9600 бит/с;

– N (None) – проверка на паритет не используется;

– 8 – передаваемые символы состоят из восьми битов;

– 1 – передаётся один стоповый бит.

Вторая строка BAT-файла пересылает HEX-файл с именем NAME.hex в порт COM1. Во время выполнения BAT-файла на экран компьютера выводится сообщение с параметрами COM-порта.

Для удобства работы можно настроить запуск командного файла *.bat непосредственно из среды Keil μ Vision2. При этом создаётся инструмент пользователя с помощью команды Customize Tools Menu из меню Tools. Эта команда открывает окно (рис. 11.4), в котором с помощью кнопки  в поле, отмеченное указателем 1, вводится название нового инструмента (например, «Load»). Затем в поле Command указывается имя исполняемого файла и путь к нему. Это целесообразно сделать в режиме обзора, нажав кнопку  (рис. 11.4, указатель 2) и выбрав соответствующий файл. Когда в поле Command появится путь к файлу *.bat, следует нажать кнопку ОК.

Теперь в меню Tools появилась новая команда – инструмент пользователя Load. Выбор этой команды запускает командный файл *.bat, в результате чего происходит загрузка программы через COM-порт в контроллер.

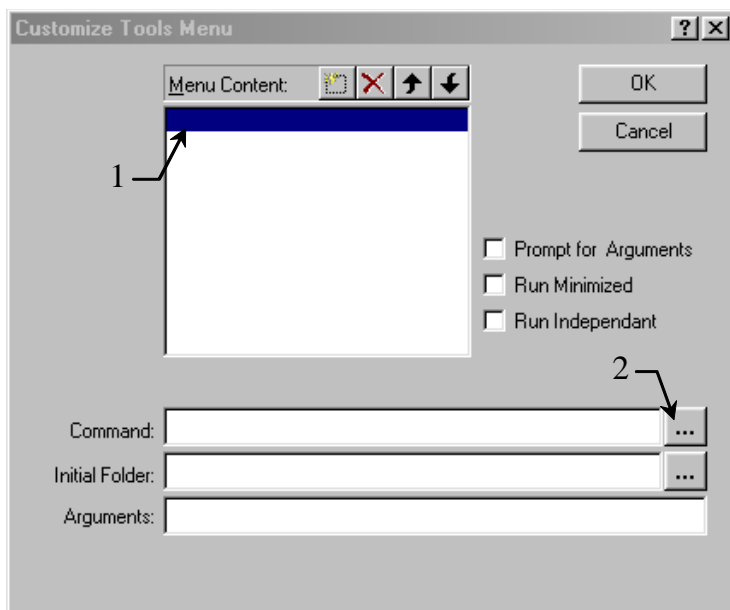


Рис. 11.4. Окно Customize Tools Menu

Порядок выполнения работы

1. Специальным кабелем подключите лабораторный контроллер к COM-порту компьютера.

2. Подключите к соответствующему разъёму контроллера блок питания или разъём интерфейса USB, являющийся альтернативным источником.

3. Запустите компьютер.

4. Проконтролируйте запуск программы-загрузчика в контроллере по наличию на его дисплее надписи «> Boot waits for hex». В случае необходимости нажмите кнопку «RESET» (верхняя кнопка).

5. Найдите в директории проекта среды μ Vision2, созданного при выполнении практической работы 10, файл с расширением *.hex.

6. Создайте в этой директории командный файл для загрузки HEX-файла в контроллер или скопируйте файл Zag.bat из директории **I:\микропроцессоры\Keil\Examples for EB552**. При использовании файла Zag.bat необходимо изменить имя загружаемого HEX-файла на то, которое соответствует HEX-файлу, полученному при выполнении практической работы 10. При создании и редактировании командного файла целесообразно использовать приложение «Блокнот».

7. Запустите командный файл и проконтролируйте его выполнение по сообщениям, выводимым на дисплей компьютера и контроллера. На дисплее контроллера в соответствии с программой, написанной на языке Си, должна появиться надпись «Hello, world!».

8. Сбросьте контроллер нажатием кнопки «RESET».

9. Запустите программу Keil μ Vision2.

10. Создайте инструмент пользователя для загрузки Hex-файла.

11. Загрузите программу в контроллер, используя инструмент пользователя из среды μ Vision2.

12. Модифицируйте программу на языке Си, изменяя сообщение, выводимое на дисплей контроллера, обработайте проект и загрузите программу в контроллер.

13. Модифицируйте программу, изменяя позицию вывода первого символа (функция `wrc(0x88)`); загрузите программу в контроллер и отметьте закономерности изменения положения выводимого на экран текста.

Контрольные вопросы

1. Почему необходимо знать принципы работы интерфейса RS-232?

2. Чем последовательные интерфейсы отличаются от параллельных?

3. Каким образом интерфейс RS-232 реализован в персональных компьютерах IBM?

4. Какие напряжения используются в интерфейсах RS-232 и TIA-574?

5. Какие режимы передачи информации по интерфейсу RS-232 возможны? В чём заключается их сущность?

6. Чем отличаются синхронные интерфейсы от асинхронных?

7. Каким образом осуществляется приём отдельных битов по интерфейсу RS-232?

8. Для чего предназначен бит паритета?

9. Как устроены кабели для интерфейса RS-232?

10. В чём заключаются основные недостатки интерфейса RS-232?

11. Назовите основные особенности интерфейса RS-485.

12. Как организован приём данных по интерфейсу RS-232 в лабораторном контроллере?

13. Как производится загрузка программного кода в контроллер?

14. Какие параметры устанавливаются при передаче информации через COM-порт?

ПРОГРАММИРОВАНИЕ КОНТРОЛЛЕРОВ НА ЯЗЫКЕ СИ

Цель работы: получить навыки разработки программного кода для микроконтроллеров на языке Си.

Задание

Разработать программу, реализующую счёт времени с отображением часов, минут и секунд.

Методические указания

Язык программирования Си предназначен для оперативного написания эффективных программ для ЭВМ. Одним из наиболее сложных вопросов при программировании реальных устройств является обеспечение их работы в реальном времени. Настоящая работа является подготовительной для изучения принципов работы системы прерываний и таймеров, используемых в микропроцессорных системах [14, 15].

При программной реализации счёта времени необходим ряд переменных, отвечающих за счёт отдельных единиц (дней, часов, минут, секунд, долей секунд). Для определения этих переменных целесообразно использовать тип данных `unsigned int` (целый беззнаковый тип), так как они не могут принимать отрицательные значения [14, 15]. Строка определения переменных имеет вид

`unsigned int <имена переменных через запятую>;`

Эту строку целесообразно разместить в начале программы. При этом переменные будут глобальными, т.е. доступными во всех функциях.

Для изменения значений переменных необходимо применять операторы инкремента. Например, оператор `sec++`; (аналогичный оператору `sec=sec+1`;) увеличивает значение переменной `sec` на единицу каждый раз при его выполнении.

При достижении переменными – счётчиками секунд, минут и часов определённых значений (60 или 24) их необходимо сбрасывать в нуль с одновременным инкрементом переменных, отвечающих за более крупные единицы. В языке Си оператор условия имеет вид

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

Если `<условие>` верно, то выполняется `<оператор1>`, иначе `<оператор2>` [14, 15]. Для объединения нескольких операторов используются операторные скобки `{}`. В языке Си используются следующие операции сравнения:

`>` – больше;

- < – меньше;
- == – равно;
- != – не равно;
- >= – больше или равно;
- <= – меньше или равно.

Для формирования программных временных задержек и бесконечных циклов удобно использовать оператор цикла `while` [14, 15]. Синтаксис этого оператора имеет вид

```
while(<условие>) <оператор>
```

<оператор> будет выполняться до тех пор, пока верно <условие>.

В языке Си любое значение целочисленной переменной, отличное от нуля, считается логической единицей, поэтому запись вида

```
while(1) {<совокупность операторов>}
```

реализует бесконечный цикл, в котором повторяется <совокупность операторов>, так как <условие> всегда истинно (1).

Для вывода на экран текста и значений переменных в языке Си используется функция `printf()`; стандартной библиотеки `stdio.h`. В проекте для лабораторного микроконтроллера используется модифицированная библиотека `stdio1.h`. Функция `printf()`; всегда имеет по крайней мере один аргумент – **строку формата**, представляющую собой набор символов, заключённый в кавычки. Строка формата может содержать **спецификаторы преобразования**. Функция `printf()`; сканирует строку формата и передаёт её символы на дисплей, пока не встретит спецификатор преобразования. В этом случае `printf()`; ищет дополнительный аргумент, который форматируется и выводится в соответствии со спецификацией. Вызов `printf()`; должен содержать столько дополнительных аргументов, сколько спецификаторов преобразования имеется в строке формата [14, 15].

Вывести значения часов, минут и секунд в общепринятом формате позволяет строка

```
printf("%2u:%02u:%02u", hour, min, sec);
```

где содержится три спецификатора преобразования, начинающиеся с символов «%» и разделённые символами «:». Символы «%» являются признаком начала спецификатора преобразования, символы «:» выводятся на дисплей как обычный текст. Число 2 обеспечивает вывод значений часов, минут и секунд не менее чем в две позиции, что обеспечивает одинаковое положение цифр на экране при любых значениях переменных. Число 0 обеспечивает вывод нуля в старший разряд минут и секунд, а символ «u» указывает на то, что выводимые числа имеют целый беззнаковый тип (`unsigned int`). При выводе на экран контроллера вместо спецификаторов преобразования помещаются значения соответствующих переменных `hour`, `min`, `sec`.

Порядок выполнения работы

1. Разработайте алгоритм решения поставленной в работе задачи.
2. Запустите программу Keil μ Vision2 и модифицируйте программу созданного в предыдущих работах проекта для решения задачи в соответствии с разработанным алгоритмом. Для сохранения исходного файла программы его можно скопировать в любую другую директорию вне среды μ Vision2. При модификации программы пользуйтесь методическими указаниями и всем исходным текстом программы.
3. Обработайте проект и произведите исправление допущенных ошибок.
4. Загрузите программу в контроллер и произведите её отладку.
5. Отметьте скорость выполнения программы и при необходимости добавьте блок задержки.
6. Подберите предельное значение переменной блока задержки для согласования хода часов программы с реальным временем.

Контрольные вопросы

1. Назовите основные элементы программы на языке Си.
2. Поясните принцип действия разработанной программы.
3. В чём заключаются основные недостатки разработанной программы?

Практическая работа 13

ИЗУЧЕНИЕ СИСТЕМЫ ПРЕРЫВАНИЙ МИКРОПРОЦЕССОРОВ

Цель работы: изучить принципы работы системы прерываний и таймеров микропроцессора, получить навыки разработки программ, действующих в реальном времени.

Задание

Разработать программу, реализующую счёт времени с отображением часов, минут и секунд с использованием прерываний по таймеру.

Методические указания

Большинство реальных задач, решаемых при помощи ЭВМ, сопровождаются событиями, которые требуют реакции микропроцессора только в определённые моменты времени. Например, при нажатии на клавишу необходимо вывести сообщение на экран; после окончания приёма байта информации по последовательному интерфейсу необходимо сохранить его в ячейке памяти. Если организовать непрерывный

опрос всех устройств, которые требуют немедленного реагирования, то это не обеспечит необходимой оперативности, так как в любой момент времени программа может реагировать только на одно событие, а также приведёт к чрезмерной загрузке микропроцессора и невозможности решения других задач. Для обеспечения своевременной реакции микропроцессора на различные события и освобождения времени на выполнение других операций в ЭВМ применяют **систему прерываний** [16 – 18].

При наступлении того или иного события, учитываемого системой прерываний, микропроцессор прекращает текущую работу и переходит к обработке этого события (**обработке прерывания**). После окончания обработки прерывания он возвращается в ту точку программы, из которой ушёл при наступлении события прерывания. Адрес ячейки программной памяти, по которому переходит микропроцессор при наступлении события прерывания, называется **вектором прерывания**. С этого адреса начинается **подпрограмма обработки прерывания** (в языке Си – **функция обработки прерывания**). Эта подпрограмма выполняет действия, необходимые для обеспечения требуемой реакции микропроцессора на событие. Поскольку подпрограммы обработки прерываний, как правило, имеют небольшое количество действий, микропроцессор большую часть времени свободен и может выполнять любую полезную работу, в том числе и обработку других прерываний.

Типичным примером, в котором необходимо использование системы прерываний, является обеспечение хода часов в режиме реального времени. Программа, созданная в практической работе 12 без использования системы прерываний, не обеспечивала необходимой точности счёта времени и загружала микропроцессор бесполезной работой в блоке задержки. При этом если обеспечить точный ход часов теоретически возможно, то освободить процессор для выполнения каких-либо других действий без использования системы прерываний довольно сложно и крайне нерационально. Выполнение любого действия, не входящего в основной цикл программы, приведёт к замедлению хода часов.

Таймеры/счётчики T/C0 И T/C1

Для формирования заданных временных интервалов в ЭВМ используют специальные устройства, которые выполняют функцию счёта (например, инкремент `sec++`;) аппаратно и независимо от текущего выполняемого микропроцессором действия. Такие устройства называются **таймерами/счётчиками**, так как кроме счёта времени могут работать в режиме счёта каких-либо событий.

В базовой части микроконтроллеров семейства MCS-51 имеются два таймера/счётчика Т/С0 и Т/С1. Микроконтроллер РСВ80С552, используемый в лабораторном контроллере, дополнительно снабжён ещё двумя таймерами [16 – 18].

Возможные режимы работы таймеров/счётчиков базовой части и значения битов для их выбора указаны в табл. 13.1. Блоки Т/С0 и Т/С1 являются 16-разрядными. С их помощью можно подсчитать максимальное количество импульсов, равное $2^{16} = 65536$ (в режиме 1 при $M0 = 1$ и $M1 = 0$). Шестнадцать разрядов разделены на два равных 8-битных регистра TL (Timer Low – младший байт таймера) и TH (Timer High – старший байт таймера), соединение которых различными способами обеспечивает возможность работы в разных режимах. Символ «х» в обозначении регистровых пар TLх и THх означает, что соответствующая информация относится как к Т/С0, так и к Т/С1.

Для формирования прерываний через заданные равные интервалы времени используется режим 2 ($M0 = 0$ и $M1 = 1$) с автоперезагрузкой. Схема, реализующая этот режим работы таймера/счётчика, представлена на рис. 13.1. В таблице 13.2 указано назначение битов регистра TMOD режимов работы таймеров/счётчиков.

Основным элементом таймера/счётчика является восьмиразрядный счётчик TLх, выполняющий счёт импульсов, поступающих на его вход, в двоичной форме. Максимальное число импульсов, которое может быть подсчитано без переполнения, составляет 255 (11111111 в двоичной системе).

13.1. Режимы работы таймеров микроконтроллеров семейства MCS-51

M1	M0	Режим работы
0	0	TLх работает как 5-битный предделитель, THх – в режиме 8-разрядного таймера/счётчика
0	1	16-битный таймер/счётчик. THх и TLх включены последовательно
1	0	8-битный таймер/счётчик с автоперезагрузкой. THх хранит значение, которое перезагружается в TLх каждый раз при его переполнении
1	1	Т/С1 останавливается. У таймера/счётчика Т/С0 регистр TL0 работает как 8-битный таймер/счётчик, а его режим определяется управляющими битами таймера Т/С0. TH0 работает только как 8-битный таймер, а его режим определяется управляющими битами таймера Т/С1

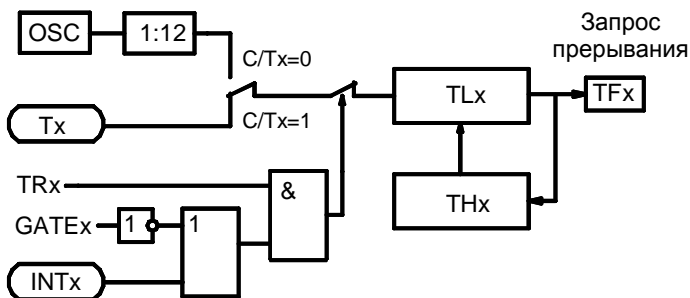


Рис. 13.1. Схема таймеров/счётчиков T/C0 и T/C1 в режиме 2

Поступление следующего, 256-го импульса вызывает переполнение счётчика TLx, он сбрасывается в состояние 00000000₂, а бит TFX (флаг прерывания от таймера) устанавливается (TFx = 1). В режиме 2 переполнение счётчика TLx вызывает не только установку флага TFX, но и загрузку двоичного числа, хранящегося в регистре Tnx, в счётчик TLx в качестве начального значения (см. табл. 13.1). Таким образом, если Tnx = 0, то флаг TFX устанавливается один раз за 256 импульсов, если Tnx = 1, то один раз за 255 импульсов и т.д., при Tnx = 255 флаг TFX будет установлен непрерывно [16 – 18].

13.2. Назначение битов регистра TMODE режимов работы таймеров/счётчиков

Символ	Разряд	Имя и назначение
GATE	TMOD.7 для T/C1 TMOD.3 для T/C0	Управление блокировкой. Если бит установлен (GATEx = 1), то счёт разрешён до тех пор, пока INTx = 1 и TRx = 1. Если бит сброшен (GATEx = 0), то счёт разрешается, как только TRx = 1
C/T	TMOD.6 для T/C1 TMOD.2 для T/C0	Бит выбора режима таймера или счётчика событий. Если бит сброшен (C/Tx = 0), то выбран режим таймера, иначе (C/Tx = 1) – режим счётчика внешних импульсов на входе Tx
M1	TMOD.5 для T/C1 TMOD.1 для T/C0	Режим работы (см. табл. 13.1)
M0	TMOD.4 для T/C1 TMOD.0 для T/C0	

При выборе режима счётчика событий ($C/Tx = 1$) вход счётчика TLx подключается к выводу Tx микроконтроллера, на который можно подать сигнал от внешнего устройства. Подсчитав число импульсов, поступивших на вход Tx за заданный интервал времени, можно определить их частоту.

В режиме таймера ($C/Tx = 0$) на вход счётчика TLx поступают импульсы с **тактового генератора** OSC с периодом, равным длительности **машинного цикла** контроллера. Все действия, выполняемые микропроцессором, синхронизируются импульсами тактового генератора. Длительность машинного цикла определяется временем выполнения одной из простых команд. Более сложные команды выполняются за несколько машинных циклов (1...4). В микроконтроллерах семейства MCS-51 машинному циклу соответствует 12 импульсов тактового генератора, поэтому на рис. 13.1 формально изображён делитель частоты на 12 (1:12).

Частота тактового генератора стабилизируется за счёт применения **кварцевого резонатора**, представляющего собой помещённый в корпус кристалл кварца, частота собственных колебаний которого постоянна и с высокой точностью соответствует номинальной. В лабораторном контроллере применяется кварцевый резонатор с номинальной частотой 11,0592 МГц. Кварцевый резонатор с обозначением номинальной частоты можно увидеть на плате контроллера через его верхнюю панель. При номинальной частоте 11,0592 МГц частота импульсов на выходе делителя 1:12 составит $f = 921,6$ кГц, а период, равный длительности машинного цикла, $T = 1,08507$ мкс.

Поскольку частота импульсов тактового генератора постоянна, в режиме таймера ($C/Tx = 0$) флаг TFx прерывания от таймера будет устанавливаться через равные промежутки времени t_{TF} , зависящие только от значения, записанного в регистр THx . Для обеспечения наибольших интервалов времени непрерывного выполнения основной программы целесообразно выбирать значения t_{TF} наибольшими, а THx – наименьшими. При $THx = 0$ величина t_{TF} составит $256T$ или $277,778$ мкс.

Работа таймера осуществляется под управлением битов $GATEx$ и TRx . Бит TRx находится в регистре $TCON$, предназначен для оперативного программного пуска и останова таймера и имеет собственный адрес ($TR1 = 0x8E$, $TR0 = 0x8C$, где символы «0x» – признак числа в шестнадцатеричной системе). В соответствии с табл. 13.2 для запуска таймера при решении задачи формирования равных интервалов времени необходимо установить $GATEx = 0$ и $TRx = 1$. Режим $GATEx = 1$ предназначен для измерения длительности импульсов на выводе $INTx$ микроконтроллера.

Прерывания по таймеру

Установка флага TFx приводит к тому, что микропроцессор отрывается от работы по основной программе и переходит на подпрограмму обработки прерывания по таймеру T/Cx, размещённую в ПЗУ по адресу 0x000B для T/C0 или 0x001B для T/C1. Для возврата в нужную точку основной программы после выполнения подпрограммы обработки прерывания происходит сохранение адреса текущей выполняемой команды основной программы в стеке [16 – 18].

При выполнении отдельных частей программы бывает необходимо отключить возможность прерывания микропроцессора. Для этого предназначен регистр IEN0 масок прерываний (табл. 13.3). Все доступные биты этого регистра устанавливаются и сбрасываются программно. Для разрешения соответствующего прерывания бит устанавливается, для запрета – сбрасывается. Для разрешения прерывания, например, только от T/C0, необходимо, чтобы ET0 = 1 и EA = 1, а остальные биты были сброшены. В соответствии со столбцом «Разряд» табл. 13.3 в двоичной системе получим IEN0 = 10000010₂, а в шестнадцатеричной IEN0 = 0x82. Строку IEN0 = 0x82; необходимо записать в начале функции main() программы на языке Си для конфигурации системы прерываний. Кроме регистра IEN0 масок прерываний существует также регистр IP0 приоритетов прерываний. Поскольку в настоящей работе используется только один источник прерываний, его использование нецелесообразно.

13.3. Регистр масок прерываний IEN0

Символ	Разряд	Имя и назначение
EA	IEN0.7	Общее разрешение прерываний. Разрешение/запрет всех прерываний независимо от состояний IEN0.4...IEN0.0
EAD	IEN0.6	Разрешение прерывания от АЦП
ES1	IEN0.5	Разрешение прерывания от модуля интерфейса I ² C
ES0	IEN0.4	Разрешение прерывания от UART
ET1	IEN0.3	Разрешение прерывания от таймера T/C1
EX1	IEN0.2	Разрешение внешнего прерывания INT1
ET0	IEN0.1	Разрешение прерывания от таймера T/C0
EX0	IEN0.0	Разрешение внешнего прерывания INT0

Прерывания от таймера в режиме 2 с автоперезагрузкой могут происходить не реже чем через $t_{TF} = 277,778$ мкс. Увеличением значения в регистре ТНх этот интервал можно только сократить. В соответствии с заданием к практической работе необходимо обеспечить увеличение переменной единиц секунд с интервалом в 1 с. Для осуществления этой задачи необходимо реализовать программный счётчик переполнений таймера, максимальное значение которого определяется интервалом времени t_{TF} . Его целесообразно разместить в подпрограмме обработки прерываний.

На рисунке 13.2 представлена блок-схема алгоритма счёта времени с использованием таймера и системы прерываний. Алгоритм состоит из двух непосредственно не связанных частей: подпрограммы обработки прерываний (рис. 13.2, *а*) и основной части (рис. 13.2, *б*). Инкремент счётчика секунд осуществляется в подпрограмме обработки прерываний, если программный счётчик переполнений таймера (переменная *time*) превышает максимальное значение. В основной части программы выполняется только проверка достижения счётчиками секунд, минут и часов максимальных значений и вывод на экран.

В этом алгоритме так же, как и в программе к предыдущей практической работе, присутствует дополнительная переменная *time*, обеспечивающая счёт более мелких единиц времени, чем секунды. Однако инкремент этой переменной производится не непрерывно, а лишь один раз в десятки или сотни машинных циклов контроллера. Всё остальное время микропроцессор может выполнять любую другую работу. В этом и состоит цель использования системы прерываний.

В языке Си определение функции обработки прерывания выглядит следующим образом.

```
void <имя функции>(void) interrupt N
{
    тело функции
}
```

Имя функции может быть любым незарезервированным словом языка Си. Вызывать эту функцию не следует ни в какой части программы. Она будет выполнена автоматически при возникновении события прерывания. По этой же причине функция не возвращает и не принимает аргументов (тип *void*). Номер *N* источника прерывания совпадает с номером соответствующего бита в регистре IEN0 масок прерываний (см. табл. 13.3).

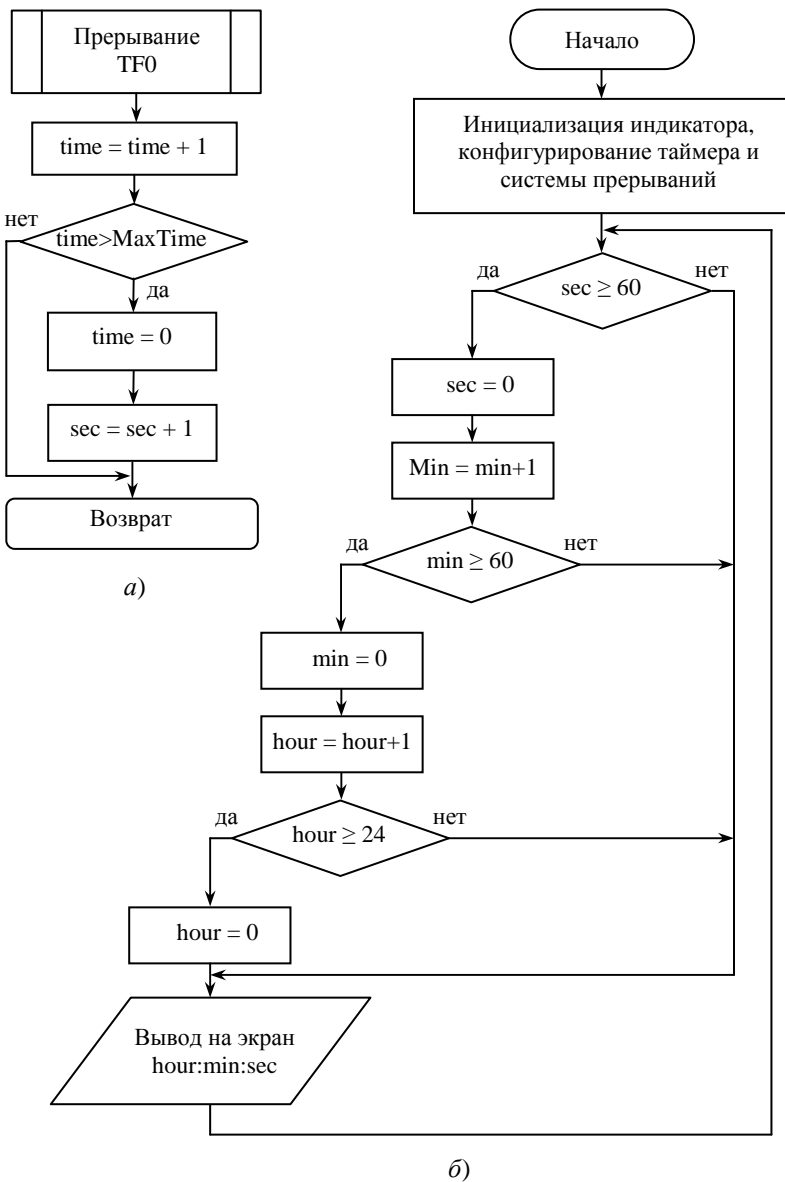


Рис. 13.2. Блок-схема алгоритма счёта времени с использованием таймера:
a – алгоритм обработки прерываний; *б* – основная часть алгоритма

Порядок выполнения работы

1. Разработайте алгоритм решения поставленной в работе задачи.
2. Ознакомьтесь с именами регистров специального назначения (SFR), используемыми в среде μ Vision2 для микроконтроллера PCB80C552. Для этого откройте файл \keil\C51\INC\Philips\REG552.H.
3. Запустите программу Keil μ Vision2 и модифицируйте программу созданного в предыдущих работах проекта для решения задачи в соответствии с разработанным алгоритмом.
4. Обработайте проект и произведите исправление допущенных ошибок.
5. Загрузите программу в контроллер и произведите её отладку.

Контрольные вопросы

1. Для чего в вычислительной технике используют систему прерываний?
2. В чём заключается принцип работы микропроцессорной системы с использованием системы прерываний?
3. Что называют вектором прерывания?
4. В чём преимущества программы, использующей систему прерываний?
5. Для чего в вычислительной технике используют таймеры?
6. Для чего в вычислительной технике используют счётчики?
7. В каких режимах могут работать таймеры/счётчики микроконтроллеров семейства MCS-51?
8. Как работают таймеры микроконтроллеров семейства MCS-51 в режиме автоперезагрузки?
9. Каким образом производится настройка таймеров/счётчиков?
10. Как рассчитать значение, которое необходимо загрузить в регистр THx для формирования равных временных интервалов заданной длительности?
11. Как устроена система прерываний микроконтроллеров семейства MCS-51?
12. Для чего в алгоритме счёта времени используется переменная time?
13. Для чего в среде μ Vision2 используется библиотечный файл REG552.H?

ИЗУЧЕНИЕ УСТРОЙСТВА КЛАВИАТУРЫ ЭВМ

Цель работы: изучить способ динамического опроса матричной клавиатуры.

Задание

Разработать программу для отображения кода нажатой клавиши матричной клавиатуры на дисплее лабораторного контроллера.

Методические указания

Для ручного ввода информации в ЭВМ используются различные коммутационные элементы: переключатели и кнопки. На их основе строятся клавиатуры, которые могут состоять более чем из ста кнопок.

Простейшая клавиатура может состоять из нескольких кнопок, подключённых к такому же количеству входов порта ввода ЭВМ. Если это 8-разрядный порт, то к нему можно подключить 8 кнопок, как это показано на рис. 14.1.

В исходном состоянии кнопки не нажаты и на всех входах порта присутствует уровень логической единицы благодаря резисторам R1...R8, соединённым с напряжением питания +5 В. При нажатии любой кнопки происходит замыкание соответствующего входа порта с корпусом схемы и на нём появляется уровень логического нуля. Для того чтобы ЭВМ могла определить факт нажатия одной или нескольких кнопок клавиатуры, ей необходимо осуществить ввод информации с порта. Если получены все единицы (шестнадцатеричный код 0xFF), то ни одна кнопка не нажата. Если один или несколько бит содержат логические нули, то нажаты соответствующие кнопки. Для идентификации нажатых кнопок необходима дополнительная программная обработка вводимой информации.

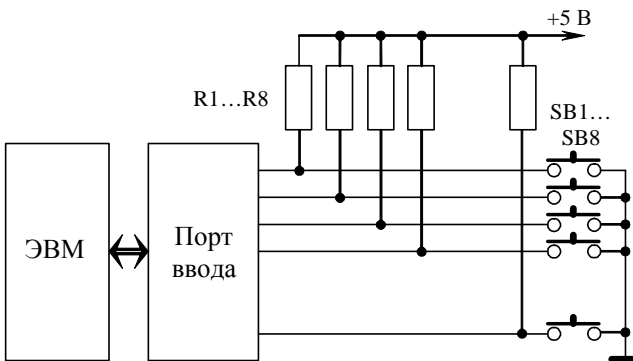


Рис. 14.1. Схема подключения простейшей клавиатуры

В рассмотренной схеме входные линии портов используются не-экономно. Можно повысить эффективность использования линий портов, если применить матричную организацию и динамический способ опроса (сканирование) кнопок клавиатуры.

На рисунке 14.2 представлена схема подключения 12-кнопочной матричной клавиатуры К1 (4×3 кнопки) к порту ввода-вывода P4 микроконтроллера РСВ80С552.

Для построения матрицы проводники располагают в виде сетки, в узлах которой размещают кнопки SB1...SB12. При нажатии кнопки замыкаются соответствующие линии столбцов (P4.0...P4.2) и строк (P4.3...P4.6). Для определения факта нажатия и идентификации нажатой кнопки контроллер осуществляет сканирование столбцов позиционным кодом и ввод информации со строк. При этом нажатая клавиша определяется программой по номерам строки и столбца. Столбцы P4.0...P4.2 используются как выходы, а строки P4.3...P4.6 – как входы. Такой способ опроса клавиатуры называется динамическим.

В процессе сканирования на каждый из выходов P4.0...P4.2 поочередно выводится логический 0 (рис. 14.3). На входах P4.3...P4.6 порта P4 изначально поддерживается уровень логической 1 за счёт «подтяжки» внутренними резисторами к источнику питания. Следовательно, если ни одна кнопка не нажата, то при вводе с порта P4 на этих входах будут получены единицы. Если же одна из кнопок будет нажата, то через замкнутый контакт логический 0, выводимый при сканировании на соответствующий столбец матрицы, попадёт на соответствующую строку. При вводе с порта P4 на входе, подключённом к этой строке, будет получен логический 0.

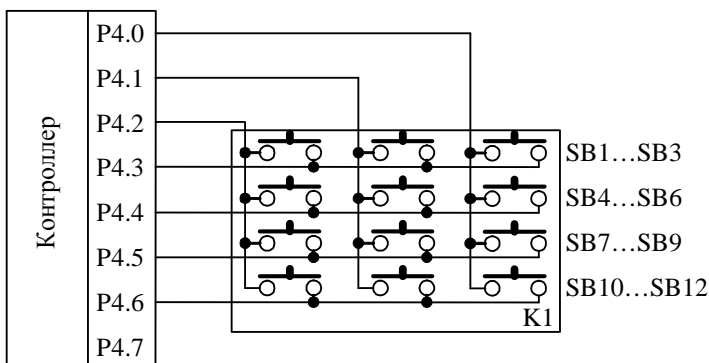


Рис. 14.2. Схема подключения матричной клавиатуры к порту контроллера

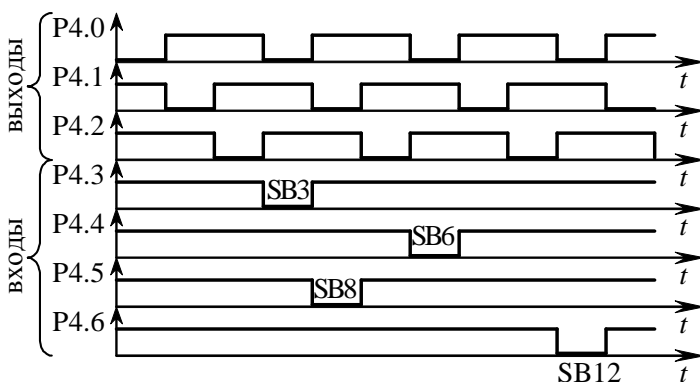


Рис. 14.3. Временная диаграмма работы матричной клавиатуры

Координаты нажатой кнопки определяются номером сканируемого столбца и номером строки, в которой получен логический 0. На рисунке 14.3 показаны сигналы на входных линиях порта P4 при нажатии кнопок во время сканирования клавиатуры. Нажатие кнопки SB3 (см. рис. 14.2) вызывает появление нуля на входе P4.3 при сканировании столбца, подключённого к выходу P4.0. По аналогии другие кнопки будут вызывать появление следующих сигналов (рис. 14.2, 14.3):

- SB6 – P4.4 = 0 при P4.0 = 0;
- SB8 – P4.5 = 0 при P4.1 = 0;
- SB12 – P4.6 = 0 при P4.0 = 0.

Алгоритм сканирования матрицы и отображения кода нажатой клавиши на индикаторе поясняет блок-схема, представленная на рис. 14.4.

В блоке 1 инициализируется переменная SCAN для опроса третьего столбца матрицы. В блоке 2 производится вывод значения переменной SCAN в порт P4, а в блоке 3 – ввод с порта P4 и присвоение полученного значения переменной KEY.

В блоке 4 выполняется проверка факта нажатия кнопки в опрашиваемом столбце клавиатуры. Если ни одна кнопка не нажата, то переменная KEY в двоичных разрядах 3...6 будет содержать логические единицы, и проверка даст отрицательный результат. Если же хотя бы одна кнопка нажата, то переменная KEY в соответствующем двоичном разряде будет содержать логический нуль, следовательно, проверка даст положительный результат.

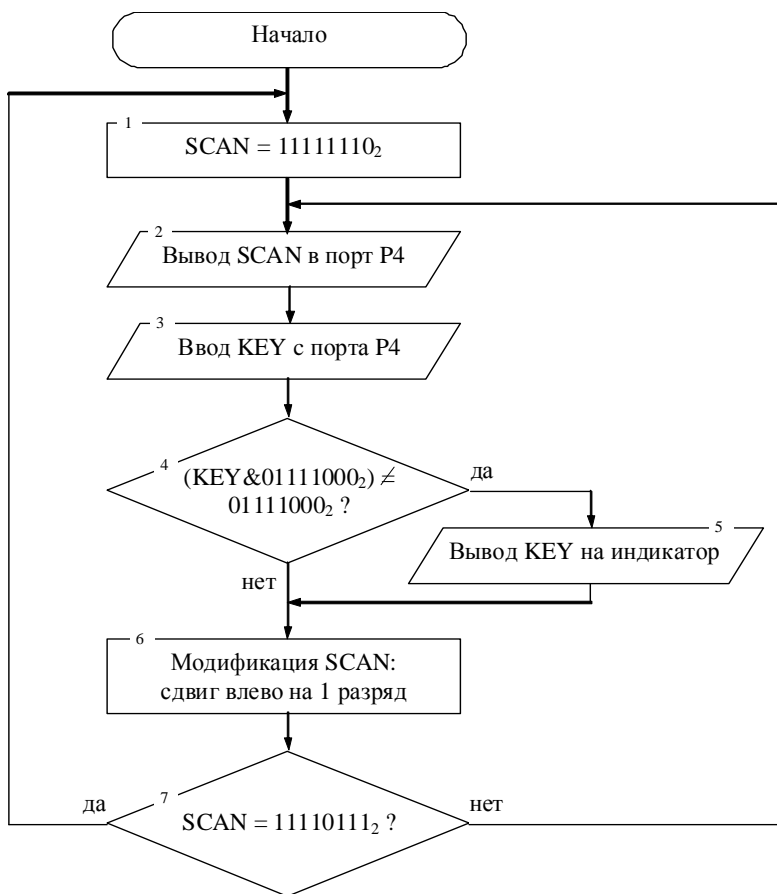


Рис. 14.4. Блок-схема алгоритма динамического опроса клавиатуры

При обнаружении факта нажатия кнопки выполняется блок 5, в котором производится вывод кода этой кнопки на индикатор. Переменная KEY содержит информацию как о состояниях входных линий P4.3...P4.6, так и о состоянии выходных – P4.0...P4.2, поэтому её значение позволяет однозначно идентифицировать нажатую клавишу. При отсутствии нажатия клавиши в соответствующем столбце блок 5 пропускается.

Блок 6 осуществляет модификацию переменной SCAN путём сдвига влево для сканирования следующего столбца.

В блоке 7 производится проверка окончания сканирования последнего столбца. Если нуль появился в третьем разряде переменной

SCAN, это означает, что просканирован третий столбец и следует перейти к сканированию первого. Для этого в блоке 1 вновь происходит инициализация переменной SCAN. Если условие в блоке 7 не выполняется, то модифицированное в блоке 6 значение переменной SCAN используется для опроса следующего столбца (2-го или 3-го) в блоках 2 и 3.

При написании программы на языке Си необходимо использовать шестнадцатеричную запись чисел. Для сдвига влево на один разряд целесообразно использовать оператор $SCAN=SCAN<<1$; или в сокращённой форме $SCAN<<=1$;. При выполнении этой операции в младший разряд поступает нуль, поэтому непосредственно после её выполнения необходимо заполнить этот разряд единицей, например применением оператора инкремента.

Порядок выполнения работы

1. Запустите программу Keil μ Vision2 и модифицируйте программу созданного в предыдущих работах проекта для решения поставленной задачи.
2. Обработайте проект и произведите исправление допущенных ошибок.
3. Подключите контроллер к персональному компьютеру.
4. Подключите плату внешних устройств к контроллеру.
5. Подключите к плате внешних устройств клавиатуру.
6. Загрузите разработанную программу в контроллер и произведите её отладку.

Контрольные вопросы

1. Как подключаются кнопки к портам ввода-вывода ЭВМ?
2. Почему активным входным сигналом является логический нуль?
3. В чём преимущества матричной клавиатуры перед линейной?
4. Поясните принцип работы матричной клавиатуры.
5. Поясните алгоритм динамического опроса матричной клавиатуры.
6. С какой целью используется маска для вводимого с порта значения переменной KEY?
7. Можно ли производить сканирование матричной клавиатуры по строкам и если «да», то какой вариант выгоднее?

ИСПОЛЬЗОВАНИЕ ДИСКРЕТНЫХ СИГНАЛОВ В ЗАДАЧАХ УПРАВЛЕНИЯ

Цель работы: изучить принцип управления блоками динамического объекта при помощи дискретных сигналов.

Задание

Реализовать включение и выключение внешних устройств при нажатии соответствующих клавиш на клавиатуре.

Методические указания

В объектах управления любой природы встречаются **аналоговые** и **дискретные** величины. Аналоговая величина изменяется непрерывно, на любом интервале возможно бесконечное множество её значений. Дискретная величина имеет конечное множество фиксированных значений. Чаще всего пользуются дискретными величинами, принимающими два значения, например «открыто» – «закрыто», «включено» – «выключено», «соответствует» – «не соответствует» и т.п.

На рисунке 15.1 представлена схема автоматизированной системы контроля и управления технологическим процессом (АСУ ТП) ректификации этилового спирта на базе ЭВМ. На схеме изображена нижняя часть ректификационной колонны.

В нижней части ректификационной колонны, называемой кубом, находится кубовая или лютерная жидкость. Для обеспечения качества получаемого в процессе ректификации продукта необходимо поддерживать постоянными величины температуры θ и уровня L лютерной жидкости.

Для стабилизации температуры θ используется контур регулирования, состоящий из первичного измерительного преобразователя (ПИП) ТЕ, модуля аналогового ввода МАВв на основе аналого-цифровых преобразователей (АЦП), модуля аналогового вывода МАВ на основе цифро-аналоговых преобразователей (ЦАП) и исполнительного устройства на основе регулирующего клапана К1. ПИП температуры, например термометр сопротивления, преобразует изменение величины θ в изменение сопротивления R . МАВв преобразует сопротивление в напряжение U_θ , а затем при помощи АЦП в двоичный цифровой код N_θ . ЭВМ использует полученный код N_θ для определения управляющего воздействия на клапан К1. Изменяя код N_{K1} на входе в МАВ, преобразующий этот код в напряжение U_{K1} , ЭВМ может управлять степенью открытия клапана К1. Если температура θ лютерной жидкости ниже заданной θ_0 , ЭВМ открывает клапан К1, и в рубашку теплообменника поступает большее количество греющего пара, в результате чего температура θ начинает возрастать. Аналогично при $\theta > \theta_0$ ЭВМ закрывает клапан К1, что обеспечивает понижение температуры θ до заданного значения θ_0 .

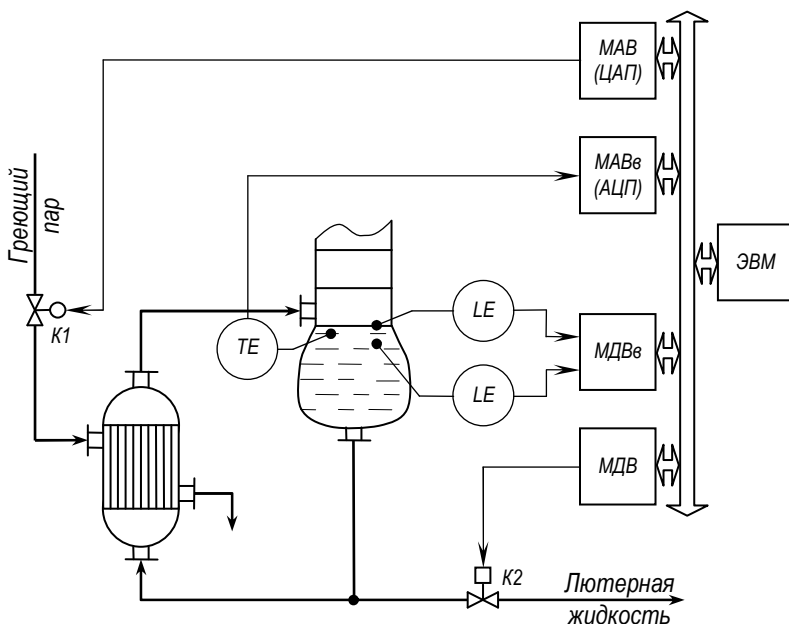


Рис. 15.1. Схема автоматизации нижней части ректификационной колонны с использованием ЭВМ

Для стабилизации уровня L лютерной жидкости используется контур, состоящий из ПИП LE верхнего и нижнего предельных уровней, модуля дискретного ввода МДВв, модуля дискретного вывода МДВ и исполнительного устройства на основе отсечного клапана K2. В отличие от регулирующего клапана K1, степень открытия которого может принимать любое значение в диапазоне от 0 до 1 (от 0% до 100%), отсечной клапан имеет только два устойчивых состояния: полностью открыт и полностью закрыт. Для управления таким клапаном используется дискретный сигнал¹ (логический ноль или логическая единица). Если уровень жидкости в кубе достигает верхнего ПИП LE, на его выходе формируется логическая единица, в результате чего ЭВМ, получая информацию через МДВв, посредством МДВ открывает клапан K2, например подачей на вход его исполнительного механизма логической единицы. Осуществляется слив лютерной жидкости, и её

¹ Термин «дискретный сигнал» строго не соответствует типу используемого сигнала, так как в теории сигналов это сигнал, определённый только в отдельные моменты времени (в отличие от аналогового сигнала, определённого в любой момент времени).

уровень L падает. При уменьшении уровня жидкости до нижнего ПИП LE на его выходе формируется логический ноль, ЭВМ закрывает клапан, и уровень L вновь начинает подниматься за счёт поступления жидкости с тарелок колонны.

В автоматизированных технологических процессах используется много дискретных сигналов не только для открытия/закрытия клапанов, но и для включения/выключения любых других устройств.

В настоящей работе необходимо обеспечить управление линиями порта P1 лабораторного контроллера при помощи его клавиатуры. Для контроля состояния линий к ним на плате внешних устройств подключены светодиоды. Необходимо обеспечить реализацию следующих функций:

- цифровые клавиши «1»...«8» включают/отключают устройство на соответствующей линии порта;
- клавиша «0» отключает все устройства;
- клавиша «9» включает все устройства;
- клавиша «*» инвертирует состояние всех линий порта;
- клавиша «#» включает/отключает чётные линии порта.

Включение/отключение какой-либо линии порта реализуется при помощи инверсии соответствующего бита порта P1. Для инверсии бита в языке Си удобно использовать операцию «ИСКЛЮЧАЮЩЕЕ ИЛИ» с соответствующей маской. Так, например, оператор $P1 \wedge= 0x04$; (в полной форме записи $P1 = P1 \wedge 0x04$;) инвертирует 2-ю линию (3-е устройство) порта P1.

Порядок выполнения работы

1. Разработайте алгоритм решения поставленной в работе задачи.
2. Запустите программу Keil μ Vision2 и модифицируйте программу созданного в предыдущей работе проекта для решения задачи.
3. Обработайте проект и произведите исправление допущенных ошибок.
4. Подключите контроллер к персональному компьютеру.
5. Подключите плату внешних устройств к контроллеру.
6. Подключите к плате внешних устройств клавиатуру.
7. Загрузите разработанную программу в контроллер и произведите её отладку.

Контрольные вопросы

1. Что называют аналоговой величиной? Приведите примеры.
2. Что называют дискретной величиной? Приведите примеры.

3. Как используются дискретные величины при управлении технологическими процессами?
4. Как в языке Си организовать инверсию отдельных битов?
5. С какими трудностями встречается программист при использовании клавиатуры?

Практическая работа 16

ИСПОЛЬЗОВАНИЕ АНАЛОГО-ЦИФРОВЫХ ПРЕОБРАЗОВАТЕЛЕЙ В ЗАДАЧАХ УПРАВЛЕНИЯ И КОНТРОЛЯ

Цель работы: получить навыки использования АЦП в системе контроля на основе ЭВМ.

Задание

Реализовать вывод на экран значения величины напряжения на выбранном канале АЦП и сигнализацию выхода этой величины за установленные пределы.

Методические указания

Аналоговые величины характеризуют состояние любого объекта. При использовании ЭВМ в автоматизированных системах управления любым объектом аналоговые величины необходимо преобразовать в цифровой код. Для этой цели в микропроцессорных системах используются встроенные или внешние АЦП. В однокристальном микроконтроллере PC80C552 фирмы Philips, на базе которого собран лабораторный контроллер, реализован 10-разрядный АЦП с 8-ю независимыми каналами [16, 17].

На рисунке 16.1 представлена схема АЦП микроконтроллера PC80C552.

При помощи аналогового мультиплексора MUX осуществляется выбор одного из восьми каналов аналого-цифрового преобразователя ADC (Analog to digital converter). Каналы АЦП подключены к порту P5 микроконтроллера. Для выбора канала используются три управляющих бита AADR0...AADR2 регистра ADCON. Напряжение U_x выбранного канала с выхода мультиплексора MUX поступает на инверсный вход компаратора А 10-разрядного АЦП прямого двоичного последовательного приближения.

Блок ADC включает в себя цифро-аналоговый преобразователь DAC (Digital to analog converter), регистр RG последовательного приближения и схему управления CS.

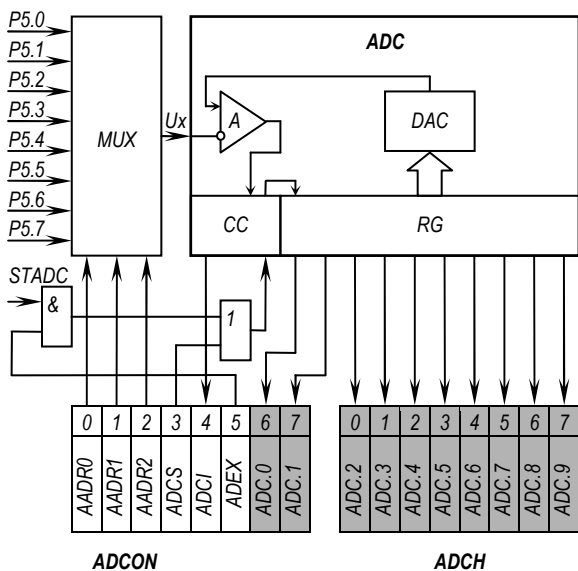


Рис. 16.1. Схема аналого-цифрового преобразователя микроконтроллера PCB80C552

АЦП последовательного приближения работает следующим образом. Логическая схема управления CC последовательным приближением в регистре RG устанавливает старший бит $ADC.9$ и очищает все остальные. На вход DAC поступает двоичный код $10\ 00000000$, под действием которого на его выходе формируется напряжение U_{DAC} , равное половине его полной шкалы. Компаратор A сравнивает напряжение U_{DAC} с входным напряжением U_x . При $U_{DAC} > U_x$ на выходе компаратора устанавливается логическая единица, иначе – логический ноль. Если входное напряжение U_x больше, чем U_{DAC} , то схема управления CC оставляет бит $ADC.9$ установленным, в противном случае сбрасывает его. Таким образом осуществляется проверка превышения входным напряжением половины шкалы АЦП. Преобразование занимает четыре машинных цикла [16, 17].

После определения состояния старшего бита те же действия выполняются для следующего бита $ADC.8$, который позволяет проверить, в какой четверти из выбранной битом $ADC.9$ половины шкалы находится напряжение входного сигнала U_x . Этот процесс повторяется до тех пор, пока не будут определены все 10 битов. После определения состояния младшего бита $ADC.0$ в регистре RG последовательного приближения будет храниться результат аналого-цифрового преобразования напряжения U_x – двоичный 10-разрядный код.

Запуск аналого-цифрового преобразования осуществляется установкой бита ADCS (ADC start) регистра ADCON. Программно можно только установить бит ADCS, сбрасывается он автоматически после завершения преобразования. Для внешнего управления запуском АЦП используется вход STADC. Если внешнее управление запуском разрешено (бит ADEX установлен), преобразование начинается при положительном перепаде (переходе из 0 к 1) на входе STADC (см. рис. 16.1).

Аналого-цифровое преобразование занимает 50 машинных циклов. После его завершения кроме сброса бита ADCS автоматически устанавливается бит ADCI (ADC Interrupt) – флаг прерывания от АЦП. По сигналу $ADCI = 1$ процессор может перейти к обработке этого прерывания по соответствующей подпрограмме, если прерывание разрешено битом EAD (см. табл. 13.3 в практической работе 13). Установка бита ADCI сигнализирует о том, что в регистрах ADCON и ADCH находится результат преобразования, который необходимо прочитать и сохранить. Поэтому, пока бит ADCI установлен, новое преобразование блокируется. Бит ADCS устанавливается только аппаратно, программно его можно сбросить. Это необходимо сделать для последующего запуска нового преобразования, так как оно возможно только при $ADCS = 0$ и $ADCI = 0$.

Старшие 8 разрядов результата аналого-цифрового преобразования хранятся в регистре ADCH, два оставшихся бита хранятся в ADCON.7 (ADC.1) и ADCON.6 (ADC.0). Пользователь может игнорировать два младших бита ADCON и использовать АЦП как 8-разрядный преобразователь (8 старших разрядов в ADCH). Для использования всех десяти разрядов в языке Си можно сформировать результат преобразования при помощи строки $Nx = ADCH * 4 + ADCON / 64$; где Nx – переменная типа int, в которую помещается 10-разрядный двоичный код. Деление и умножение двоичных чисел на 2^n , где n – целое, реализует сдвиг вправо или влево соответственно на n разрядов. Операция $ADCH * 4$ осуществляет сдвиг содержимого регистра ADCH (старших восьми разрядов) влево на 2 разряда, оставляя свободными два младших разряда для содержимого регистра ADCON (биты ADC.0 и ADC.1). Операция $ADCON / 64$ сдвигает старшие разряды регистра ADCON на шесть позиций вправо. При этом младшие разряды теряются, а старшие заполняются нулями. После выполнения операции сложения формируется 10-разрядный двоичный код в младших разрядах двух байтов переменной Nx типа int.

Код Nx пропорционален входному напряжению U_x . Коэффициент пропорциональности определяется напряжениями питания аналоговой части микроконтроллера:

$$Nx = \frac{U_x}{U_0} 1024 ,$$

где $U_0 = 2,5$ В – половина напряжения питания. В программе необходимо определить напряжение U_x выбранного канала x АЦП по коду N_x . Для переменной U_x целесообразно выбрать тип `float`.

Вывод значения напряжения выбранного канала на дисплей контроллера осуществляется при помощи функции `printf("U%u=%f В",chan,Ux)`, где `%u` – спецификатор преобразования для переменной `chan` номера канала, определённой типом `unsigned char`; `%f` – спецификатор преобразования для переменной U_x напряжения канала, определённой типом `float`. Значение переменной `chan` должно изменяться в диапазоне `0...7` в соответствии с нажатием клавиш `1...8` на клавиатуре контроллера, что обеспечивает оперативный выбор одного из каналов АЦП. В программе на языке Си для реализации этого выбора целесообразно использовать оператор `switch`. Строка `ADCON=chan`; устанавливает биты `AADR0...AADR2` в соответствии с номером выбранного канала. Старшие биты регистра `ADCON` сбрасываются. Для запуска аналого-цифрового преобразования необходимо установить бит `ADCS`. Это позволяет сделать строка `ADCON=0x08|ADCON`, использующая логическую операцию ИЛИ (символ `«|»`). Для сброса бита `ADCI` после сохранения результата преобразования можно воспользоваться строкой `ADCON=ADCON&0xEF;`

На плате внешних устройств лабораторного контроллера организована подача различных напряжений на все каналы АЦП. Все напряжения могут быть одновременно изменены при помощи регулятора, установленного на плате. Для сигнализации выхода напряжений каналов за установленные пределы следует использовать светодиоды, установленные на плате и соответствующие номеру канала. Сигнализацию необходимо настроить так, чтобы при вращении ручки регулятора от начального положения до конечного обеспечивалась сигнализация нижней и верхней предупредительных границ для всех каналов АЦП.

Прерывание от АЦП имеет номер 10. Определение функции обработки прерывания может иметь вид

```
void IntADC(void) interrupt 10
{
    тело функции
}
```

При конфигурировании системы прерываний в регистре масок прерываний необходимо кроме других используемых битов установить также и бит `EAD` регистра `IEN0` (см. табл. 13.3 в практической работе 13).

Порядок выполнения работы

1. Разработайте алгоритм решения поставленной в работе задачи. Определите, какие строки необходимо изменить и какие добавить в программу, созданную при выполнении практической работы 15. Используйте систему прерываний.

2. Запустите программу Keil μ Vision2 и модифицируйте программу созданного в предыдущей работе проекта для решения текущей задачи.

3. Обработайте проект и произведите исправление допущенных ошибок.

4. Подключите контроллер к персональному компьютеру.

5. Подключите плату внешних устройств к контроллеру.

6. Подключите к плате внешних устройств клавиатуру.

7. Загрузите разработанную программу в контроллер и произведите её отладку. При нажатии на цифровые клавиши должен осуществляться выбор канала АЦП. Отображение текущего значения напряжения должно происходить непрерывно. При вращении ручки регулятора при помощи светодиодов должна осуществляться сигнализация выхода напряжения за установленные пределы.

Контрольные вопросы

1. С какой целью в автоматизированных системах контроля и управления применяют аналого-цифровые преобразователи?

2. Для чего в схемах АЦП применяют мультиплексор?

3. Какие виды АЦП Вы знаете?

4. Как устроен и в чём состоит принцип работы АЦП последовательного приближения?

5. Какие биты и регистры используются для управления АЦП микроконтроллера РСВ80С552?

6. Опишите алгоритм работы с АЦП микроконтроллера РСВ80С552.

7. Какие трудности возникают при работе с 10-разрядным кодом и как их преодолеть?

8. От чего зависит код, полученный в результате аналого-цифрового преобразования?

9. Каким образом можно установить или сбросить отдельный бит того или иного регистра?

ЗАКЛЮЧЕНИЕ

В ходе выполнения практических работ настоящего практикума студенты бакалавриата приобретают навыки работы с сетевыми приложениями операционной системы Windows. Представленные задания позволяют усвоить знания об устройстве и принципе работы современной вычислительной техники, а также способах и средствах обмена информацией между отдельными устройствами сетей вычислительных машин. Практическая работа с промышленными контроллерами развивает способности программирования конкретных устройств, совершенствует логическое мышление и позволяет на интуитивном уровне заложить основы для управленческой деятельности.

Усвоение базовых понятий, рассмотренных в практикуме, даёт возможность дальнейшего освоения дисциплины. Для формирования законченного представления о сетях вычислительных машин целесообразно более детально ознакомиться с современными протоколами передачи данных, используемыми в офисной и промышленной технике.

СПИСОК ЛИТЕРАТУРЫ

1. Бройдо, В.Л. Архитектура ЭВМ и систем : учебник для вузов / В.Л. Бройдо, О.П. Ильина. – СПб. : Питер, 2006.
2. Бройдо, В.Л. Вычислительные системы, сети и телекоммуникации : учебник для вузов / В.Л. Бройдо. – 2-е изд. – СПб. : Питер, 2005.
3. Пятибратов, А.П. Вычислительные системы, сети и телекоммуникации : учебник для вузов / А.П. Пятибратов, Л.П. Гудынов, А.А. Кириченко ; под ред. проф. А.П. Пятибратова. – 3-е изд. – М. : Финансы и статистика, 2005.
4. Архитектура компьютерных систем и сетей : учебное пособие для вузов / М.И. Семенов, И.Т. Трубилин, В.И. Лойко, Т.П. Барановская. – М. : Финансы и статистика, 2004.
5. Велихов, А.В. Компьютерные сети : учебное пособие для вузов / А.В. Велихов. – М. : Новый издательский дом, 2005.
6. Воеводин, В.В. Параллельные вычисления / В.В. Воеводин, Вл.В. Воеводин. – СПб. : БВХ – Петербург, 2002.
7. Жмакин, А.П. Архитектура ЭВМ / А.П. Жмакин. – СПб. : БВХ – Петербург, 2006.
8. Комарцева, А.Г. Нейрокомпьютеры : учебное пособие для вузов / А.Г. Комарцева, А.В. Максимов. – М. : МГТУ им. Баумана, 2002.
9. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум. – СПб. : Питер, 2002.
10. Цилькер, Б.Я. Организация ЭВМ и систем : учебник для вузов / Б.Я. Цилькер, С.А. Орлов. – СПб. : Питер, 2006.

11. Иванова, Т.Н. Корпоративные сети связи / Т.Н. Иванова. – М. : ЭкоТрендз, 2001.
12. Оливер, В.Г. Компьютерные сети: принципы, технологии, протоколы / В.Г. Оливер, Н.А. Оливер. – Питер, 2001.
13. Дебра, Л.Ш. Основы компьютерных сетей. Пер. с англ. / Л.Ш. Дебра. – М. : Издательский дом «Вильямс», 2003.
14. Марапулец, Ю.В. Программирование на языке высокого уровня [Электронный ресурс] : учебное пособие / Ю.В. Марапулец // Режим доступа: <http://window.edu.ru>. – Петропавловск-Камчатский: КамчатГТУ, 2008. – 189 с. – Загл. с экрана.
15. Дейл, Н. Программирование на С++ [Электронный ресурс] : учебник / Н. Дейл, Ч. Уимз, М. Хедингтон // Режим доступа: <http://e.lanbook.com>. – М. : ДМК Пресс, 2007. – 672 с. – Загл. с экрана.
16. Микроконтроллеры семейства MCS-51 [Электронный ресурс] : учебное пособие / В.Н. Веприк, В.А. Афанасьев, А.И. Дружинин и др. // Режим доступа: <http://window.edu.ru>. – Новосибирск : Изд-во Новосиб. гос. техн. ун-та, 1997. – 62 с. – Загл. с экрана.
17. Евланов, Ю.Н. Однокристалльный микроконтроллер 80C552 [Электронный ресурс] : методическое пособие / Ю.Н. Евланов, В.А. Новиков, А.А. Шатохин // Режим доступа: <http://window.edu.ru>. – М. : Изд-во МЭИ, 2001. – 60 с. – Загл. с экрана.
18. Бояринов, А.Е. Архитектура микроконтроллеров семейства MCS-51 [Электронный ресурс] : конспект лекций / А.Е. Бояринов, И.А. Дьяков // Режим доступа: <http://window.edu.ru>. – Тамбов : Изд-во Тамб. гос. техн. ун-та, 2005. – 64 с. – Загл. с экрана.
19. Воробьева, Г.С. Методические рекомендации к выполнению курсового проекта по дисциплине «Интерфейсы микропроцессорных систем» [Электронный ресурс] : методические указания / Г.С. Воробьева, В.В. Яковлев // Режим доступа: <http://window.edu.ru>. – Загл. с экрана.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
Практическая работа 1. ТЕСТИРОВАНИЕ АППАРАТНЫХ СРЕДСТВ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ	4
Практическая работа 2. СЕТЕВЫЕ ТОПОЛОГИИ	11
Практическая работа 3. ЛИНИИ СВЯЗИ	12
Практическая работа 4. АДРЕСАЦИЯ В КОМПЬЮТЕРНОЙ СЕТИ	13
<i>Практическая работа 5. СРЕДСТВА УСТРАНЕНИЯ НЕИСПРАВНОСТЕЙ В ТСР/IP</i>	17
<i>Практическая работа 6. УСТРОЙСТВА СВЯЗИ. ПРОЕКТИРОВАНИЕ СЕТИ КРУПНОЙ ФИРМЫ</i>	20
<i>Практическая работа 7. ИСПОЛЬЗОВАНИЕ УДАЛЁННЫХ СЕТЕВЫХ РЕСУРСОВ</i>	21
<i>Практическая работа 8. МЕТОДЫ ЗАЩИТЫ КОМПЬЮТЕРА ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА ИЗ ВНЕШНЕЙ СЕТИ И ПОИСК УЯЗВИМОСТЕЙ В СИСТЕМЕ ЗАЩИТЫ</i>	24
<i>Практическая работа 9. ИЗУЧЕНИЕ ЗАПОМИНАЮЩИХ УСТРОЙСТВ ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН</i>	29
<i>Практическая работа 10. ИЗУЧЕНИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ МИКРОКОНТРОЛЛЕРОВ</i>	40
<i>Практическая работа 11. ПЕРЕДАЧА ИНФОРМАЦИИ ПО ИНТЕРФЕЙСУ RS-232</i>	56
<i>Практическая работа 12. ПРОГРАММИРОВАНИЕ КОНТРОЛ- ЛЕРОВ НА ЯЗЫКЕ СИ</i>	66
<i>Практическая работа 13. ИЗУЧЕНИЕ СИСТЕМЫ ПРЕРЫВА- НИЙ МИКРОПРОЦЕССОРОВ</i>	68
<i>Практическая работа 14. ИЗУЧЕНИЕ УСТРОЙСТВА КЛАВИА- ТУРЫ ЭВМ</i>	77
<i>Практическая работа 15. ИСПОЛЬЗОВАНИЕ ДИСКРЕТНЫХ СИГНАЛОВ В ЗАДАЧАХ УПРАВЛЕНИЯ</i>	82
<i>Практическая работа 16. ИСПОЛЬЗОВАНИЕ АНАЛОГО- ЦИФРОВЫХ ПРЕОБРАЗОВАТЕЛЕЙ В ЗАДАЧАХ УПРАВЛЕНИЯ И КОНТРОЛЯ</i>	85
ЗАКЛЮЧЕНИЕ	90
СПИСОК ЛИТЕРАТУРЫ	90

Учебное издание

БАЛАБАНОВ Павел Владимирович,
БОЯРИНОВ Алексей Евгеньевич,
САВЕНКОВ Александр Петрович

**ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И
СЕТИ В ЗАДАЧАХ
УПРАВЛЕНИЯ КАЧЕСТВОМ**

Практикум

Редактор Е.С. Кузнецова
Компьютерное макетирование И.В. Евсевой

Подписано в печать 31.08.2012.
Формат 60×84/16. 5,35 усл. печ. л. Тираж 70 экз. Заказ № 463

Издательско-полиграфический центр ФГБОУ ВПО «ТГТУ»
392000, г. Тамбов, ул. Советская, д. 106, к. 14