Министерство образования Республики Беларусь

Учреждение образования
«Полоцкий государственный университет»

ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к лабораторным работам
для студентов специальностей
1-40 02 01 «Вычислительные машины, системы и сети»,
1-40 01 01 «Программное обеспечение
информационных технологий»

Новополоцк 2007

УДК 681.5(075.8)
ББК 32.81я73

Одобрены и рекомендованы к изданию
методической комиссией радиотехнического факультета

Кафедра технической кибернетики

Составители:
С. А. ТАРАСОВ, канд. техн. наук, доцент;
Н. В. БРОВКО, преподаватель-стажер

Рецензенты:
Р. П. БОГУШ, канд. техн. наук, доцент, зав. кафедрой;
В. А. РЫМАРЕВ, ассистент кафедры конструирования и технологии радиоэлектронных средств

Лабораторная работа № 1

ИССЛЕДОВАНИЕ РАБОТЫ УСТРОЙСТВА ЧТЕНИЯ И ЗАПИСИ НА ГИБКОМ МАГНИТНОМ ДИСКЕ

Цель работы – изучить физическую и логическую организацию накопителя на гибких магнитных дисках и прерывание int 13h BIOS для работы с контроллером накопителя на гибком магнитном диске.

Теоретические сведения

Физическая организация накопителя на гибком магнитном диске. Дискета представляет собой круглую пластину, покрытую с двух сторон магнитным материалом. Информация записывается раздельно на две стороны дискеты. В дисководе к поверхности дискеты прижимаются две (по одной с каждой стороны) головки чтения/ записи. Пара головок может перемещаться вдоль радиуса дискеты, занимая на нем ряд фиксированных положений. При вращении дискеты точки ее поверхности, которые могут находиться в контакте с головками, образуют концентрические окружности, именуемые дорожками. Дорожки делятся на сектора. Цилиндр – совокупность всех дорожек, расположенных на разных поверхностях и равноудаленных от оси вращения. С точки зрения адресации понятия «дорожка» и «цилиндр» являются синонимами. Обмен информацией с дискетой происходит через прямой доступ к памяти. Сектор – это наименьший объем (блок) информации, передаваемый за одну операцию чтения/ записи. Объем сектора – 512 байт. Кластер – минимальный объем дискового пространства, который может быть выделен для размещения файла. Все файловые системы, используемые для работы с жесткими дисками, основаны на кластерах, которые состоят из одного или нескольких смежных секторов. Чем меньше размер кластера, тем более эффективно используется дисковая память. Если размер кластера не задан во время форматирования, он выбирается операционной системой в зависимости от объема диска. Стандартные значения подобраны таким образом, чтобы снизить потерю дискового пространства и степень возможной фрагментации тома. Кластеры также называют единичными блоками. Адрес сектора состоит из трех составляющих:

- номер дорожки (нумерация дорожек начинается с 0);
- номер головки (нумерация головок начинается с 0);
- номер сектора на дорожке (нумерация секторов начинается с 1).

При записи на диск больших объемов информации необходимо свести к минимуму затраты по переключению на следующий сектор диска. Поэтому принят такой порядок определения следующего сектора при последовательной записи (чтении). Следующим считается сектор, расположенный следующим на той же дорожке, под той же головкой (при записи сектора головки за счет вращения диска установятся над следующим сектором, так что затраты времени на переключения практически нулевые). При заполнении всей дорожки следующим принято считать первый сектор дорожки, расположенный на том же цилиндре под следующей головкой (этот сектор будет находиться под головками, а электрическое переключение на другую головку – процесс быстрый). И только при заполнении всего цилиндра меняется номер дорожки в адресе, этот процесс требует механического перемещения головок и, следовательно, больших затрат времени.

Логическая организация накопителя на гибких магнитных дисках

Логическая структура дискеты включает в себя следующие элементы:

- Boot – сектор;
- FAT (2 копии);
- корневой каталог;
- область данных.

Boot-сектор является самым первым сектором логического диска (номер логического сектора – 0). Для дискеты его физический адрес – дорожка 0, головка 0, сектор 1. При загрузке системы с дискеты именно этот сектор считывается в память программой POST и выполняет дальнейшую загрузку. Boot-сектор содержит информацию о физических параметрах диска и о размещении на нем системной информации, используемой далее системой при работе с диском, поэтому даже если дискета и несистемная, т. е. загрузка с нее не выполняется, на ней все равно имеется Boot-сектор с данными о носителе. Формат Boot-сектора различен для версий DOS до 4.0 и от 4.0 и выше. Далее приводится описание Boot-сектора для DOS 4.0 и выше.

Первые 3 байта сектора содержат команду JMP, обеспечивающую переход на программу загрузки.

Следующие 8 байт – символьный идентификатор программного средства, производившего форматирование дискеты. Если дискета форматировалась системной утилитой FORMAT, там записан номер версии операционной системы.

Поле SectSize (2 байта) – размер сектора, всегда содержит значение 512.

Поле ClustSize (1 байт) показывает, сколько секторов содержится в одном кластере – единице распределения дисковой памяти.

Поле ResSect (2 байта) содержит число резервных секторов, расположенных до начала FAT, обычно это 1 – перед FAT имеется только Boot-сектор.

Поле FatCnt (1 байт) – число копий FAT, всегда 2.

Поле RootSize (2 байта) показывает размер корневого каталога, чтобы узнать число секторов в корневом каталоге надо содержимое этого поля разделить на 16.

Поле TotSecs (4 байта) – общее число секторов на диске для диска, отформатированного в DOS 4.0 или выше, содержит 0, если число секторов больше 65535, в этом случае число секторов может быть выбрано из поля LongTotSecs.

Байт Media идентифицирует тип носителя. Возможны такие его значения: 0xFF – 2 стороны, 8 секторов на дорожке; 0xFE – 1 сторона, 8 секторов на дорожке; 0xFD – 2 стороны, 9 секторов на дорожке; 0xFC – 1 сторона, 9 секторов на дорожке; 0xF9 – 2 стороны, 15 секторов на дорожке; 0xF8 – жесткий диск.

Поле FatSize (2 байта) – число секторов в одной копии FAT.

Поле TrkSecs (2 байта) – число секторов на дорожке.

Поле HeadCnt (2 байта) – число поверхностей.

Поля HidnSecL и *HidnSecH* для DOS 4.0 и выше можно интерпретировать как одно поле: dword HidnSec (4 байта) – это количество скрытых секторов. Для DOS ниже 4.0 используется 2-байтное число скрытых секторов – HidnSecL, и на этом область данных Boot-сектора кончается.

Поле Drive – номер дисковода, на котором диск форматировался.

Поле DOS4_flag содержит код 0x29, если диск форматировался в DOS 4.0 и выше.

Серийный номер тома – случайное число, записываемое в поле VolNum при форматировании, может использоваться в дальнейшем для идентификации диска, как и метка тома (поле VolLabel).

Поле FatForm содержит символьную последовательность 'FAT12' или 'FAT16', в зависимости от формата FAT. После этой системной информации в Boot-секторе записана программа начальной загрузки.

Для версий DOS более ранних, чем 4.0 блок системной информации, в Boot-секторе имеет меньший размер – до поля HidnSecL включительно.

Поля от SectSize по FatSize включительно образуют так называемый блок параметров BIOS (BPB – BIOS Parameter Block), он формируется в оперативной памяти драйвером диска и в дальнейшем используется при всех операциях с данным диском. В DOS 4.0 и выше принят расширенный BPB – от поля SectSize до LongTotSecs включительно.

Формат Boot-сектора зависит не от версии DOS, установленной на той ПЭВМ, на которой дискета читается, а от версии на той ПЭВМ, где дискета форматировалась. Поскольку формат ранних версий полностью перекрывается форматом поздних, совместимость по носителям сохраняется.

FAT (File Allocation Table – таблица размещения файлов)

Следом за Boot-сектором на диске расположена FAT. С точки зрения распределения дискового пространства диск разбит на кластеры. Кластер представляет собой группу последовательно расположенных секторов. Если DOS получает запрос на выделение дискового пространства, она выделяет сразу целый кластер. Размер кластера является компромиссом между двумя противоречивыми требованиями: с точки зрения экономии дискового пространства выгоден малый размер кластера, т.к. для маленького файла (размером даже в один байт) выделяется целый кластер, большая часть которого не используется. Но с другой стороны, при малом размере кластера их на диске получится очень много, и управление ими усложняется. FAT представляет собой «карту» дискового пространства – массив элементов, каждый из которых соответствует одному кластеру диска. Номер элемента соответствует номеру кластера.

Поскольку область диска, содержащая системную информацию, распределена предварительно, она в FAT не отражается. Первые два элемента FAT не используются, (первый байт содержит код, совпадающий с полем Media Boot-сектора), нумерация кластеров области данных диска начинается, таким образом, с 2. Размер элемента FAT может быть 12 или 16 бит. Описание содержимого полей далее дается для 12-битного формата FAT, в скобках указываются значения для 16-битного формата. Значения элементов FAT от 2 до 0xFEF (0xFFEF) включительно – информационные значения. Такое значение – номер следующего кластера, распределенного данному файлу. Таким образом, FAT обеспечивает списковую структуру распределения – каждый ее элемент содержит указание на следующий элемент. Поскольку кластеры распределяются файлу по мере его заполнения, файл не обязательно занимает смежные кластеры, FAT может обеспечить связывание в цепочку разнесенных по диску кластеров. Другие значения

элементов зарезервированы для системной информации. Значение 0 идентифицирует свободный кластер. Значения элементов от 0xFF0 до 0xFF6 (от 0xFFF0 до 0xFFF6) – резервные кластеры. Значение 0xFF7 (0xFFF7) – сбойный кластер. Значения от 0xFF8 до 0xFFF (от 0xFFF8 до 0xFFFF) – признак конца цепочки, последнего кластера файла.

Корневой каталог

Как уже упоминалось, на диске хранятся две идентичные копии FAT, расположенные одна за другой. Следом за FAT размещается корневой каталог. Корневой каталог – главный каталог диска, с которого начинается дерево подкаталогов диска. В FAT12 и FAT16 для него выделено 16 кбайт для хранения 512 элементов. В системе FAT32 корневой каталог является файлом произвольного размера. Корневой каталог состоит из элементов каталога, которые состоят из следующих полей.

Поле FNAME (11 байт = 8 байт для короткого имени + 3 байта для расширения) содержит имя файла или подкаталога, причем для файла байты 0-7 поля содержат имя, а 8 – 10 – расширение, неиспользуемые байты поля заполнены пробелами. Если файл удаляется из каталога, то в соответствующем элементе первый символ имени заменяется кодом 0xE5, это оставляет возможность восстановления случайно удаленного файла, если, однако, за это время элемент каталога не будет использован для нового файла. Код 0 в первом символе имени означает, что элемент свободен и никогда не использовался. При внесении в каталог нового файла система сначала использует элементы удаленных файлов, и лишь затем – свободные.

Поле ATTR (1 байт) – поле атрибутов содержит признаки, характеризующие файл. Формат байта атрибутов файла показан на рис. 1.1.

Рис. 1.1. **Формат байта атрибутов файла**

Разряды байта атрибутов устанавливаются в 1 в случае, если у файла имеется соответствующее свойство:

- бит 0 – файл только для чтения;
- бит 1 – скрытый файл;
- бит 2 – системный файл;
- бит 3 – идентификатор тома;
- бит 4 – каталог;
- бит 5 – архивированный файл;
- биты 6 и 7 – зарезервированы, должны быть установлены в 0.

Поля Time (2 байта) и *Date* (2 байта) содержат время и дату последней модификации файла. Формат времени: ЧЧЧЧЧМММММММССССС (Часы, Минуты, Секунды); формат даты: ГГГГГГГММММДДДДД (Год, Месяц, День).

Поле cl (2 байта) – номер первого кластера, распределенного файлу, т. е. начало той цепочки, которая продолжается в FAT.

Поле size (4 байта) – размер файла в байтах. В DOS не предусмотрены какие-либо специальные признаки конца файлов. Применяемый в некоторых случаях символ ^Z (код 26) интерпретируется как признак конца конкретными программами, но не DOS. DOS же определяет конец файла по его размеру, получаемому из этого поля элемента каталога. Для подкаталога это поле содержит 0.

Почти все, сказанное выше о корневом каталоге, справедливо и для подкаталогов. Разница между ними и корневым каталогом состоит в их размещении. Корневой каталог размещается на определенном месте (вслед за FAT) и имеет фиксированный размер. Подкаталогам распределяется дисковая память как файлам, т. е. при нехватке места в подкаталоге ему выделяется новый кластер, и это выделение отражается в FAT, таким образом, размер подкаталога не ограничивается. В любом подкаталоге обязательно присутствуют два элемента с именами «.» и «..» и атрибутами «подкаталог». Первый из них – ссылка на самого себя, второй – ссылка на предшествующий узел дерева подкаталогов.

Прерывание int 13h BIOS для работы с контроллером накопителя на гибком магнитном диске

Регистр – внутренняя ячейка памяти микропроцессора, используемая для выполнения текущих операций. Т. к. доступ к регистру происходит во много раз быстрее, чем к ячейкам оперативной памяти, размещение промежуточных результатов в регистрах дает значительный выигрыш в скорости исполнения программы. Микропроцессор имеет большое число двух-

байтовых внутренних регистров, каждому из которых присвоено «имя», или мнемоника. Все регистры можно условно разделить на три группы:

- регистры общего назначения- AX,BX,CX,DX;
- регистры-указатели – DI,SI,BP,SP,IP;
- сегментные регистры – CS,DS,SS,ES.

Регистры общего назначения. Регистры общего назначения можно рассматривать как четыре 16-битовых или восемь 8-битовых регистров. В первом случае регистры имеют имена AX, BX, CX, DX. Эти регистры образованы из 8-битовых регистров AL, AH, BL, BH, CL, CH, DL и DH. здесь L и H означают младшие и старшие байты 16-битовых регистров

Например, регистры AL и AH образуют соответственно младший и старший байты регистра AX.

Сегментные регистры CS, DS, SS, ES. Программы и данные хранятся в отдельных областях памяти. Эти области (или сегменты) могут иметь объем до 64 кбайт. Микропроцессор может иметь дело одновременно с четырьмя сегментами. Начальные адреса этих сегментов хранятся в его четырех сегментных регистрах.

Регистр дополнительного сегмента ES (extra segment) указывает на текущий дополнительный сегмент, который используется при выполнении операций над строками.

Язык Си поддерживает исключительно удобный механизм доступа к внутренним регистрам микропроцессора через псевдопеременные, ссылка на которые в Си-программе компилируется в ссылку на внутренний регистр микропроцессора:

_AL, _AH, _BL, _BH, _CL, _CH, _DL, _DH;

_AX, _BX, _CX, _DX;

_DI, _SI, _BP, _SP, _IP;

_CS,_DS,_SS,_ES.

Для того чтобы записать в регистр необходимое значение, достаточно присвоить псевдопеременной это значение, например: _AX=0x07.

Прерывания

В самом общем виде – это наличие в аппаратуре специальных средств, с помощью которых выполнение текущей программы приостанавливается и процессор переходит к так называемой программе обслуживания прерывания. Механизм прерываний позволяет организовать выполнение тех или иных функций ядра и быструю реакцию процессора на возникновение каких-то внешних событий.

Часто обработчикам программных прерываний требуется передать какие-то значения, задающие конкретное действие, характеристики ситуа-

ции и т.п., и получить какие-то результаты по завершению исполнения прерывания. Для такого обмена данными используются внутренние регистры процессора. Такого вида прерывания используются в данной лабораторной работе. Перед вызовом прерывания необходимо передать определенные параметры, значения которых заносятся в регистры.

Описание функций 13-ого прерывания BIOS для работы с контроллером на гибком магнитном диске

Функция 00H – привести в исходное состояние дисковую систему. Приводит в исходное состояние контроллер диска и подключенные к нему дисководы, позиционер головок чтения-записи перемещается к цилиндру 0 и подготавливает систему к операциям ввода-вывода на диске.

При вызове AH = 00H, DL = дисковод (00H, 01H,…, 80H, 81H,…).

При возврате если функция выполнена успешно, флаг переноса сброшен (CF = 0).

Если функция не выполнена, флаг переноса установлен (CF=1) и AH = состояние (см. Int 13H с функцией 01H).

Эта функция вызывает сброс и рекалибровку дискового контроллера, позиционер головки устанавливается на нулевой цилиндр. Если в адресе дисковода старший бит (бит 7) установлен в 1, выполняется сброс контроллера НМД. Сброс рекомендуется выполнять после того, как произошла ошибка при выполнении других операций, таких как чтение или запись. После сброса можно попытаться повторить операцию. Адрес дисковода 0 соответствует первому флоппи-диску A, 1 – второму B и т. д. Адреса 80, 81 соответствуют первому и второму физическим накопителям на жестком магнитном диске. Функцию следует вызывать после сбоя при выполнении запросов на чтение, запись, верификацию или форматирование на гибком диске перед повторением операции. Если при вызове функции DL = 80H (т. е. выбирается жесткий диск), приводятся в исходное состояние контроллеры гибкого и жесткого дисков.

Функция 01H – получить состояние дисковой системы. Возвращает состояние последней дисковой операции.

При вызове AH = 01H, DL = дисковод (00H, 01H,…, 80H, 81H,…).

При возврате AH = 00H, AL = состояние дисковода после завершения последней операции.

Эта функция может быть использована для анализа результата выполнения дисковой операции и получения кода ошибки. Передаваемый в регистре AL код ошибки (табл. 1.1, Г – только для гибкого диска; Ж – только для жесткого диска) функция берет из области данных BIOS – из байта с адресом 0000:0441h.

При использовании жесткого диска код ошибки 11H (ошибка скорректированных данных) указывает на обнаружение восстановимой ошибки в процессе предшествующего чтения сектора (Int 13H с функцией 02H).

Таблица 1.1

Коды ошибок дисковой операции

Код ошибки	Описание
00H	отсутствие ошибки
01H	недопустимая команда
02H	не найдена адресная метка
03H	диск защищен от записи (Г)
04H	не найден сектор
05H	сброс в исходное состояние не выполнен (Ж)
06H	гибкий диск снят (Г)
07H	дефектная таблица параметров (Ж)
08H	нарушение границ ПДП (Г)
09H	ПДП пересек границу 64 Кбайт
0AH	флаг дефектного сектора (Ж)
0BH	флаг дефектной дорожки (Ж)
0CH	не найден тип носителя (Г)
0DH	недопустимое число секторов при форматировании
0EH	обнаружена метка адреса управляющих данных - уровень арбитража ПДП вышел из диапазона (Ж)
0FH	ошибка ПДП
10H	невосстановимая ошибка циклического контроля (ECC)
11H	ошибка данных скорректирована кодом проверки корректировки (Ж)
20H	отказ контроллера
40H	сбой поиска дорожки
80H	тайм-аут диска (отсутствие ответа)
AAH	дисковод не готов (Ж)
BBH	неопределенная ошибка (Ж)
CCH	отказ записи (Ж)
E0H	ошибка регистра состояния (Ж)
FFH	операция опознания не выполнилась (Ж)

Функция 02H – прочитать сектор. Читает в память один или более секторов. Эта функция позволяет прочитать один или несколько секторов диска в буфер, находящийся в оперативной памяти. Необходимо задать для начального сектора номер дорожки, головки и самого сектора.

Для НГМД номер дорожки и сектора задается следующим образом: биты регистра CX 7...0 задают номер сектора, а биты 15...8 – номер дорожки.

Перед чтением необходимо подготовить таблицу параметров дискеты или диска (для операций с НГМД).

При вызове AH = 02H, AL = число секторов, CH = цилиндр (дорожка), CL = сектор, DH = головка, DL = дисковод (00H, 01H,…, 80H, 81H,…), ES: BX = сегмент: относительный адрес буфера.

При возврате, если функция выполнена успешно, флаг переноса сброшен (CF=0) и AH= 00H, AL = число переданных секторов. Если функция не выполнена, флаг переноса установлен (CF=1) и AH = состояние (см. Int 13H с функцией 01 H).

При использовании жестких дисков код ошибки 11H свидетельствует о том, что ошибка чтения была скорректирована алгоритмом ECC (проверки и корректировки ошибки); в этом случае регистр AL содержит длину посылки. Возвращенные данные, скорее всего, правильны, хотя имеется незначительная вероятность неправильной корректировки. Если была запрошена многосекторная пересылка, операция завершилась передачей сектора, содержащего ошибку чтения. При использовании гибких дисков ошибка может возникнуть, если к моменту запроса выключен мотор. ПЗУ BIOS не организует автоматического ожидания раскрутки мотора до требуемой скорости перед попыткой выполнения операции чтения. Запрашивающая программа должна установить в исходное состояние дисковую систему (Int 13H с функцией 00H) и трижды повторить операцию перед тем, как прийти к заключению, что ошибка вызвана чем-то другим.

Функция 03H – записать сектор. Записывает из памяти на диск один или более секторов. Функция записи секторов аналогична предыдущей функции, за исключением направления перемещения данных. В этом случае данные записываются из буфера в сектора диска.

При вызове AH = 03H, AL = число секторов, CH = цилиндр, CL = сектор, DH = головка, DL = дисковод (00H, 01H,…, 80H, 81H,…).

При возврате, если функция выполнена успешно, флаг переноса сброшен (CF=0) и AH = 00H, AL = число переданных секторов.

Если функция не выполнена, флаг переноса установлен (CF=1) и AH = состояние (см. Int 13H с функцией 01H).

Замечания. При использовании гибких дисков ошибка может возникнуть, если к моменту запроса выключен мотор. ПЗУ BIOS не организует автоматического ожидания раскрутки мотора до требуемой скорости перед попыткой выполнения операции записи. Запрашивающая программа должна установить в исходное состояние дисковую систему (Int 13H с функцией 00H) и трижды повторить операцию перед тем, как прийти к заключению, что ошибка вызвана чем-то другим.

Функция 04H – верифицировать сектор. Верифицирует адресные поля одного или более секторов. Эта операция не пересылает данные ни в память, ни из памяти.

С помощью этой функции можно убедиться, что указанные сектора существуют и их можно прочесть. Данные проверяются по методу избыточного циклического контроля (CRC). Адрес буфера не нужен, т. к. чтения данных в оперативную память при проверке секторов не происходит.

При вызове AH = 04H, AL = число секторов, CH = цилиндр (дорожка), CL = сектор, DH = головка , DL = дисковод (00H, 01H,…, 80H, 81H,…).

При возврате, если функция выполнена успешно, флаг переноса сброшен (CF=0) и AH = 00H, AL = число верифицированных секторов.

Если функция не выполнена, флаг переноса установлен (CF=1) и AH = состояние (см. Int 13H с функцией 01H). Эту функцию можно использовать для проверки, установлен ли диск в дисководе гибких дисков. Если используется компьютер с BIOS, выпущенной ранее 11.15.85, регистры ES:BX должны указывать на буфер соответствующего размера, как и при выполнении операции чтения.

Функция 05H – форматировать дорожку. Инициализирует поля адресов секторов и дорожки на указанной дорожке диска. Функция форматирования предназначена для начального формирования структуры дорожки диска, она разрушает все имеющиеся на дорожке данные. С помощью функции 05h можно за один раз отформатировать только одну дорожку с указанным номером. Для этой функции необходимо задать буфер формата.

Перед вызовом функции форматирования регистры ES:BX должны содержать полный адрес буфера формата. Для дискет перед форматированием этот буфер должен представлять собой заполненный массив 4-байтовых элементов – номера дорожки, головки, сектора и кода размера сектора. Код размера сектора может иметь следующие значения:

- 128 байт на сектор;
- 256 байт на сектор;
- 512 байт на сектор;
- 1024 байт на сектор.

Количество элементов в массиве должно быть равно количеству создаваемых на дорожке секторов, т. е. для каждого сектора буфер формата должен содержать один описывающий его четырехбайтовый элемент.

При форматировании флоппи-дисков с помощью этой функции таблица параметров дискеты должна содержать правильное значение количества секторов на дорожке и другие параметры.

При вызове AH = 05H, AL = чередование (жесткие диски PC/XT), CH = цилиндр, DH = головка, DL= дисковод (00H, 01H,…, 80H, 81H,…), ES:BX= сегмент: относительный – адрес списка адресных полей.

При возврате, если функция выполнена успешно, флаг переноса сброшен (CF = 0) и AH = 00H.

Если функция не выполнена, флаг переноса установлен (CF = 1) и AH = состояние (см. Int 13H с функцией 01H).

На гибких дисках список адресных полей состоит из последовательности 4-байтных записей, по одной записи на сектор, имеющих следующий формат (табл. 1.2).

Таблица 1.2

Формат адресного поля для гибкого диска

Байт	Содержимое
0	Цилиндр
1	Головка
2	Сектор
3	Код размера сектора: ▪ 00H, если 128 байт на сектор; ▪ 01H, если 256 байт на сектор; ▪ 02H, если 512 байт на сектор (стандарт); ▪ 03H, если 1024 байт на сектор

На гибких дисках число секторов на дорожку берется из таблицы параметров гибкого диска в BIOS, адрес которой записан в векторе прерывания Int 1EH. Если эта функция используется для формирования гибких дисков на машинах PC/AT или PS/2, ей должен предшествовать вызов Int 13H с функцией 17H для выбора носителя, предназначенного для форматирования. При использовании жестких дисков два старших бита 10-бит номера цилиндра помещаются в два старших бита регистра CL. При использовании жестких дисков машин PC/XT-286, PC/AT и P5/2 регистр ES:BX указывает на 512-байтный буфер, содержащий пары байтов для каждого сектора физического диска. Нулевой байт содержит 00H – для работоспособного сектора, 80H – для дефектного сектора. Первый байт содержит номер сектора.

Например, для форматирования дорожки с 17 секторами и чередованием, равным 2, ES:BX должен указывать на следующий 34-байт массив в начале 512-байт буфера:

```
db 00h,01h,00h,0ah,00h,02h,00h,0bh,00h,03h,00h,0ch
db 00h,04h,00h,0dh,00h,05h,00h,0eh,00h,06h,00h,0fh
db 00h,07h,00h,10h,00h,08h,00h,11h,00h,09h
```

Функция 08H – получить параметры дисковода. С помощью этой функции программа может определить тип дисковода, количество дисководов, обслуживаемых первым дисковым контроллером, и другие пара-

метры дисковода, которые нужны программе для организаций доступа к диску на физическом уровне. Возвращает различные параметры указанного дисковода.

При вызове AH = 08H, AL = дисковод (00H, 01H,…, 80H, 81H,…).

При возврате если функция выполнена успешно, флаг переноса сброшен (CF=0) и BL = тип дисковода (гибкие диски PC/AT и PS/2):

- 01H, если 360 Кбайт, 40 дорожек, 5,25";
- 02H, если 1,2 Мбайт, 80 дорожек, 5,25";
- 03H, если 720 Кбайт, 80 дорожек, 3,5";
- 04H, если 1,44 Мбайт, 80 дорожек, 3,5".

CH = младшие 8 бит максимального номера цилиндра, CL = биты 6 – 7: два старших бита максимального номера цилиндра, биты 0 – 5: максимальный номер сектора, DH = максимальный номер головки, DL = число дисководов, ES:DI = сегмент: относительный адрес таблицы параметров дисковода.

Если функция не выполнена, флаг переноса установлен (CF=1) и AH = состояние (см. Int 13H с функцией 01H).

На машинах PC и PC/XT эта функция поддерживается только на жестких дисках. Значение, возвращаемое в регистре DL, отражает действительное число физических дисководов, прикрепленных к адаптеру запрошенного дисковода.

Пример программы, использующей функцию 03h Int 13h записи из памяти на диск одного сектора

```
#include<stdio.h>
#include<conio.h>
#include<dos.h>
#include <stdlib.h>
#include <time.h>
//Массив данных для записи в сектор
static char data_write[512];
//Значение сегмента и смещения буфера в памяти
char *seg,*off;
int disc;
void main(void) {
disc=0x00;//дисковод A
//Выполнить функцию "сброс" для дисковода
{
_AH=0x00;
_DL=disc;
geninterrupt(0x13);
}
//Сегмент: относительный адрес буфера
```

```
//формирование смещения в памяти блока данных для записи
off=(char*) FP_OFF( data_write);
//формирование сегмента в памяти блока данных для записи
seg=(char*) FP_SEG( data_write);
//формирование указателя в памяти блока данных для записи
ptr1=(char*) MK_FP(seg,off);
 //Запись в сектор данных по адресу ES:BX
  do{
  _BX = (int ) off;
  _ES = (int ) seg;
  _AH = 0x03;
  _AL = 1;
  _CH = 0x01;
  _CL = 0x1;
  _DH = 0x00;
  _DL = disc;
  geninterrupt(0x13);
    }while((_AH!=0x0 || _AL==0));
  }
```

Чтение элемента FAT

Выборку элемента 12-битной FAT можно вести по следующему алгоритму: выбирается слово по смещению i*1,5 от начала FAT, где i – номер кластера; если i – четное, содержимое элемента FAT составляют младшие 12 бит этого слова, в противном случае – старшие 12 бит. Для 16-битной FAT слово по смещению i*2 содержит элемент FAT.

```
//выделение памяти под FAT
byte *buff = (byte*) realloc(buff, FatSize*512);

//здесь код чтения FAT в buff

//форматная распечатка FAT
word ss;
for(i=0;i<10; i++);
   if (fat12) {
   int m=(i*3)/2; ss=*(word *)(buff+m);
   if(i%2) /*нечетный элемент */ ss>>=4;
   else /* четный элемент */ ss&=0x0fff;
   if (ss>0x0fef) printf(" %03xH ",ss);
   else printf("%5d ",ss);
   }
  else {
   m=i*2; ss=*(word *)(buff+m);
   if (ss>0xffef) printf("%04xH ",ss);
   else printf("%5d ",ss);
}
```

Порядок выполнения работы

Согласно одному из следующих вариантов задания написать и отладить программу на языке Си или Ассемблер, работающую с накопителем на гибких магнитных дисках, используя прерывание 13h:

1. верифицирование, запись сектора на дискету и чтения сектора с дискеты, а также форматирование дорожки дискеты;
2. вывод следующей информации: количество кластеров на диске, количество секторов на кластер, количество байт на сектор, количество плохих кластеров;
3. чтение содержимого корневого каталога: имя файла, атрибут, дата, время, номер кластера, размер.

Контрольные вопросы

1. Опишите физическую организацию накопителя на гибком магнитном диске.
2. Опишите назначение и формат boot-сектора.
3. Опишите назначение и формат FAT – таблицы размещения файлов.
4. Опишите назначение и формат корневого каталога.
5. Опишите основные функции 13-ого прерывания BIOS для работы с контроллером на гибком магнитном диске.

Содержание отчета

Отчет должен содержать:

1. титульный лист;
2. тему и цель лабораторной работы;
3. задание на лабораторную работу;
4. описание алгоритма программы (блок-схема или текстовое описание);
5. прокомментированный листинг программы;
6. выводы по результатам работы.

Литература

1. Кулаков, В. Программирование на аппаратном уровне: спец. справочник / В. Кулаков. – 2-е изд. – СПб. : Питер, 2003.
2. Деревянко, А. С. Системное программное обеспечение персональных ЭВМ: учеб. пособие / А. С. Деревянко. – Харьков : ХГПУ, 1994.
3. Нортон, П. Программно-аппаратная организация компьютера IBM PC / П. Нортон; пер. с англ. С. Писарева, Б. Шура. – Киев, 1987.
4. Фролов, А. Библиотека системного программиста: Операционная система MS-DOS / А. Фролов, Г. Фролов. – М. :Диалог-МИФИ, 1992.

Лабораторная работа № 2

КЛАВИАТУРА

Цель работы – изучить принципы работы и программирования контроллера клавиатуры.

Теоретические сведения

Процесс взаимодействия системы с клавиатурой представлен на рис. 2.1.

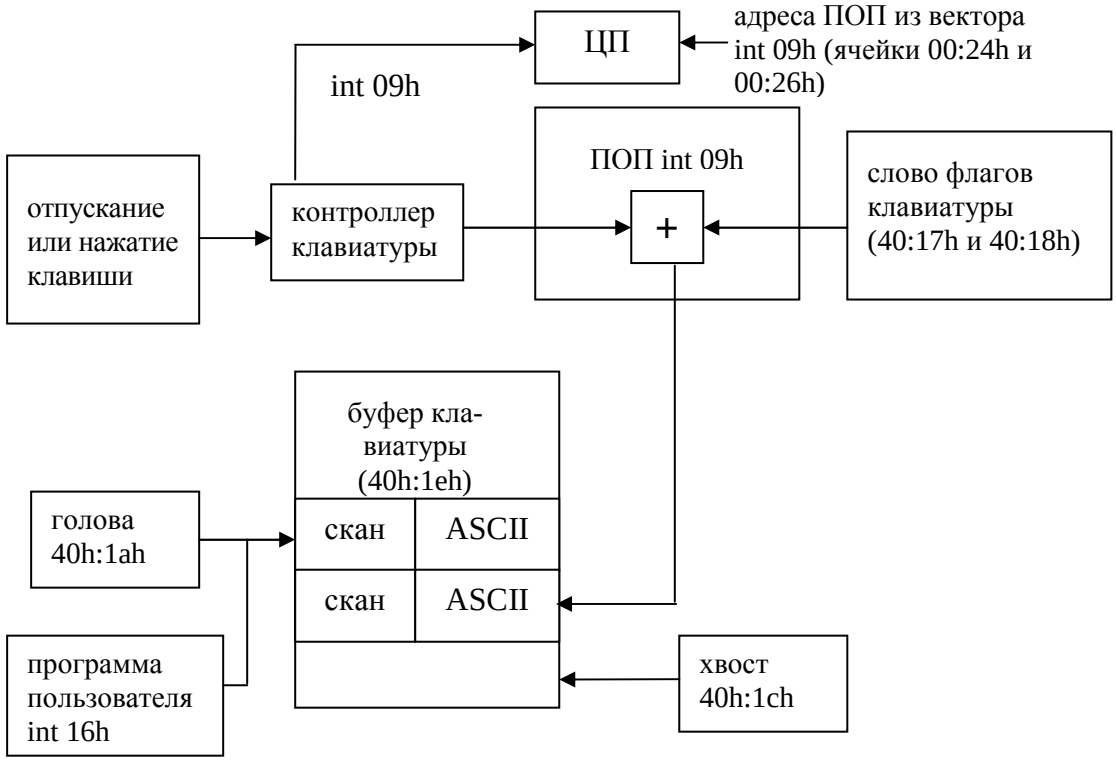

Рис. 2.1. **Процесс взаимодействия системы с клавиатурой**

Работой клавиатуры управляет специальная электронная схема – контроллер клавиатуры. В его функции входит распознавание нажатой клавиши и помещение закрепленного за ней кода в свой выходной регистр (порт), обычно с номером 60h. Код клавиши, поступающий в порт, называется скан-кодом и является, по существу, порядковым номером клавиши, хотя последовательность скан-кодов не всегда совпадает с порядком расположения клавиш на клавиатуре. При этом каждой клавише присвоены как бы два скан-кода, отличающиеся друг от друга на 80h. Один скан-код (меньший,

код нажатия) засылается контроллером в порт 60h при нажатии клавиши, другой (больший, код отпускания) – при ее отпускании. Таким образом, каждое нажатие на клавишу дважды регистрируется в микросхеме интерфейса клавиатуры. Скан-код однозначно указывает на нажатую клавишу, однако по нему нельзя определить, работает ли пользователь на нижнем или верхнем регистре, а также вводит ли он латинский или русские буквы. С другой стороны, скан-коды присвоены всем клавишам клавиатуры, в т. ч. управляющим клавишам <Shift>, <Ctrl>, <Alt>, <Caps Lock> и др.

Таким образом, очевидно, что определение введенного символа должно включать в себя не только считывание скан-кода нажатой клавиши, но и выяснение того, не были ли перед этим нажаты, например, клавиши <Shift> (верхний регистр) или <Caps Lock> (фиксация верхнего регистра). Всем этим анализом занимается программа обработки прерываний от клавиатуры.

Нажатие (а также и отпускание) любой клавиши вызывает сигнал аппаратного (внешнего) прерывания, заставляющий процессор прервать выполняемую программу и перейти на программу обработки прерывания (ПОП) от клавиатуры. Эта программа хранится по фиксированному адресу в постоянном запоминающем устройстве BIOS, являясь, таким образом, элементом «встроенного», или «зашитого», программного обеспечения. Процессор вместе с сигналом прерывания получает еще и тип прерывания, или его номер. За клавиатурой закреплен номер 09h. Адрес программы обработки прерываний от клавиатуры располагается, таким образом, в векторе 09h, занимающем слова с адресами 24h и 26h. Получив тип прерывания и определив по нему адрес вектора, процессор извлекает из вектора адрес программы обработки прерываний и осуществляет переход на ее выполнение. Поскольку программа обработки прерываний от клавиатуры вызывается через вектор 09h, ее часто называют программой INT 09h (INT – от английского Interrupt (прерывание)).

Программа INT 09h, помимо порта 60h, работает еще с двумя областями оперативной памяти: кольцевым буфером ввода, располагаемым по адресам от 40h:1Eh до 40h:3Dh, куда помещаются коды ASCII нажатых клавиш, и словом состояния (словом флагов) клавиатуры, находящимся по адресам 40h:17h и 40h:18h, где фиксируется состояние управляющих клавиш (<Shift>, <Caps Lock>, <Num Lock> и др.). Программа INT 09h, получив управление в результате прерывания от клавиатуры, считывает из порта 60h скан-код и анализирует его значение. Если скан-код принадлежит одной из управляющих клавиш и к тому же представляет собой код нажа-

тия, в слове флагов клавиатуры устанавливается бит (флаг), соответствующий нажатой клавише. Например, при нажатии правой клавиши <Shift> в слове флагов устанавливается бит 0, при нажатии левой клавиши <Shift> – бит 1, при нажатии любой клавиши <Ctrl> – бит 2, а при нажатии <Alt> (тоже любой) – бит 3. Биты флагов сохраняют свое состояние пока клавиши (по одиночке или в любых комбинациях) остаются нажатыми. Если управляющая клавиша отпускается, программа INT 09h получает скан-код отпускания и сбрасывает соответствующий бит в слове флагов.

Кроме состояния указанных клавиш в слове флагов фиксируются еще режимы <Scroll Lock>, <Num Lock>, <Caps Lock> и <lnsert>, а в 101-клавишной клавиатуре на компьютерах PC/AT также состояния клавиш <SysRq>,<Ctrl>-левая, <Alt>-левая и режим паузы (<Ctrl>/<Num Lock>).

Флаги состояния клавиатуры

Два байта, расположенные в ячейках памяти 0040:0017 и 0040:0018, содержат биты, отражающие состояния служебных клавиш и другую полезную информацию (табл. 2.1, 2.2).

Таблица 2.1

Биты статуса по адресу 0040:0017

Бит	Клавиша	Значение, когда бит=1
7	Insert	Режим вставки включен
6	CapsLock	Режим CapsLock включен
5	NumLock	Режим NumLock включен
4	ScrollLock	Режим ScrollLock включен
3	Alt	Клавиша нажата
2	Ctrl	Клавиша нажата
1	Левый shift	Клавиша нажата
0	Правый shift	Клавиша нажата

Таблица 2.2

Биты статуса по адресу 0040:0018

Бит	Клавиша	Значение, когда бит=1
7	Insert	Режим вставки включен
6	CapsLock	Режим CapsLock включен
5	NumLock	Режим NumLock включен
4	ScrollLock	Режим ScrollLock включен
3	Ctrl+NumLock	Задержка (пауза)
2	SysRq	Клавиша нажата
1	Левый Alt	Клавиша нажата
0	Левый Ctrl	Клавиша нажата

Прерывание клавиатуры немедленно обновляет эти биты статуса, как только будет нажата одна из клавиш-переключателей, даже если не было считано ни одного символа из буфера клавиатуры. Прерывание клавиатуры проверяет состояние статусных битов перед тем, как интерпретировать нажатые клавиши, поэтому когда программа меняет один из этих битов, то эффект такой же, как и при физическом нажатии соответствующей клавиши.

При нажатии любой другой клавиши (не управляющей) программа INT 09h считывает из порта 60h ее скан-код нажатия и по таблице трансляции скан-кодов в коды ASCII формирует двухбайтовый код, старший байт которого содержит скан-код, а младший – код ASCII. При этом если скан-код характеризует клавишу, то код ASCII определяет закрепленный за ней символ. За каждой клавишей закреплено, как правило, не менее двух символов («а» и «А», «1» и «!», «2» и «@» и т. д.). Поэтому каждому скан-коду соответствуют, как минимум, два кода ASCII. В процессе трансляции программа INT 09h анализирует состояние флагов, так что если нажата, например, клавиша Q (скан-код 10h, код ASCII буквы Q – 51h, а буквы q – 71h), то формируется двухбайтовый код 1071h, но если клавиша Q нажата при нажатой клавише <Shift> (смена регистра), то результат трансляции составит 1051h. Тот же код 1051h получится, если при нажатии клавиши Q был включен режим <Caps Lock> (заглавные буквы), однако при включенном режиме <Caps Lock> и нажатой клавише <Shift> образуется код 1071h, поскольку в такой ситуации клавиша <Shift> на время нажатия переводит клавиатуру в режим нижнего регистра (строчные буквы).

Полученный в результате трансляции двухбайтовый код засылается программой INT 09h в кольцевой буфер ввода, который служит для синхронизации процессов ввода данных с клавиатуры и приема их выполняемой компьютером программой.

Объем кольцевого буфера составляет 15 слов, при этом дисциплина его обслуживания такова, что коды символов извлекаются из него в том же порядке, в каком они в него поступали. За состоянием буфера следят два указателя. В хвостовом указателе (слово по адресу 40:1Ch) хранится адрес первой свободной ячейки, в головном указателе (40: 1Ah) – адрес самого старого кода, принятого с клавиатуры и еще не востребованного программой. В начале работы, когда буфер пуст, оба указателя (и хвостовой, и головной) указывают на первую ячейку буфера.

Программа INT 09h, сформировав двухбайтовый код, помещает его в буфер по адресу, находящемуся в хвостовом указателе, после чего этот адрес увеличивается на 2, указывая опять на первую свободную ячейку. Ка-

ждое последующее нажатие на какую-либо клавишу добавляет в буфер очередной двухбайтовый код и смещает хвостовой указатель.

Выполняемая программа, желая получить код нажатой клавиши, должна вызвать прерывание INT 16h, которое активизирует драйвер клавиатуры BIOS. Драйвер считывает из кольцевого буфера содержимое ячейки, адрес которой находится в головном указателе, и увеличивает этот адрес на 2. Таким образом, программный запрос на ввод с клавиатуры фактически выполняет прием кода не с клавиатуры, а из кольцевого буфера.

Если компьютер находится в пассивном состоянии ожидания команд DOS с клавиатуры, то за состоянием кольцевого буфера ввода следит командный процессор COMMAND.COM. Как только в буфере появляется код символа, командный процессор с помощью соответствующих системных программ переносит его в свой внутренний буфер командной строки, очищая при этом кольцевой буфер ввода, а также выводит символ на экран, организуя режим эхо-контроля. При получении кода клавиши <Enter> (0Dh) командный процессор предполагает, что ввод команды закончен, анализирует содержимое своего буфера и приступает к выполнению введенной команды. При этом командный процессор работает практически лишь с младшими половинами двухбайтовых кодов символов, именно, с кодами ASCII.

Расширенные коды ASCII

До сих пор рассматривались коды ASCII, которым соответствуют определенные клавиши терминала и которые можно отобразить на экране. Это буквы (прописные и строчные), цифры, знаки препинания и специальные знаки, используемые в программах и командных строках, например, [, $, # и др.

Однако имеется ряд клавиш, которым не назначены какие-то отображаемые на экране символы. Это, например, функциональные клавиши <F1>, <F2>...<F10>; клавиши управления курсором <Home>, <End>, <PgUp>, <PgDn>, <Стрелка вправо>, <Стрелка вниз> и др.

В таблице трансляции, с которой работает программа INT 09h, всем таким скан-кодам соответствует нулевой код ASCII. Поэтому при нажатии, например, <клавиши <F1> (скан-код 3Bh) в кольцевой буфер ввода поступает двухбайтовый код 3B00h, а при нажатии клавиши <Home> (скан-код 47h) – двухбайтовый код 4700h. Двухбайтовые коды, содержащие на месте кода ASCII ноль, называются расширенными кодами ASCII. Эти коды (и соответствующие им клавиши) широко используются для управления программами. Например, в оболочке DOS Norton Commander

нажатие функциональных клавиш вызывает выполнение определенных операций: <FI> – вывод на экран справочника, <F5> – копирование файла, <F8> – его удаление и т. д.

Программы, работающие с расширенными кодами ASCII, должны, очевидно, считав из кольцевого буфера ввода младший байт и убедившись, что он равен нулю, считать далее и старший байт (скан-код) и в зависимости от его значения выполнить соответствующее действие.

Буфер клавиатуры

Буфер клавиатуры занимает 32 байта памяти с адреса 40:1Eh до 40:3Dh. Запись информации в буфер выполняет программа BIOS, адрес которой находится в векторе 09h, чтение функции BIOS прерывания 16h. Буфер клавиатуры рассчитан на 15 нажатий клавиш, генерирующих двухбайтовые коды. Плюс два дополнительных байта для кодов клавиши ENTER.

Буфер организуется как кольцевая очередь, доступ к которой осуществляется с помощью указателя «головы» (адрес 40:1Ah), и указателя «хвоста» (адрес 40:1Ch). Значения, записанные в указателях, равны смещению от сегмента 40h. Указатель «хвоста» задает смещение до слова, где будет записан обработчиком прерывания 9 код буферизуемой клавиши, т. е. первое свободное слово буфера. Указатель «головы» задает смещение слова, которое будет возвращено запросу буферизованного ввода с клавиатуры, сделанного операционной системой или BIOS.

При каждом нажатии клавиши, для которой генерируется 1 двухбайтовый код, обработчик 9 прерывания BIOS, используя текущее значение указателя «головы», записывает в память образованный двухбайтовый код. После этого указатель «головы» увеличивается на 2. Если указатель «головы» перед доступом к буферу указывает на верхнюю границу буфера (на слово 40:3Eh), указатель после записи в буфер «перепрыгивает» на начало буфера, т. е. ему присваивается значение 40:1Eh.

Указатель «головы» используется BIOS-обработчиком прерывания 16h, которое вызывается непосредственно из прикладной программы или функциями MS-DOS ввода с клавиатуры. Если выполняется чтение буфера с разрушением информации (функция AH = 0 прерывания 16h), обработчик читает два байта памяти по адресу, смещение которого задает указатель «головы», а сегмент равен 40h. Затем обработчик прерывания 16h увеличивает (продвигает) указатель «головы» на 2. Если значения указателей «головы» и «хвоста» равны между собой, буфер пуст. В этом случае обработчик прерывания 16 выполняет бесконечный цикл ожидания, условием выхода из которого будет неравенство указателей.

Расширение буфера клавиатуры

Буфер клавиатур имеет одно слабое место. Его емкость ограничена лишь 15 символами. В некоторых ситуациях желательно вводить символы в буфер быстрее, чем программа может их обрабатывать. Но после ввода 15 символов происходит переполнение буфера, раздается звуковой сигнал, и ввод с клавиатуры приходится прекращать.

Переполнение буфера случается довольно редко в тех прикладных программах, которые большую часть времени заняты обработкой данных клавиатуры, это, например, такие программы, как текстовые процессоры.

В программах же обработки данных при пересчете электронных таблиц или компиляции часто происходит отставание. Некоторые программы создают свой собственный буфер клавиатуры для расширения миниатюрного буфера INT 09H. Данные здесь хранятся в том виде, в котором они поступили, и обрабатываются тогда, когда приходит их время. Однако такой подход является скорее исключением, чем правилом.

Для понимания логики расширения буфера полезно рассмотреть достоинства и недостатки существующего варианта буфера клавиатуры. Как уже отмечалось, к основным недостаткам буфера относится его ограниченная длина в 16 слов, позволяющая хранить всего лишь 15 символов. Однако размер буфера определяется только указателями начала и конца буфера. При изменении указателей меняются положение и размер буфера.

Другой недостаток состоит в том, что эти указатели содержат лишь 16-битные смещения. В качестве сегмента, используемого INT 09H и INT 16H, предполагается сегмент в области данных BIOS, имеющий адрес 0040H. Значение сегментного адреса изменить нельзя, а это означает, что буфер клавиатуры должен располагаться внутри 64 кбайт сегмента памяти 0040H. Часть памяти в пределах этих границ используется только в процессе загрузки и доступна для расширения буфера клавиатуры. Простейший способ увеличения буфера клавиатуры – размещение его в этой части памяти.

Перепрограммирование прерывания клавиатуры

В ответ на выдачу в порт 60h скан-кода вызывается прерывание клавиатуры INT9h. Задача этого прерывания – преобразовать скан-код символа, основываясь на состоянии клавиш-переключателей, и поместить его в буфер клавиатуры. Прерывания «ввода с клавиатуры» DOS и BIOS на самом деле всего лишь прерывания «ввода из буфера клавиатуры». В прерывании клавиатуры можно выделить три основных шага:

- прочитать скан-код и послать клавиатуре подтверждающий сигнал;
- преобразовать скан-код в номер кода или в установку регистра статуса клавиш-переключателей;
- поместить код клавиши в буфер клавиатуры.

В момент вызова прерывания скан-код будет находиться по адресу 60h. Поэтому сначала надо этот код прочитать и сохранить на стеке. Затем используется адрес 61h, чтобы послать сигнал подтверждения микропроцессору клавиатуры (необходимо установить бит 7 в 1, а затем сразу изменить его назад в 0). После этого скан-код анализируется на предмет того, была ли клавиша нажата (код нажатия) или отпущена (код освобождения) и обрабатываются все коды нажатия. После того как введенный символ идентифицирован, процедура ввода с клавиатуры должна найти соответствующий ему код ASCII или расширенный код и поместить его в буфер клавиатуры. Для этого процедура должна сначала проверить, имеется ли в буфере место для следующего символа. Если место есть, то для вставки символа в буфер, необходимо поместить его в позицию, на которую указывает «хвост» буфера и затем увеличить указатель «хвоста» на 2 байта.

Следующий код на языке Си представляет собой программу обработки прерывания от клавиатуры, которая распознаёт нажатие «горячей» комбинации клавиш LeftCtrl+RightShift+F3; при первом нажатии «горячей» комбинации переходит в режим блокировки ввода клавиши 3, при втором – отменять этот режим; при этом системная обработка всех других клавиш не нарушается.

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

#ifdef__cplusplus
#define__CPPARGS...
#else
#define__CPPARGS
#endif

/*адрес старого обработчика прерывания 9h*/
void interrupt (*old9)(__CPPARGS);

/*адрес нового обработчика прерывания 9h*/
void interrupt new9(__CPPARGS);

unsigned char F3_code=61; /* скан-код "F3" */
```

```
unsigned char key3_code=4;/* скан-код "3" */
char f=0;                 /* флаг блокирования:
                          1-клавиша заблокирована*/

/*-------------------------------------------------------*/
void main()
{
  char string[80]; /*символьный массив*/
  clrscr();

 /* запоминаем адрес старого обработчика прерывания 9h*/
  old9=getvect(9);
 /* записываем в таблицу векторов прерываний адрес нового обра-
ботчика прерывания*/
  setvect(9, new9);

  printf("\n\n\r\"hot\" combination: ");
  printf("Left Shift, Right Ctrl, F3\n\r");
  printf("Key blocking: ");
  printf("3");
   /*
   проверка реакции программы на нажатие "горячей" комбинации
   клавиш и блокирование/разблокирование ввода клавиши "3"
   */
  printf("\r\nInput symbols>");
  scanf("%s",string);
  /*
    восстанавливаем в таблице векторов прерываний адрес старого
    обработчика
  */
  setvect(9,old9);

}
/*-------------------------------------------------------*/

/*
процедура обработки прерывания от клавиатуры
*/
void interrupt new9(__CPPARGS)
{
/*
c - переменная, которая используется для подтверждения приема
из клавиатуры, в случае, если была нажата клавиша "3", а флаг
f показывал, что эта клавиша заблокирована;
x, y - переменные, которые используются для сохранения коорди-
нат курсора на экране в момент вызова процедуры обработки пре-
рывания;
*/
  unsigned char c,x,y;
  /*
```

```
byte17 - байт флага состояния клавиатуры в области данных
BIOS   по адресу 0040:0017;
byte18 - байт флага состояния клавиатуры в области данных
BIOS по адресу 0040:0018;
*/
unsigned char byte17,byte18;
/*
 mask - маска, которая используется для определения
 нажатия клавиши левый Shift (в этом случае бит 1
 в byte17 установлен в 1);
 mask17 - маска, которая используется для определения
 нажатия клавиши Ctrl (в этом случае бит 2 в byte17
 установлен в 1);
 mask18 - маска, которая используется для определения
 нажатия клавиши левый Ctrl (в этом случае бит 0
 в byte18 установлен в 1);
*/
unsigned char mask=0x02;
unsigned char mask17=0x04;
unsigned char mask18=0x01;
/*
выделяем биты 1и 2 в флаге по адресу 0040:0017
*/
byte17=peekb(0x40,0x17);
/*
 выделяем бит 0 в флаге состояния клавиатуры по адресу
 0040:0018
*/
byte18=peekb(0x40,0x18);
/*
если нажата F3 и левый Shift и правый Ctrl
*/
if((inportb(0x60)==F3_code)&&(byte17&mask)&&
             (byte17&mask17)&&(!(byte18&mask18)))
{
 cputs("\7");
 x=wherex();
 y=wherey();
 gotoxy(55,3);

 if(f==0)
 {
  f=1;
  printf("Key \"3\" blocking ");
 }
 else
 {
  f=0;
  printf("Key \"3\" unblocking");
 }
 gotoxy(x,y);
```

```
  /*восстанавливаем старый обработчик прерываний*/
  (*old9)();
  }

/*если режим блокирования и нажата клавиша 3*/
if( (f==1) && (inportb(0x60)==key3_code) )
  {
   /*
   посылаем подтверждение приема в клавиатуру. Для этого в порт
   61h на короткое время выставляем "1" по шине старшего разря-
   да.
   */
  c=inportb(0x61);
   outportb(0x61,c|0x80);
   outportb(0x61,c);
   /*сброс контроллера прерываний*/
   outportb(0x20,0x20);
  }
  else
   /*восстанавливаем старый обработчик прерываний*/
   (*old9)();
}
```

Описание используемых программных средств

Отладчик debug – программа для проверки и отладки исполнительных файлов MS-DOS. Выполненная без параметров команда debug запускает программу Debug.exe и выводит приглашение команды debug, представленное дефисом (-). Выход из программы – команда q (quit).

Программа KeyView позволяет заглянуть в буфер клавиатуры и изучить его работу при записи и выборке данных. Буфер клавиатуры отображается на экране в виде двух прямоугольных полос, одна над другой. Каждая полоса разделена на восемь двухбайтовых ячеек, т. е. всего 16 двухбайтовых позиций в памяти. Адреса в буфере идут последовательно слева направо, начиная с начала буфера (смещение 1Eh) в левой части верхней полосы, и заканчивается концом буфера – в правой части нижней полосы. В нижней части каждой ячейки показаны текущие ASCII и скан-коды, находящиеся в буфере. Порядок расположения ASCII и скан-кодов соответствует их расположению в памяти. Над каждой парой ASCII/скан-кодов помещено ее представление в виде ASCII символа. Он отображается только для справок и не входит в содержимое буфера. При запуске программы в буфер клавиатуры программой KeyView помещена комбинация FF для обнаружения расширенной клавиатуры. Для ввода символа необходимо нажать соответствующую клавишу, например «а». В позицию, на которую

указывает «хвост», будет занесен ASCII код 61h и скан-код 1Eh. Введенный символ будет высвечен на экране. Кроме этого указатель «хвоста» сместится на следующую свободную позицию. Указатель «головы» останется неизменным, указывая на «а» – логическое начало буфера. Забор символа производится нажатием клавиши F1, при этом указатель «головы» сместиться на следующую позицию и символ, размещенный в «голове», будет отображен внизу экрана. Когда все 16 элементов буфера заполнятся, т.е. «хвост» достигнет конца буфера (крайне правой ячейки нижней полосы), то он перейдет на начало буфера в левой части верхней полосы. В последнюю ячейку данные не будут занесены, вместо этого раздастся сигнал, из динамика указывающий на переполнение буфера. Это произойдет потому, что указатель «хвоста» обошел буфер по кругу и достиг логического конца. Включение/ выключение режима поддержки расширенной клавиатуры осуществляется с помощью клавиши F2.

Порядок выполнения работы

1. Запустить отладчик DEBUG. Ввести команду D 40:80 L4. Эта команда предписывает вывести дамп из 4 байтов, расположенных в сегменте 40h со смещением 80h. Результат выполнения команды должен выглядеть следующим образом: 0040:0080 1E 00 3E 00. Т. к. в процессорах Intel принята обратная техника записи, то двухбайтная величина смещения, записанная в 40:80h, преобразуется процессором во время считывания в 001Eh, а записанная по адресу 40:82h – в 003Eh. Используются только величины смещений, т. к. BIOS полагает, что сегментный адрес буфера клавиатуры равен 40h (сегментный адрес области данных BIOS).
2. Используя смещение начала и конца буфера, полученные в пункте 1, убедиться, что буфер рассчитан на 16 элементов.
3. Просмотреть содержимое буфера. Для этого запустить отладчик DEBUG и ввести команду D 40:1E L20. В правой части дампа представлено содержимое памяти в алфавитно-цифровом виде. По нечетным адресам расположены скан-коды нажатых клавиш, а по четным (слева от скан-кодов) – ASCII-коды. Найти в дампе команду D 40:1E L20.
4. Просмотреть текущее положение «головы» и «хвоста», хранящиеся в двух последовательных словах (2 байта) по 40:1Ah и 40:1Ch. Для этого в отладчике DEBUG ввести команду D 40:1A L4.
5. С помощью программы KeyView изучить отличия скан-кодов цифровых клавиш на основной и дополнительной клавиатуре.
6. Изучить с помощью программы KeyView реакцию системы на Ctrl/C и на Ctrl/Break. Ввести в буфер несколько символов и нажать

Ctrl/Break. Затем очистить буфер, снова ввести в буфер несколько символов и нажать Ctrl/C. Пронаблюдать за изменениями буфера в обоих случаях. В операционной системе DOS команды Ctrl/Break и Ctrl/C взаимозаменяемые. Хотя в действительности это не совсем так. Ctrl/Break – это команда INT 09h, а Ctrl/C – команда DOS.

7. Запустить программу Key.com. Нажать любое сочетание клавиш, удерживая клавишу Shift. Убедиться, что при этом клавиатура посылает один и тот же скан-код независимо от клавиши Shift. Нажать !/1, на экране отобразится скан-код 02h, затем нажать Alt и !/1. Код все рано останется 02h. Запустить KeyView. Программа покажет, что при нажатии !/1 заносится 31h (ASCII код единицы) и скан-код 02h. Но при нажатии комбинации Alt и !/1 INT 09h заносит в буфер ASCII-код 00h и скан-код 78h. Значение 78h превышает максимальное значение клавиш на клавиатуре. Программа INT 09h преобразует эту комбинацию клавиш в специальный скан-код так, что прикладная программа может быстро определить комбинацию нажатых клавиш без дополнительной проверки байта Shift-статуса. Как ASCII, так и скан-коды, сформированные INT 09h, не отражают реальных кодов поступающих от клавиатуры. Проверить и записать расширенные коды для следующих сочетаний клавиш Alt и F4, Ctrl и F4, Shift и F4, используя KeyView и Key.

8. Очистить буфер, нажимая F1. Ввести в буфер пять символов, нажимая «A», F10, F11, F12, «B». Указатель «хвоста» переместится в соответствии с введенными пятью символами. Перевести переключатель F2 в положение «INACTIVE». Выбрать все пять символов, нажимая F1. Клавиша «A» вернется назад как «A», а F10 – как пробел (ASCII-0). После того как будет выбран код F10, указатель пропустит F11 и F12, тем самым удаляя их из буфера, и остановится на «B». Нажать F1 и выбрать символ «B». Объясняется это тем, что при деактивации поддержки расширенных функций KeyView использует старые функции INT 16h выбора символов. Старые функции INT 16h не знают о существовании клавиш F11 и F12, поэтому они игнорируют их коды и пропускают их до тех пор, пока не был обнаружен понятный код «B».

9. Расширить буфер клавиатуры. Для этого набрать и отладить программу KBDBUFF.COM, перемещающую буфер клавиатуры в область памяти, начинающуюся с адреса 0040:0200H. Программа KBDBUFF перемещает буфер простым изменением смещений начала и конца буфера клавиатуры на 0040:0200H и 0040:0300H соответственно. Указатели «головы» и «хвоста» также изменяются с тем, чтобы указывать внутрь нового буфера. При загрузке программы оба указателя устанавливаются на начало буфера, что инициализирует буфер как пустой. Для создания

KBDBUFF.COM необходимо сформировать файл KBDBUFF.SCR, содержащий следующие команды для отладчика DEBUG (после команды INT 20H и после последней команды Q в файле KBDBUFF.SCR должна быть пустая строка, комментарии к программе KBDBUFF, приведенные в листинге, в файл не вставляются):

```
n KBDBUFF.COM   ; Имя файла KBDBUFF.COM
a   ; Ассемблер
mov ax, 0040   ; 40H занести в сегмент данных
mov ds, ax
cli   ; запретить прерывания
mov word ptr [001A], 0200; смещение «головы» 200H
mov word ptr [001C], 0200; смещение «хвоста» 200H
mov word ptr [0080], 0200; смещение начала буфера 200H
mov word ptr [0082], 0300; смещение конца буфера ЗООН
sti   ; разрешить прерывания
int 20   ; закончить программу
; выход из ассемблера
rcx   ; в регистре CX длина файла
  ; длина файла = 21H байт
w   ; записать файл
q   ; выход из отладчика
```

Затем необходимо направить эти команды в отладчик, введя в командной строке DOS: DEBUG < KBDBUFF.SCR. После того, как программа KBDBUFF создана, необходимо поместить команду KBDBUFF в начало файла AUTOEXEC.BAT. Теперь всякий раз при загрузке начало буфера клавиатуры будет перемещаться в абсолютную ячейку памяти 00600H, а емкость буфера будет увеличиваться до 256/2 = 128 символов.

10. Согласно одному из следующих вариантов задания написать и отладить программу на языке Си или Ассемблер, работающую с клавиатурой, используя прерывание 9h:

- обработка прерывания от клавиатуры, распознавание нажатия «горячей» комбинации клавиш; при первом нажатии «горячей» комбинации переходить в режим блокировки ввода заданной клавиши, при втором – отменять этот режим; при этом системная обработка всех других клавиш нарушаться не должна;
- переопределить режимы NumLock, CapsLock, ScrollLock так, чтобы они включались при нажатии клавиш 1, 2, 3 соответственно;

- при нажатии на клавишу F1 вывести на экран статус клавиш NumLock, CapsLock, ScrollLock;
- при нажатии на клавишу 1 вывести на экран содержимое буфера и произвести его очистку.

Контрольные вопросы

1. Опишите процесс взаимодействия системы с клавиатурой.
2. Какое прерывание закреплено за клавиатурой?
3. Опишите работу прерывания от клавиатуры.
4. Для чего используется, и что представляет собой ASCII-код клавиши?
5. Опишите работу буфера клавиатуры.
6. Для чего используется и что представляет собой скан-код клавиши?
7. Для чего используются флаги состояния клавиатуры.
8. Опишите порядок перепрограммирования прерывания клавиатуры.

Содержание отчета

Отчет должен содержать:

1. титульный лист;
2. тему и цель лабораторной работы;
3. задание на лабораторную работу;
4. выводы и результаты по каждому выполненному пункту;
5. описание алгоритма программы (блок-схема или текстовое описание);
6. прокомментированный листинг программы;
7. выводы по результатам работы.

Литература

1. Кулаков, В. Программирование на аппаратном уровне: спец. справочник / В. Кулаков. – 2-е изд. – СПб. : Питер, 2003.

2. Деревянко, А. С. Системное программное обеспечение персональных ЭВМ: учеб. пособие / А. С. Деревянко. – Харьков : ХГПУ, 1994.

3. Нортон, П. Программно-аппаратная организация компьютера IBM PC / П. Нортон; пер. с англ. С. Писарева, Б. Шура. – Киев, 1987.

4. Журден, Р. Справочник программиста на персональном компьютере фирмы IBM / Р. Журден. – Р.-Prentice-Hall Press, 1986.

Лабораторная работа № 3

ИЗУЧЕНИЕ ПРИНЦИПОВ РАБОТЫ ПАРАЛЛЕЛЬНОГО ПОРТА

Цель работы – изучить основные принципы работы и программирования параллельного порта.

Теоретические сведения

Общее описание

В минимальную конфигурацию самого первого компьютера IBM PC был включен контроллер параллельного порта. Его основное назначение – подключение принтера к персональному компьютеру с использованием параллельного интерфейса. Все последующие модели компьютеров (XT, AT и PS/2) также имеют в своей конфигурации параллельный порт.

Помимо стандартного режима использования (режим совместимости) параллельный порт, установленный в моделях PS/2, допускает работу в расширенном режиме, который поддерживает двунаправленный ввод-вывод. Использование расширенного режима создает предпосылки для подключения к параллельному порту активных нестандартных устройств. Некоторые современные IBM совместимые компьютеры укомплектовываются контроллером параллельного порта, который также может работать в двунаправленном режиме, хотя такие компьютеры и не являются моделью PS/2.

BIOS поддерживает до 3-х параллельных портов, которые определяются на этапе начального тестирования компьютера программой POST (Power-On-Self-Test). При обнаружении соответствующего порта BIOS записывает адрес его регистра данных, начиная с адреса 0:408H, и присваивает ему имя LPTn (n может принимать значения от 1 до 3). BIOS может работать и с 4-мя параллельными портами, однако для этого программист должен сам позаботиться о том, чтобы соответствующий адрес регистра данных был записан в определенную для LPT4 область – по адресу 0:410h.

Двунаправленный порт обычно используется в режиме совместимости – именно этот режим устанавливается изначально при выполнении программы POST. Однако программист может использовать расширенный режим работы порта для подключения нестандартной аппаратуры. В этом случае на компьютерах PS/2 выбор расширенного режима работы параллельного порта производится при конфигурации аппаратуры компьютера

путем записи нулевого значения в бит 7 порта 0102h. В некоторых компьютерах двунаправленный режим контроллера параллельного порта может быть выбран при выполнении программы начальной конфигурации BIOS.

Большинство устройств печати подключаются к компьютеру через контроллер параллельного интерфейса и продаются вместе с соединительным кабелем, имеющим со стороны компьютера стандартный 25-ти штырьковый разъем. Значения сигналов на контактах данного разъема показаны в табл. 3.1. Минус впереди названия сигнала (например, – STROBE) показывает, что сигнал инверсный (обратный). Это означает, что при подаче на этот контакт уровня логического нуля контроллер проинвертирует этот сигнал, т. е. изменит на противоположный (на лог. 1).

Таблица 3.1

Назначение контактов 8-ми разрядного параллельного порта

Контакт	Название
1	- STROBE (Синхронизация). Низкий уровень сигнала – запись, высокий уровень сигнала – чтение
2	Данные, разряд 0
3	Данные, разряд 1
4	Данные, разряд 2
5	Данные, разряд 3
6	Данные, разряд 4
7	Данные, разряд 5
8	Данные, разряд 6
9	Данные, разряд 7
10	- ASK (Подтверждение).
11	BUSY (Занято)
12	PE (Отсутствие бумаги)
13	SELECT (Выбор)
14	- AUTO FEED XT (Автоподача). Когда уровень низкий, сообщает о передаче данных
15	ERROR (Ошибка)
16	- INIT (Инициализация)
17	- SELECT IN (Разрешение). Когда уровень низкий, сообщает о передаче адреса
18 - 25	GROUND (Уровень 0)

Необходимо иметь в виду, что значения контактов на разъеме со стороны принтера могут быть разными у разных принтеров. Таким образом, соединительный кабель в общем случае не является унифицированным и может потребовать перепайки. В данном случае следует руководствоваться рисунком и описанием параллельного интерфейса, приводимым в документации на каждый принтер.

Параллельный порт обеспечивает номинальный ток до 0,55мА, а пиковый – до 20мА. Выходное напряжение порта соответствует уровню TTL, т. е. логическая единица представлена напряжением 5,0 В, логический ноль – 0,5 В.

Описание регистров

В архитектуре IBM PC регистры ввода/ вывода соединяются с системной шиной через порты. Порт рассматривается как байтовая ячейка в пространстве адресов ввода/вывода; диапазон адресов – от 0 до 0ffff. Поскольку общая размерность регистров устройства, как правило, превышает 1 байт, для соединения с устройством используется массив портов. Начальный адрес массива портов при соединении с некоторым устройством называют базовым адресом этого устройства.

Способ доступа к регистрам устройства определяется схемой их подключения к портам ввода/ вывода. Наиболее просто доступ организуется при непосредственном подключении регистров к портам. В этом случае чтение или запись порта означают чтение или запись соответствующего регистра устройства. Адаптер параллельной связи организован именно таким образом: каждый из трех байтовых регистров этого адаптера подключен к отдельному порту.

Регистр данных. Регистр данных параллельного порта представляет собой 8-миразрядный регистр, доступный по чтению и записи. Регистр располагается по базовому адресу. Адрес базового порта хранится в BIOS по адресу: для LPT1 – 0:0408h, для LPT2 – 0:040Ah, для LPT3 – 0:040Ch. Биты D7-D0 регистра данных определяют значения передаваемого или считываемого байта информации. Битам регистра назначены соответственно разъемы от 9 до 2 в стандартном 25-ти штырьковом разъеме.

В режиме совместимости запись данных приводит к их немедленной передаче в линию. Передача данных в двунаправленном режиме несколько сложнее и управляется путем записи бита направления в регистр управления. Только при выполнении записи (бит направления равен 0), байт передается в линию, в противном случае запись значения в регистр производится, но в линию байт не передается. Операция чтения регистра данных приводит к чтению последнего записанного значения в режиме совместимости и при передаче в двунаправленном режиме. При выполнении чтения при приеме (бит направления равен 1) в двунаправленном режиме из регистра считывается значение линии, т. е. принимаемого байта.

Регистр состояния. Регистр состояния параллельного порта представляет собой 8-ми разрядный регистр, доступный только по чтению. В табл. 3.2 приведен формат регистра состояния и описаны значения битов регистра. Регистр состояния находится по адресу базовый адрес + 1.

Таблица 3.2

Формат регистра состояния

Бит	Название	Назначение
7	BUSY	Определяет инвертированное состояние линии «занято»: 0 – устройство занято; 1 – стройство свободно. Сигнал «занято» может формироваться из-за ошибки, а также в том случае, когда принтер отключен или отсутствует
6	ACK	Показывает инвертированное состояние готовности к приему очередного байта: 0 –устройство готово к приему; 1 – стройство не готово к приему
5	PE	Показывает текущий сигнал от принтера о состоянии бумаги. Бит устанавливается в 1, когда принтер вырабатывает сигнал конец бумаги (Paper End)
4	SEL	Указывает текущее состояние сигнала «выборка» (Select) и устанавливается в 1, когда устройство было выбрано
3	ERR	Задает инвертированное состояние ошибки в устройстве. Бит устанавливается в 0 при выработке принтером сигнала ошибки (Error)
2	IRQS	Принимает значение 0, когда устройство подтвердило прием предыдущего байта информации сигналом подтверждения (ACKnowlegement). Значение данного бита имеет смысл только для двунаправленного параллельного порта. Режим подтверждения устройством приема символа и выработки прерывания управляется битом IRQE управляющего порта. Обычно, прерывание от устройства LPT1 поступает на IRQ5, а от LPT2 – на IRQ7
1	RESERVE	Зарезервирован, должен быть равен 0
0	RESERVE	Зарезервирован, должен быть равен 0

Регистр управления. Регистр управления параллельного порта представляет собой 8-ми разрядный регистр, доступный по чтению и записи. Формат регистра управления описывается табл. 3.3 Регистр управления находится по адресу базовый адрес + 2.

Таблица 3.3

Формат регистра управления

Бит	Название	Назначение
7	RESERVE	Зарезервирован, должен быть равен 0
6	RESERVE	Зарезервирован, должен быть равен 0
5	DIR	Используется для задания типа операции при работе в расширенном режиме (или направления передачи данных): 0 – операция записи; 1 – перация чтения. Этот бит имеет смысл только для двунаправленного параллельного порта. В режиме совместимости, а также для обычного параллельного порта значение бита игнорируется
4	IRQE	Управляет прерыванием. Когда бит равен 1, параллельный порт посылает прерывание при выработке сигнала АСК со стороны устройства
3	SELIN	Управляет состоянием сигнала выборки устройства (Select In). Когда бит установлен в 1, устройство считается выбранным. Данной линии соответствует разъем 17

Окончание табл.

2	INIT	Управляет инвертированным состоянием сигнала инициализации устройства (Init). При этом установка нулевого значения бита означает инициализацию принтера
1	AFD	Управляет состоянием сигнала автоматический прогон строки (Automatic Feed XT). Когда бит установлен в 1, принтер после печати каждой строки будет автоматически переходить на новую строку. Следует отметить, что, несмотря на упоминание XT в названии сигнала, использование данного бита не ограничено для других моделей компьютеров. Обычно установка данного бита не требуется, т. к. большинство редакторов, используемых для подготовки документов, формируют символ перехода на следующую строку в конце каждой строки текста
0	STRB	Управляет синхронной передачей данных в устройство. Когда он принимает значение 1, передаваемые данные могут считываться с линий данных

Программирование адаптера параллельного интерфейса на уровне портов ввода/ вывода

Для получения базового адреса LPT порта необходимо обратиться к ячейке памяти BIOS по соответствующему адресу (0:0408h для LPT1):

```
mov ax, 0
mov es, ax
mov dx, es:[0408h] ; в dx адрес базового регистра LPT1
```

Для инициализации адаптера последовательного интерфейса необходимо сбросить бит 2 регистра управления в нуль на время, равное тысяче тактов пустого цикла. В этот момент нужно, чтобы был установлен только бит 3 регистра управления. Таким образом, для инициализации адаптера необходимо:

- в регистр управления записать значение 12 (1100 – установка битов 2 и 3);
- сделать задержку;
- послать в регистр управления значение 8 (1000 – сброс бита 2).

Для записи символа в параллельный порт необходимо:

- записать очередной выводимый символ в регистр данных;
- проверить занятость устройства и организовать ожидание его освобождения: опрос бита 7 регистра состояния;
- выдать стробирующий импульс: установить и сразу сбросить бит 0 регистра управления.

Для организации цикла в языке Ассемблер используется конструкция типа:

```
mov cx, число циклов
```

```
label:
.....
loop label
```

В регистр cx заносится количество циклов. Метка label указывает на начало цикла. Команда loop выполняется до тех пор, пока значение регистра cx не равно 0. После выполнения команды loop значение регистра cx уменьшается на единицу автоматически.

Команды ввода/ вывода языка Ассемблер

Каждое устройство ввода-вывода имеет один или несколько встроенных регистров, каждый из которых имеет соответствующий адрес. Всего существует 65 536 адресов ввода-вывода. Из них 512 адресов назначены системному каналу ввода/ вывода и могут использоваться различными адаптерами. Другие 256 адресов используются системной шиной для управления подключенными туда устройствами ввода-вывода.

Команда IN пересылает данные из устройства ввода-вывода в регистр AL. Если адрес устройства находится в пределах 0 – 255, то он может содержаться в команде как непосредственное значение. Если адрес больше 255, то он сообщается косвенно и содержится в регистре DX.

Команда OUT записывает содержимое регистра AL в регистр устройства ввода-вывода, адрес которого указывается так же, как и в команде IN. В случае использования микропроцессора Intel x86 команды IN и OUT, могут пересылать слова в устройства ввода вывода. Источником и приемником в этом случае является регистр AX.

Описание используемых аппаратных средств

В ходе выполнения работы используется испытательный стенд, схема которого приведена на рис. 3.1.

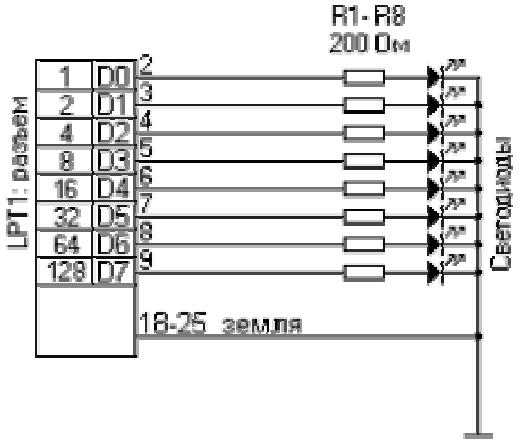

Рис. 3.1. **Схема испытательного стенда**

Данный стенд позволяет наглядно убедиться в работоспособности написанной программы. При появлении на контакте уровня логической единицы, соответствующий светодиод загорится.

Порядок выполнения работы

1. Выключить питание компьютера.
2. Подсоединить с помощью кабеля типа CENTRONIC, испытательный стенд к компьютеру.
3. Включить питание компьютера.
4. Разработать и отладить программу на языке Си или Ассемблер записи символа в параллельный порт.

Контрольные вопросы

1. Опишите режимы работы параллельного порта.
2. Опишите назначение контактов параллельного порта.
3. Какие существуют регистры для программирования параллельного порта?
4. Опишите порядок программирования параллельного интерфейса на уровне портов ввода/ вывода.

Содержание отчета

Отчет должен содержать:

1. титульный лист;
2. тему и цель лабораторной работы;
3. задание на лабораторную работу;
4. описание алгоритма программы (блок-схема или текстовое описание);
5. прокомментированный листинг программы;
6. выводы по результатам работы.

Литература

1. Кулаков, В. Программирование на аппаратном уровне: спец. справочник / В. Кулаков. – 2-е изд. – СПб. : Питер, 2003.
2. Деревянко, А. С. Системное программное обеспечение персональных ЭВМ: учеб. пособие / А. С. Деревянко. – Харьков : ХГПУ, 1994.
3. Нортон, П. Программно-аппаратная организация компьютера IBM PC / П. Нортон; пер. с англ. С. Писарева, Б. Шура. – Киев, 1987.
4. Журден, Р. Справочник программиста на персональном компьютере фирмы IBM / Р. Журден. – P.-Prentice-Hall Press, 1986.

Лабораторная работа № 4

ОСНОВЫ РАБОТЫ С УСТРОЙСТВОМ ВВОДА ТИПА «МЫШЬ»

Цель работы – изучить функции прерывания драйвера мыши 33h и в соответствии с индивидуальным заданием разработать алгоритм и программу, работающую с манипулятором типа «мышь», используя прерывание 33h.

Теоретические сведения

Мышь является устройством, предназначенным для ввода координат и подачи команд. Универсальные программы управления манипулятором – драйверы мыши – были предложены фирмой Microsoft. Они обеспечивают унифицированный интерфейс для работы с манипуляторами «мышь» любого типа. Драйвер позволяет выполнять свыше 40 различных функций. Существует несколько фактических стандартов на способы подключения координатных устройств к компьютеру. Эти стандарты предусматривают различные способы подключения устройства и различные форматы передачи данных. В настоящее время применяются два основных способа подключения мыши к персональному компьютеру – подключение через последовательный порт (Serial Mouse) и подключение через разъём дополнительного устройства PS/2.

Форматы передачи данных Serial Mouse

Внутренняя структура драйвера мыши определяется, в первую очередь, используемым мышью форматом передачи данных. Для устройств, подключаемых через последовательный порт, применяется ряд различных форматов: группа форматов, базирующихся на протоколе MS Mouse, и формат PC Mouse. Группа форматов Microsoft Mouse в настоящее время стала основной для координатных устройств, подключаемых к последовательному порту, вытеснив из этой области другие виды протоколов. Все форматы этой группы являются расширениями 7-битного формата данных фирмы Microsoft, приведенного в табл. 4.1. Обозначения в таблице расшифровываются следующим образом:

- X0 – X7 – перемещение по оси X (целое число со знаком);
- Y0 – Y7 – перемещение по оси Y (целое число со знаком);
- L – состояние левой кнопки (0 – отпущена, 1 – нажата);
- R – состояние правой кнопки (0 – отпущена, 1 – нажата).

Таблица 4.1

Стандартный формат Microsoft (MS Mouse)

Номер байта в посылке	Номер бита						
	6	5	4	3	2	1	0
1	1	L	R	Y7	Y6	X7	X6
2	0	X5	X4	X3	X2	X1	X0
3	0	Y5	Y4	Y3	Y2	Y1	Y0

Данный формат был введён для двухкнопочной мыши. Средняя кнопка трёхкнопочной мыши, поддерживающей работу с несколькими протоколами, при работе в режиме MS Mouse эквивалентна правой. Старший бит посылки (бит 6) используется для самоконтроля и синхронизации: признаком начала передачи очередного пакета из трех байт служит единица в этом бите. Программа-обработчик прерывания мыши должна удостовериться, что следующие два байта данных имеют в шестом разряде нулевое значение. В противном случае произошёл сбой в процессе передачи и следует проигнорировать принятый пакет.

Скорость приема-передачи данных принята равной 1 200 бод, длина передаваемого слова – 7 бит, контроль по чётности не используется, число стоповых битов равно 1 (прежде чем начинать работу с мышью, нужно загрузить эти значения в регистры последовательного порта, к которому она подключена). Передача данных производится только в том случае, если изменяется состояние кнопок мыши или координат X и Y. Ось Y в режиме MS Mouse направлена сверху вниз, как у дисплея. Для обеспечения нормальной работы с трехкнопочными устройствами протокол Microsoft пришлось дополнить четвертым байтом, который служит одной единственной цели – обеспечивает передачу состояния средней кнопки мыши (в пятом разряде, обозначенном символом M). Обязательно нужно учитывать, что передача пакета из четырех слов выполняется только в случае изменения состояния средней кнопки, а в остальных случаях передаются первые три слова. Данный протокол, получивший название Microsoft Plus, показан в табл. 4.2.

Таблица 4.2

Формат Microsoft Plus (M+) для трехкнопочной мыши

Номер байта в посылке	Номер бита						
	6	5	4	3	2	1	0
1	1	L	R	Y7	Y6	X7	X6
2	0	X5	X4	X3	X2	X1	X0
3	0	Y5	Y4	Y3	Y2	Y1	Y0
4	0	M	0	0	0	0	0

Предложенный фирмой IBM формат Mouse System (PC Mouse), который показан в табл. 4.3, в настоящее время почти не применятся, однако до сих пор поддерживается некоторыми универсальными устройствами. Особенность данного формата состоит в том, что для определения перемещения по оси X нужно сложить значения X' и X" (байты 2 и 4), а для определения перемещения по оси Y – значения Y' и Y" (байты 3 и 5). Такой способ передачи координаты предназначен для обеспечения уникальности признака начала кадра (единица в бите 7 и нули в битах 3 – 6 первого байта посылки). Ось Y мыши в данном формате направлена вверх, т. е. противоположно оси Y дисплея.

Таблица 4.3

Формат Mouse System (PC Mouse)

Номер байта в посылке	Номер бита							
	7	6	5	4	3	2	1	0
1	1	0	0	0	0	L	M	R
2	X7'	X6'	X5'	X4'	X3'	X2'	X1'	X0'
3	Y7'	Y6'	Y5'	Y4'	Y3'	Y2'	Y1'	Y0'
4	X7"	X6"	X5"	X4"	X3"	X2"	X1"	X0"
5	Y7"	Y6"	Y5"	Y4"	Y3"	Y2"	Y1"	Y0"

Для работы с устройством типа PC Mouse необходимо установить следующие параметры последовательного порта: скорость приема-передачи – 1 200 бод, длина передаваемого слова – 8 бит, контроль по четности не используется, число стоповых бит равно 1. Передача данных производится только в том случае, если изменяется состояние кнопок или координат X и Y.

Формат передачи данных PS/2

Стандартный формат передачи данных для мыши PS/2-типа, разработанный фирмой IBM, показан в табл. 4.4. Обозначения в таблице расшифровываются следующим образом:

- X0 – X7 – перемещение по оси X;
- Y0 – Y7 – перемещение по оси Y;
- XV – признак возникновения переполнения по X (1 – переполнение);
- YV – признак возникновения переполнения по Y (1 – переполнение);
- XS – знак перемещения по X;
- YS – знак перемещения по Y;
- M – состояние средней кнопки (0 – отпущена, 1 – нажата);

- R – состояние правой кнопки (0 – отпущена, 1 – нажата);
- L – состояние левой кнопки (0 – отпущена, 1 – нажата).

Таблица 4.4

Стандартный формат PS/2 Mouse

Номер байта в посылке	Номер бита							
	7	6	5	4	3	2	1	0
1	YV	XV	YS	XS	1	M	R	L
2	X7	X6	X5	X4	X3	X2	X1	X0
3	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

Особенность данного формата заключается в том, что координаты X и Y являются двоичными 9-разрядными числами (старший разряд – знаковый). Ось Y мыши в данном формате направлена вверх, т. е. противоположно оси Y дисплея. Преимуществом формата PS/2 является простота, а недостатком – отсутствие самосинхронизации (первый байт кода не обладает отличительными признаками, позволяющими обнаружить сбой в порядке принимаемых данных, поэтому контроль приходится осуществлять другими способами). Мышь PS/2 подключаются не к последовательному порту, а к разъёму дополнительного устройства PS/2. Мышь обслуживается тем же контроллером материнской платы, что и клавиатура, т. е. получает команды и передаёт данные через те же порты ввода-вывода. Отличие заключается в том, что при приёме пакета данных от мыши контроллер вырабатывает прерывание IRQ12. Это прерывание необходимо закрепить за мышью с помощью процедуры BIOS SETUP, иначе оно может быть захвачено каким-либо другим устройством и станет для мыши недоступным.

Мышь PS/2 может находиться в одном из режимов работы:

- потоковый режим – мышь выдаёт пакет данных, если произошло изменение координат устройства, нажатие или отпускание кнопки. Максимальная скорость передачи данных определяется установленной частотой дискретизации;
- режим дистанционного управления – мышь осуществляет передачу пакета данных только по запросу со стороны компьютера, т. е. по команде считывания данных Read Data;
- эхо-режим – любой байт данных, переданный компьютером (кроме ECh и FFh), мышь отсылает обратно. После включения напряжения питания или получения команды Reset мышь ожидает примерно 0,5 секунды и посылает компьютеру последовательность байтов: AAh, 00h. Затем по умолчанию устанавливается инкрементальный потоковый режим, масштабирование 1:1, темп передачи 100 пакетов данных в секунду. После этого

мышь отключается и больше никаких операций не производит, пока компьютер не передаст команду Enable (которая активизирует мышь).

Функции прерывания MS-DOS 33h

1. *Функция 00h* – сброс драйвера мыши. Сбрасывает (инициализирует) драйвер мыши.

Входные данные: AX = 0000H.

Выходные данные: AX = состояние мыши;

AX = FFFFH: драйвер мыши установлен;

AX = 0000H: ошибка, драйвер мыши не установлен;

BX = число кнопок мыши.

Примечания. Программа инициализации выполняет следующие задачи: перемещает маркер мыши в центр экрана и стирает его изображение на экране. После разрешения вывода маркера маркер мыши по умолчанию имеет вид инверсного прямоугольника. Этот маркер всегда воспроизводится на нулевой экранной странице независимо от текущего видеорежима. Областью перемещения мыши становится весь экран. Устанавливает обработчик событий (event handler) (по умолчанию не устанавливается). Устанавливает эмуляцию светового пера (по умолчанию не устанавливается). Задает скорость перемещения маркера мыши. По умолчанию относительная скорость равно 8 микки на 8 горизонтальных элементов и 16 микки на 16 вертикальных. Задает максимальную скорость мыши (по умолчанию равна 64 микки в секунду). Микки (mickey) – это самое маленькое расстояние, которое отслеживается мышью. Оно примерно равно 1/200 дюйма.

2. *Функция 01h* – вывод маркера мыши. Выводит на экран маркер мыши. Этот маркер отображает любое движение мыши, перемещаемой пользователем.

Входные данные: AX = 0001H.

Выходные данные: отсутствуют.

Примечания. Эта функция увеличивает на единицу значение внутреннего счетчика, который определяет, должен ли маркер мыши быть виден на экране. После инициализации драйвера мыши функцией 00H этот счетчик содержит -1 (т. е. маркер мыши не виден). Когда после обращения к функции 01H значение этого счетчика становится нулевым, маркер мыши появляется на экране. Драйвер мыши отображает перемещение мыши даже тогда, когда маркер не воспроизводится на экране. После обращения к этой функции маркер может появляться не в том месте, в каком он находился в момент удаления маркера в результате обращения к функции 00H или 02H.

3. *Функция 02h* – удаление маркера мыши. Удаляет маркер мыши с экрана.

Входные данные: AX = 0002H.

Выходные данные: отсутствуют.

Примечания. Эта функция уменьшает на единицу значение внутреннего счетчика, который определяет, должен ли маркер мыши быть виден на экране. Если счетчик имеет значение 0, то маркер мыши воспроизводится на экране; если счетчик имеет значение -1, то маркер удаляется с экрана. Драйвер мыши отображает перемещение мыши даже тогда, когда маркер не воспроизводится на экране. После обращения к этой функции маркер может появляться не в том месте, в каком он находился в момент удаления маркера в результате обращения к функции 00H или 02H.

4. *Функция 03h* – чтение положения маркера/ состояния кнопок. Возвращает текущее положение маркера мыши и текущее состояние кнопок мыши.

Входные данные: AX = 0003H.

Выходные данные*:* BX = состояние кнопок мыши;

Бит 0 = 1: нажата левая кнопка;

Бит 1 = 1: нажата правая кнопка;

Бит 2 = 1: нажата средняя кнопка;

Биты 3-15: не используются;

CX = координата X;

DX = координата Y.

Примечания. Координаты, возвращаемые в регистрах CX и DX, являются координатами элементов изображения на виртуальном экране мыши, а не физическими координатами на реальном экране. Если на мыши имеются только две кнопки, то информация о центральной кнопке не имеет значения.

5. *Функция 04h* – перемещение маркера мыши. Перемещает активный маркер мыши в указанную точку экрана.

Входные данные: AX = 004H;

CX = координата X;

DX = координата Y.

Выходные данные: отсутствуют.

Примечания. Координаты, возвращаемые в регистрах CX и DX, являются координатами элементов изображения на виртуальном экране мыши, а не физическими координатами на реальном экране. Если указанная в обращении позиция находится за пределами диапазона перемещения маркера мыши, заданного функциями 07H и 08H, то функция корректирует

координаты таким образом, что маркер остается внутри диапазона. Маркер перемещается в новую позицию даже в том случае, если он не воспроизводится на экране. После того, как воспроизведение маркера мыши снова будет разрешено, он появится в новой позиции.

6. *Функция 05h* – определение числа нажатий кнопки мыши. Информирует вызывающую программу о том, сколько раз была нажата указанная кнопка мыши с момента последнего обращения к функции 05H. Функция 05H также сообщает вызывающей программе координаты маркера на экране в момент последнего нажатия кнопки.

Входные данные: AX = 0005H;
BX = кнопка мыши;
BX = 0: левая кнопка мыши;
BX = 1: правая кнопка мыши;
BX = 2: средняя кнопка мыши.

Выходные данные: BX = состояние всех кнопок мыши:
бит 0 = 1: нажата левая кнопка;
бит 1 = 1: нажата правая кнопка;
бит 2 = 1: нажата средняя кнопка;
биты 3-15: не используются;
CX = горизонтальная координата в момент последнего нажатия;
DX = вертикальная координата маркера в момент последнего нажатия.

Примечания. Координаты, возвращаемые в регистрах CX и DX, являются координатами элементов изображения на виртуальном экране мыши, а не физическими координатами на реальном экране. При обращении к этой функции счетчик числа нажатий указанной клавиши сбрасывается в ноль.

7. *Функция 06h* – определение числа отпусканий кнопки мыши. Информирует вызывающую программу о том, сколько раз была отпущена указанная кнопка мыши с момента последнего обращения к функции 06H. Функция 06H также сообщает вызывающей программе координаты маркера на экране в момент последнего отпускания кнопки.

Входные данные: AX = 0006H;
BX = кнопка мыши;
BX = 0: левая кнопка мыши;
BX = 1: правая кнопка мыши;
BX = 2: средняя кнопка мыши.

Выходные данные: BX = состояние всех кнопок мыши:
бит 0 = 1;

бит 1 = 1;
бит 2 = 1;
CX = горизонтальная координата маркера
в момент последнего отпускания кнопки;
DX = вертикальная координата маркера в момент
последнего отпускания кнопки.

Примечания. Координаты, возвращаемые в регистрах CX и DX, являются координатами элементов изображения на виртуальном экране мыши, а не физическими координатами на реальном экране. При обращении к этой функции счетчик числа нажатий указанной клавиши сбрасывается в ноль.

8. *Функция 07h* – задание диапазона перемещения по горизонтали. Определяет диапазон перемещения маркера мыши по горизонтали. После того, как диапазон установлен, пользователь не может вывести маркер мыши за его пределы.

Входные данные: AX = 0007H;
CX = минимальная горизонтальная координата
маркера;
DX = максимальная горизонтальная координата
маркера;

Выходные данные: отсутствуют.

Примечание. Координаты, передаваемые в регистрах CX и DX, описывают положение элементов изображения на виртуальном экране мыши, а не физические координаты на реальном экране. Если в момент обращения к функции 07H маркер мыши находится за пределами устанавливаемого диапазона, то драйвер мыши автоматически перемещает его внутрь диапазона. Если значение DX меньше значения CX, то эти параметры меняются местами.

9. *Функция 08h* – задание диапазона перемещения мыши по вертикали. Определяет диапазон перемещения маркера мыши по вертикали. После того, как диапазон установлен, пользователь не может вывести маркер мыши за его пределы.

Входные данные: AX = 0008H;
CX = минимальная вертикальная
координата маркера;
DX = максимальная вертикальная координата
маркера;

Выходные данные: отсутствуют.

Примечание. Координаты, передаваемые в регистрах CX и DX, описывают положение элементов изображения на виртуальном экране мыши, а не физические координаты на реальном экране. Если в момент обращения к функции 08H маркер мыши находится за пределами устанавливаемого диапазона, то драйвер мыши автоматически перемещает его внутрь диапазона. Если значение DX меньше значения CX, то эти параметры меняются местами.

10. *Функция 09h* – описание маркера мыши (в графическом режиме). Описывает внешний вид маркера мыши в графическом режиме, а также битовое поле, корректирующее элементы изображения вокруг маркера мыши.

Входные данные: AX = 0009H;
BX = ширина маркера, начиная с левого края битового поля;
CX = высота маркера, начиная с верхнего края битового поля;
EX = адрес сегмента битового поля;
DX = смещение битового поля.

Выходные данные: отсутствуют.

Примечания. Битовое поле состоит из 64 байтов, из которых первые 32 являются результатом операции AND, а остальные 32 байта результат операции OR с текущими элементами изображения.

11. *Функция 0ah* – описание маркера мыши (в тестовом режиме). Описывает битовую маску, определяющую внешний вид маркера в текстовом режиме.

Входные данные: AX = 000AH;
BX = тип маркера;
BX = 0: программный;
BX = 1: аппаратный;
CX = маска AND (программный маркер) или начальная линия (аппаратный маркер);
DX = маска XOR (программный маркер) или конечная линия (аппаратный маркер).

Выходные данные: отсутствуют.

Примечания. Если выбран программный маркер, то код символа, находящегося под маркером, и байт атрибутов этого символа логически умножаются (AND) на маску, заданную в регистре CX, а затем выполняется операция «исключающее или» (XOR) между результатом умножения и

маской в регистре DX. Для байта атрибутов эти операции выполняются со старшим байтом регистров CX и DX (CH и DH), а для кода символа – с младшим байтом (CL и DL). Аппаратный маркер имеет такую же форму как обычный текстовый курсор. В монохромном режиме значения начальной и конечной линий изменяются в диапазоне от 0 до 13. В цветном режиме значение линий изменяется от 0 до 7.

12. *Функция 0bh* – определение величины перемещения. Определяет расстояние между текущим положением мыши и положением мыши в момент последнего обращения к функции 0BH.

Входные данные: AX = 000BH.

Выходные данные: CX = расстояние от последней
точки по горизонтали (в микки);
DX = расстояние от последней точки
по вертикали (в микки).

Примечания. Эти значения должны интерпретироваться как числа со знаком. Положительные значения указывают на перемещение в нижнюю или правую часть экрана, а отрицательные – в верхнюю или левую часть экрана. Расстояния выражены в микки (1 микки = 1/200 дюйма), а не в элементах изображения.

13. *Функция 0fh* – задание скорости маркера. Устанавливает соотношение между микки (1 микки = 1/200 дюйма) и элементами изображения на экране. Это соотношение определяет чувствительность мыши и скорость перемещения по экрану.

Входные данные: AX = 000FH;
CX = число микки по горизонтали;
DX = число микки по вертикали.

Выходные данные: отсутствуют.

Примечания. Значения регистров CX и DX могут изменяться в диапазоне от 1 до 32767. По умолчанию скорость задается равной 8 микки по горизонтали и 16 микки по вертикали. Таким образом, по горизонтали маркер двигается вдвое быстрее, чем по вертикали. Обращение к функции 00H (сброс драйвера мыши) отменяет любые установленные значения скорости и заменяет их значениями по умолчанию.

14. *Функция 10h* – область исключения. Описывает любую область экрана как область исключения. При входе в область исключения маркер мыши исчезает.

Входные данные: AX = 0010H;

CX = координата X, верхний левый угол области исключения;
DX = координата Y, верхний левый угол области исключения;
SI = координата X, правый нижний угол области исключения;
DI = координата Y, правый нижний угол области исключения.

Выходные данные: отсутствуют.

Примечания. Координаты, передаваемые в регистрах CX, DX, DI и SI описывают положение элементов изображения на виртуальном экране мыши, а не физические координаты на реальном экране. Обращение к функции 00H (сброс драйвера мыши) или к функции 01H (вывод маркера мыши) отменяет координаты области исключения.

15. *Функция 13h* – задание предельной скорости для удвоения скорости маркера. Эта функция задает предельное значение скорости мыши, при которых происходит удвоение скорости. Если скорость перемещения мыши превышает определенный предел, то драйвер мыши удваивает скорость маркера путем удвоения значения соотношения между микки и элементами изображения на экране.

Входные данные: AX = 0013H;
DX = предельная скорость, выраженная в микки на секунду;

Выходные данные: отсутствует.

Примечания. 1 микки – 1/200 дюйма. Чтобы предотвратить удвоение скорости мыши, можно установить более высокий предел. Скорость свыше 5 000 микки в секунду достичь практически невозможно.

16. *Функция 1ah* – задание чувствительности мыши. Определяет соотношение между физическим перемещением и перемещением маркера мыши. Определяет также максимальную скорость, при которой происходит удвоение скорости мыши.

Входные данные: AX = 001AH;
BX = число микки по горизонтали;
CX = число микки по вертикали;
DX = предельная скорость для удвоения скорости Мыши.

Выходные данные: отсутствуют.

Примечания. Значения регистров CX и DX могут изменяться от 1 до 32 767. По умолчанию устанавливается 8 микки по горизонтали и 16 микки

по вертикали. Таким образом, по горизонтали маркер движется вдвое быстрее, чем по вертикали. Чтобы предотвратить удвоение скорости мыши, можно установить более высокий предел. Скорость свыше 5 000 микки в секунду достичь практически невозможно. Обращение к функции 00H (сброс драйвера мыши) отменяет установленные ранее значения скорости и заменяет их значениями по умолчанию.

17. *Функция 1bh* – определение чувствительности мыши. Возвращает параметры, установленные ранее в результате обращения к функциям 1ah,0fh или 13h.

Входные данные: AX = 001BH.

Выходные данные: BX = число микки по горизонтали;
CX = число микки по вертикали;
DX = предельное значение скорости
для удвоения скорости мыши.

18. *Функция 1ch* – задание интенсивности аппаратных прерываний мыши. Определяет частоту считывания аппаратным обеспечением мыши текущего положения мыши и состояния ее кнопок.

Входные данные: AX = 001CH;
BX = интенсивность прерываний;
бит 0: прерывание отсутствует;
бит 1: 30 прерываний в секунду;
бит 2: 50 прерываний в секунду;
бит 3: 100 прерываний в секунду;
бит 4: 200 прерываний в секунду;
бит 5 – 15: не используются.

Выходные данные: отсутствуют.

Примечания. Эта функция может быть использована только для подключенной к порту мыши. Если в регистре BX установлены в единицу несколько битов, то действует только самый младший. Разрешение мыши возрастает с увеличением интенсивности прерываний. Увеличение числа прерываний от мыши снижает скорость выполнения основной программы.

19. *Функция 1fh* – деактивизация драйвера мыши. Переводит в неактивное состояние текущий драйвер мыши и возвращает адрес программы обработки прерывания, которая использовалась для прерывания 33H.

Входные данные: AX = 001FH.

Выходные данные: AX = код ошибки:
AX = FFFFH: ошибка;
AX = 001FH: ошибка;

ES = адрес сегмента
использовавшегося обработчика событий;
BX = смещение использовавшегося обработчика
событий.

Примечания. Обращение к этой функции отключает все установленные ранее активные программы обработки драйверов мыши. Исключением является программа обработки прерывания 33H, но вызывающая программа может записать в этот вектор прерывания первоначальное значение, поскольку соответствующий адрес возвращается в регистрах ES:BX.

20. *Функция 20h* – активизация драйвера мыши. Активизирует драйвер мыши, отключенный ранее функцией 1FH.

Входные данные: AX = 0020H.

Выходные данные: отсутствуют.

21. *Функция 21h* – сброс драйвера мыши. Инициализирует драйвер мыши и запрещает маркер мыши и установленный на данный момент обработчик событий.

Входные данные: AX = 0021H.

Выходные данные: AX = состояние ошибки:
AX = FFFFH: ошибка;
AX = 0021H: без ошибок;
BX = число кнопок мыши.

Примечания. В отличие от функции 00H эта функция не выполняет полного аппаратного сброса устройства.

22. *Функция 24h* – определение типа мыши. Определяет тип установленной мыши и номер версии драйвера мыши.

Входные данные: AX = 0024H.

Выходные данные: BH = целая часть номера версии;
BL = дробная часть номера версии;
CH = тип мыши:
CH = 1: параллельная мышь;
CH = 2: последовательная мышь;
CH = 3: подключенная к порту мышь;
CH = 4: мышь PS/2;
CH = 5: мышь фирмы Хьюлетт Паккард;
CL = номер IRQ;
CL = 0:PS/2;
CL = 2,3,4,5 или 7: номер IRQ в PC.

Примечания. Если номер версии драйвера равен, например, 6.24, то значение 6 возвращается в регистре BH, а 24 в регистре BL.

Таким образом, функции драйвера вызываются через прерывание 33h. Записав номер функции в регистр AX, результат получаем в регистрах AX, BX, CX, DX. Для генерации программного прерывания процессора 8086 можно воспользоваться библиотечной функцией языка Си int86, которая имеет следующий прототип:

int int86(int intr_num, union REGS*inregs, union REGS*outregs);

– функция генерирует программное прерывание микропроцессора 8086, причем номер прерывания определяется аргументом intr_num. Перед выполнением прерывания функция копирует значения регистров из объединения inregs в сами регистры. После возврата из прерывания, функция копирует текущие значения регистров в параметр outregs.

Порядок выполнения работы

Согласно одному из следующих вариантов задания написать и отладить программу на языке Си или Ассемблер, работающую с манипулятором типа «мышь», используя прерывание 33h:

1. рисование прямой линии с помощью мыши. Линия должна соединять две точки, первая точка имеет координаты нажатия левой кнопки мыши, а вторая – правой;
2. рисование кривой линии с помощью мыши. Рисование осуществляется перемещением манипулятора «мышь» при нажатой левой кнопке мыши;
3. рисование прямоугольника с помощью мыши. Координаты левой верхней вершины определяются нажатием левой кнопки мыши, а правой нижней – правой кнопки;
4. вывод координат в процессе перемещения мыши. Координаты должны выводится в определенной пользователем области исключения. Процесс вывода координат мыши включается нажатием левой, а выключается нажатием правой кнопки мыши;
5. определение нажатия любой кнопки мыши и вывод на экран времени нажатия;
6. вывод на экран в место, на которое указывает курсор, текстовой строки по нажатию левой кнопки мыши и возможность изменения её цвета с помощью правой кнопки мыши.

Контрольные вопросы

1. Какие Вы знаете стандарты на способы подключения мыши к компьютеру?
2. Какое прерывание MS-DOS служит для вызова функций драйвера мыши?

3. Опишите назначение функций драйвера мыши, использованных в ходе выполнения работы.

4. Опишите порядок работы с прерыванием MS-DOS для вызова функций драйвера мыши.

Содержание отчета

Отчет должен содержать:

1. титульный лист.
2. тему и цель лабораторной работы.
3. задание на лабораторную работу.
4. описание алгоритма программы (блок-схема или текстовое описание).
5. прокомментированный листинг программы.
6. выводы по результатам работы.

Литература

1. Кулаков, В. Программирование на аппаратном уровне: спец. справочник / В. Кулаков. – 2-е изд. – СПб. : Питер, 2003.
2. Деревянко, А. С. Системное программное обеспечение персональных ЭВМ: учеб. пособие / А. С. Деревянко. – Харьков : ХГПУ, 1994.
3. Нортон, П. Программно-аппаратная организация компьютера IBM PC / П. Нортон; пер. с англ. С. Писарева, Б. Шура. – Киев, 1987.
4. Журден, Р. Справочник программиста на персональном компьютере фирмы IBM / Р. Журден. – Р.-Prentice-Hall Press, 1986.

Лабораторная работа № 5

МОДЕМЫ

Цель работы: изучение работы модемов и их программирование.

Теоретические сведения

Большое число периферийных устройств персонального компьютера подключается к адаптеру асинхронной последовательной связи. Перенос информации между адаптером и внешним устройством организуется по правилам последовательного асинхронного интерфейса RS-232. Наиболее часто адаптер асинхронной последовательной связи используется для организации обмена информацией между компьютерами с использованием телефонных линий связи. В этом случае внешним устройством является специальный аппаратный узел – *модем* (образован из слов МОдулятор – ДЕМодулятор).

Применение модемов позволяет организовать связь с относительно малыми затратами на аппаратную реализацию при практически неограниченной дальности, т. е. позволяет получить глобальный вид связи.

Принципы реализации модемной связи

Под модемной связью понимается такая система передачи данных, в которой приемником и передатчиком служит модем, а в качестве канала передачи выступает коммутируемая телефонная сеть общего пользования (КТСОП). Но модем, как устройство, не является оконечным потребителем или источником информации. Он предназначен только для передачи, а оконечным устройством является компьютер, к которому он подключён. Таким образом, данная система передачи данных (СПД), организованная для связи двух компьютеров (рис. 5.1), представляет собой следующее:

- оконечное оборудование данных в точке А;
- интерфейс между оконечным оборудованием данных и аппаратурой канала данных в точке А;
- аппаратура канала данных в точке А;
- канал передачи между точками А и В;
- аппаратура канала данных в точке В;
- интерфейс между оконечным оборудованием данных и аппаратурой канала данных в точке В;
- оконечное оборудование данных в точке В.

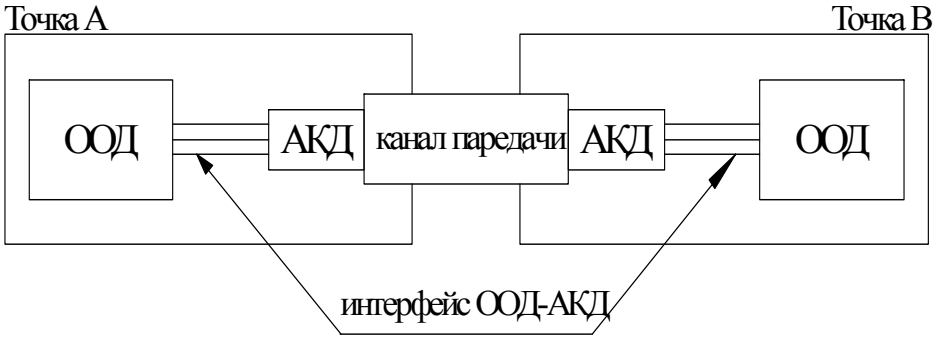

Рис. 5.1. **Система передачи данных компьютер-компьютер через модемную связь**

Оконечное оборудование данных (ООД) – общий термин, используемый для обозначения оконечного прибора пользователя или его части (компьютер, принтер, плоттер и т. д.). ООД может быть как источником информации, так и её получателем, или и тем и другим вместе. Как правило, ООД передает и/ или принимает информацию, используя для этой цели другие устройства – аппаратуру канала данных.

Аппаратура канала данных (АКД) также носит название аппаратуры передачи данных. Её функции состоят в обеспечении возможности передачи информации между двумя или большим числом АКД по каналу передачи. С этой целью аппаратура канала должна обеспечить соединение с ООД с одной стороны, и с каналом передачи – с другой. В качестве АКД обычно выступает модем.

Под интерфейсом понимается программно-аппаратные (в данном случае только аппаратные) средства, обеспечивающие взаимодействие двух систем или процессов в точке их сопряжения. В стандартном исполнении, когда модем является внешним устройством, интерфейсом является стандарт RS-232C (RS – Recommended Standard – рекомендуемый стандарт) разработанный Ассоциацией электронной промышленности США (Electronic Industries Association – EIA). В нашей стране подобный стандарт закреплен ГОСТ 18145-81.

Наличие данных и приёмно-передающего устройства ещё не означает возможность обмена. Для этого необходимы набор правил, управляющих совместной работой, и определение форматов, описывающих язык общения и представление информации. Сводная характеристика передаваемого в интерфейс слова называется *форматом слова*.

Передача информации между источником и приемником возможна тогда, когда они используют одинаковый формат слова. Только в этом случае приемник может обнаружить конец слова. При передаче слов информации реализуется так называемый *старт-стопный метод*. Его суть в

том, что каждое передаваемое слово начинается старт-битом, позволяющим приемнику определить начало передачи слова. Затем передается бит за битом байт информации. Завершение передачи слова отмечается специальными стоп-битами. Электрически логическому нулю в интерфейсе соответствует высокий потенциал +12 В, а логической единице – 0 В. Старт-бит – это всегда единица, стоп-бит – всегда нуль.

Если приемник полагает, что наступило время принимать стоп-биты (логический ноль), но из интерфейса поступает логическая единица, приемник фиксирует ошибку кодирования. Формат слова определяет следующие особенности переноса информации через интерфейс:

- число битов, используемых для кодирования самого переносимого символа;
- наличие или отсутствие контроля по четности;
- способ формирования контрольного бита;
- число стоп-битов.

Линии интерфейса RS-232

Полный стандарт RS-232C включает 25 линий, но на практике используется лишь некоторое их подмножество (табл. 5.1). Обмен сигналами между адаптером компьютера и модемом строится по стандартизованному сценарию, в котором каждый сигнал генерируется сторонами лишь после наступления определенных условий или, говоря образно, «одна сторона ожидает реплики партнера и только потом произносит следующее слово». Такая процедура обмена информацией называется запрос-ответным режимом, или «рукопожатием» (handshaking).

Большинство из приведенных в табл. 1 сигналов как раз и нужны для аппаратной реализации «рукопожатия» между адаптером и модемом. Обмен сигналами между сторонами интерфейса RS-232 выглядит так:

- компьютер после включения питания выставляет сигнал DTR, который постоянно удерживается активным. Если модем включен в электросеть и исправен, он отвечает компьютеру сигналом DSR. Этот сигнал служит подтверждением того, что DTR принят, и информирует компьютер о готовности модема к приему информации;
- если компьютер получил сигнал DSR и хочет передать данные, он выставляет сигнал RTS;
- если модем готов принимать данные, он отвечает сигналом CTS. Он служит для компьютера подтверждением того, что RTS получен модемом и модем готов принять данные от компьютера. С

этого момента адаптер компьютера может бит за битом передавать информацию по линии TXD;

- получив байт данных, модем может сбросить свой сигнал CTS, информируя компьютер о необходимости «притормозить» передачу следующего байта, например из-за переполнения внутреннего буфера;
- программа компьютера, обнаружив сброс CTS, прекращает передачу данных, ожидая повторного появления CTS.

Таблица 5.1

Сигналы интерфейса RS-232

Номер линии	Обозначение	Использование сигнала
1		Экран
От компьютера в модем		
2	TXD	Передача данных в модем
4	RTS	Запрос на передачу (Request To Send): информирует модем о том, что компьютер хочет передать данные в линию
20	DTR	Готовность терминала данных (Data Terminal Ready): информирует модем о том, что компьютер готов к обмену информацией
От модема в компьютер		
3	RXD	Прием данных от модема
5	CTS	Готовность модема (Clear To Send): информирует компьютер о том, что модем готов принимать данные от компьютера и передавать их в линию
6	DSR	Готовность набора данных (Data Set Ready): информирует компьютер о том, что модем готов к обмену данными
7	GND	Логический нуль
8	DCD	Соединение с модемом удаленной станции (Data Carrier Detected): информирует компьютер о том, что модем установил связь с модемом удаленной станции
22	RI	Индикатор вызова (Ring Indicator): информирует компьютер о том, что на телефон, к которому подключен модем, поступил «звонок»

Когда модему необходимо передать данные в компьютер, он (модем) выставляет сигнал на разъеме 8 – DCD. Программа компьютера, принимающая данные, обнаружив этот сигнал, читает приемный регистр, в который сдвиговый регистр «собрал» биты, принятые по линии приема данных RXD. Когда для связи используются только приведенные в табл. 1 данные, компьютер не может попросить модем «повременить» с передачей следующего байта. Как следствие, существует опасность переопределения

помещенного ранее в приемном регистре байта данных вновь «собранным» байтом. Поэтому при приеме информации компьютер должен очень быстро освобождать приемный регистр адаптера. В полном наборе сигналов RS-232 есть линии, которые могут аппаратно «приостановить» модем.

Типы модемов

В настоящее время выпускается огромное количество всевозможных модемов, начиная от простейших, обеспечивающих скорость передачи около 300 бод, до сложных факс – модемных плат, позволяющих вам послать с вашего компьютера факс или звуковое письмо в любую точку мира. На практике широко используются так называемые hayes-совместимые модемы. Эти модемы поддерживают разработанный фирмой Hayes набор AT – команд управления модемами. В настоящее время такие модемы широко используются во всем мире для связи персональных компьютеров через телефонные линии. Аппаратно модемы выполнены либо как отдельная плата, вставляемая в слот на материнской плате компьютера, либо в виде отдельного корпуса с блоком питания, который подключается к последовательному асинхронному порту компьютера. Первый из них называется внутренним модемом, а второй – внешним.

Типичный модем содержит следующие компоненты: специализированный микропроцессор, управляющий работой модема, оперативную память, хранящую значения регистров модема и буферизующие входную/выходную информацию, постоянную память, динамик, позволяющий выполнять звуковой контроль связи, а также другие вспомогательные элементы. Современный модем, скорее всего, дополнительно содержит электрически перепрограммируемую постоянную память, в которой может быть сохранена конфигурация модема даже при выключении питания.

Чтобы модемы могли обмениваться друг с другом информацией, надо, чтобы они использовали одинаковые способы передачи данных по телефонным линиям. Для разработки стандартов передачи данных был создан специальный международный консультативный комитет по телеграфии и телефонии (CCITT) и приняты следующие рекомендации.

- CCITT V.21 – 300 bps. Модем, регламентированный данной рекомендацией, предназначен для передачи данных по выделенным и коммутируемым линиям. Он работает в асинхронном дуплексном режиме. Для передачи и приема данных используется способ частотной модуляции.
- CCITT V.22 – 1 200 bps. Модем, работающий в соответствии с данной рекомендацией, использует асинхронно-синхронный дуплексный режим передачи. Асинхронно-синхронный режим означает, что компьютер

передает модему данные в асинхронном режиме. Модем удаляет из потока данных компьютера стартовые и стоповые биты. И уже в синхронном виде передает их удаленному компьютеру. Для модуляции передаваемого сигнала применяется метод дифференциальной фазовой модуляции.

- CCITT V.22bis – 2 400 bps. Дуплексный модем, со скоростью передачи данных 2 400 bps. При передаче со скоростью 2 400 bps используется метод квадратурной модуляции, а при скорости 1 200 – метод дифференциальной фазовой модуляции. На скорости 1 200 bps модем CCITT V.22bis совместим с CCITT V.22.
- CCITT V.23 – 600/ 1 200 bps . Асинхронный модем, использующий метод частотной модуляции. Модем может работать в дуплексном режиме со скоростью передачи данных по прямому каналу 600/ 1 200 bps, а по обратной - только 75 bps. Этот стандарт не совместим с CCITT V.21, V.22, V.22bis.
- Bell 103 — 300 bps, Bell 212A — 1 200 bps. Bell – это американский стандарт, не совместимый со стандартами CCITT.

Протоколы обмена данными. Протоколы коррекции ошибок нижнего уровня

При передаче данных по зашумленным телефонным линиям всегда существует вероятность, и она достаточно велика, что данные, переданные одним модемом (компьютером), будут приняты другим модемом в искаженном виде. Для того чтобы пользователь имел гарантии, что его данные переданы без ошибок, используются протоколы коррекции ошибок.

Общая форма передачи данных по протоколам с коррекцией ошибок следующая: данные передаются отдельными блоками (пакетами) по 16 – 20 000 байт, в зависимости от качества связи. Каждый блок снабжается заголовком, в котором указана проверочная информация, например, контрольная сумма блока. Принимающий компьютер самостоятельно подсчитывает контрольную сумму каждого блока и сравнивает ее с контрольной суммой из заголовка блока. Если эти две контрольные суммы совпали, принимающая программа считает, что блок передан без ошибок. В противном случае принимающий компьютер передает передающему запрос на повторную передачу этого блока.

Протоколы коррекции ошибок могут быть реализованы как на аппаратном уровне, так и на программном. Аппаратный уровень реализации более эффективен. Например, быстродействие аппаратной реализации протокола MNP примерно на 30 % выше, чем программной.

- *MNP (Microcom Network Protocols)* – серия наиболее распространенных аппаратных протоколов, впервые реализованная на модемах фир-

мы Microcom. Эти протоколы обеспечивают автоматическую коррекцию ошибок и компрессию передаваемых данных.

- *MNP1*. Протокол коррекции ошибок, использующий асинхронный полудуплексный метод передачи данных. Это самый простой из протоколов MNP.
- *MNP2*. Протокол коррекции ошибок, использующий асинхронный дуплексный метод передачи данных.
- *MNP3*. Протокол коррекции ошибок, использующий синхронный дуплексный метод передачи данных между модемами (интерфейс модем - компьютер остается асинхронным).
- *MNP4*. Протокол, использующий синхронный метод передачи, обеспечивает оптимизацию фазы данных, которая несколько улучшает неэффективные протоколы MNP2 и MNP3. Кроме того, при изменении числа ошибок на линии соответственно меняется и размер блоков передаваемых данных. При увеличении числа ошибок размер блоков уменьшается, увеличивая вероятность успешного прохождения отдельных блоков. Эффективность этого метода составляет около 20 % по сравнению с простой передачей данных.
- *MNP5*. Дополнительно к методам MNP4, MNP5 использует простой метод сжатия передаваемой информации. Символы, часто встречающиеся в передаваемом блоке, кодируются цепочками битов меньшей длины, чем редко встречающиеся символы. Дополнительно кодируются длинные цепочки одинаковых символов. Обычно при этом текстовые файлы сжимаются до 35 % своей исходной длины. Вместе с 20 % улучшениями в MNP4 это дает повышение эффективности до 50 %.
- *MNP6*. Дополнительно к методам протокола MNP5 автоматически переключается между дуплексным и полудуплексным методами передачи в зависимости от типа информации. Протокол MNP6 также обеспечивает совместимость с протоколом V.29.
- *MNP7*. По сравнению с ранними протоколами использует более эффективный метод сжатия данных.
- *MNP9*. Использует протокол V.32 и соответствующий метод работы, обеспечивающий совместимость с низкоскоростными модемами.
- *MNP10*. Предназначен для обеспечения связи на сильно зашумленных линиях, таких, как линии сотовой связи, междугородные линии. Это достигается при помощи следующих методов: многократного повторения попытки установить связь; изменения размера пакетов в соответствии с изменением уровня помех на линии; динамического изменения ско-

рости передачи в соответствии с уровнем помех линии. При использовании модемов с аппаратной поддержкой протокола MNP, как правило, можно установить скорость работы последовательного порта вдвое выше скорости модема.

Все протоколы MNP совместимы между собой снизу вверх. При установлении связи происходит установка наивысшего возможного уровня MNP-протокола. Если же один из связывающихся модемов не поддерживает протокол MNP, то MNP-модем работает без MNP-протокола.

- *V.42*. Протокол с коррекцией ошибок и преобразованием, асинхронный-синхронный. Использует метод компрессии, при котором определяется частота появления отдельных символьных строк и происходит их замена на последовательности символов меньшей длины (на токены). Этот метод носит название Lempel-Ziv. Данный метод компрессии обеспечивает 50 % сжатие текстовых файлов. Вместе с 20 % выигрышем от синхронного преобразования это увеличивает эффективность на 60 %.

Протоколы передачи файлов

В отличие от протоколов нижнего уровня данные протоколы позволяют организовать прием и передачу файлов.

- *ASCII* – этот протокол работает без коррекции ошибок. В результате при передаче файлов по телефонным линиям из-за шума принятый файл может сильно отличаться от передаваемого.
- *Xmodem*. Наиболее распространены три разновидности протокола Xmodem: оригинальный протокол Xmodem, Xmodem с CRC, 1K Xmodem.

Оригинальный протокол Xmodem разработал Вард Кристенсен в 1977 г. При передаче файлов с помощью протоколов Xmodem формат данных должен быть следующим: 8-битовые данные, один стоповый бит и отсутствие проверки на четность. Для передачи используется полудуплексный метод, т. е. данные могут передаваться в каждый момент времени только в одном направлении.

Протокол Xmodem Checksum передает данные пакетами по 128 байт. Вместе с пакетом передается его контрольная сумма. При получении пакета контрольная сумма вычисляется снова и сравнивается с суммой, вычисленной на передающей машине. Если эти две суммы совпадают, считается, что пакет передан без ошибок. Этот метод обеспечивает достаточно хорошую защиту от ошибок. Только один из 256 пакетов может содержать ошибки, хотя контрольная сумма правильная.

Xmodem с CRC. Более защищенным от ошибок является протокол Xmodem CRC (Cyclic Redundancy Check). Xmodem CRC – протокол с проверкой циклическим избыточным кодом. В нем 8-битовая контрольная сумма заменена на 16-битовый циклический избыточный код. Этот протокол гарантирует вероятность обнаружения ошибок, равную 99,9984 %. Только один из 700 биллионов плохих пакетов будет иметь правильный CRC-код. Протокол Xmodem CRC также передает данные пакетами по 128 байт.

1K Xmodem. Если передача идет без ошибок, протокол 1K Xmodem увеличивает размер пакета с 128 до 1 024 байт. При увеличении числа ошибок размер пакета снова уменьшается. Такое изменение длины пакета позволяет увеличить скорость передачи файлов. В остальном, протокол 1K Xmodem совпадает с протоколом Xmodem CRC.

- *Ymodem.* Протокол Ymodem разработал Чак Форсберг в 1984 – 1985 гг. Протокол Ymodem похож на протокол 1K Xmodem, но имеет одно отличие: протокол Ymodem может передавать или принимать за один заход несколько файлов. Существует модификация протокола Ymodem – Ymodem G. Протокол Ymodem G предназначен для использования с модемами, автоматически осуществляющими коррекцию ошибок на аппаратном уровне. Например, MNP-модемы с аппаратной реализацией MNP. В этом протоколе упрощена защита от ошибок, т. к. ее выполняет сам модем. Другой особенностью протокола Ymodem является то, что вместе с файлом передаются все его атрибуты. В результате как минимум имя файла и дата останутся неизменными.
- *Zmodem* – это достаточно быстрый протокол передачи данных, использующий окна. Zmodem осуществляет передачу данных пакетами по несколько штук в окне. При этом принимающий данные компьютер не передает сигнал подтверждения или сигнал переспроса неправильного пакета, пока не получит все пакеты в окне. Протокол Zmodem, так же как и протокол 1K Xmodem, может изменять длину пакета (блока) от 64 до 1 024 байт в зависимости от качества линии. Кроме того, протокол обладает следующей полезной особенностью: если при передаче файла произошел сбой на линии и весь файл не был передан, то в следующий раз при передаче этого же файла он автоматически начнет передаваться с того места, где произошел обрыв связи. Таким образом, очень большие файлы можно передавать по частям. Из всех протоколов верхнего уровня, описанных выше, этот протокол самый быстрый и удобный.

- *Bimodem*. Особенностью протокола Bimodem является возможность одновременной пересылки двух файлов в разных направлениях. Кроме того, одновременно с передачей файлов можно общаться с оператором удаленного компьютера при помощи клавиатуры.
- *Kermit.* Широко известны две разновидности протокола Kermit – стандартный и Super Kermit. Этот протокол был разработан в Колумбийском университете в 1981 г. для связи между различными типами компьютеров, включая большие компьютеры, мини-компьютеры и персональные компьютеры. В отличие от протоколов Xmodem и Ymodem он использует для передачи данных пакеты переменной длины с максимальным размером 94 байт. Так же как и Ymodem, протокол Kermit может передавать или принимать несколько файлов за один сеанс. Протокол Super Kermit предназначен специально для использования в сетях типа Telenet или Tymnet. Эти сети имеют очень большие задержки при передаче данных. Так что если ждать подтверждения для каждого пакета, это может привести к резкому снижению скорости обмена. В протоколе Super Kermit эта проблема решается следующим способом. Несколько пакетов передается за один раз (в одном окне). Все действия по контролю над ошибками остаются, за исключением того, что принимающий данные компьютер не передает сигнал подтверждения или сигнал на переспрос неправильного пакета, пока не получит все пакеты в окне. В результате использования такого механизма происходит резкое сокращение времени задержки. Окно может содержать от одного до 31 пакета. В дополнение Kermit использует также предварительную компрессию данных для увеличения эффективной скорости обмена данными.

Программирование модемов

После выпуска американской фирмой Hayes модемов серии Smartmodem, система команд, использованная в ней, стала неким стандартом, которого придерживаются остальные фирмы – разработчики модемов. Система команд, применяемая в этих модемах, носит название hayes-команд, или AT-команд.

Со времени выпуска первых AT-совместимых модемов набор их команд несколько расширился, но все основные команды остались без изменения. Все команды, передаваемые компьютером модему, надо начинать префиксом AT (ATtention – внимание) и заканчивать символом возврата каретки (<CR>). Только команда A/ и Escape-последовательность «+++» не требуют для себя префикса AT.

После префикса AT могут идти одна или сразу несколько команд. Для ясности эти команды могут быть отделены друг от друга символами

пробела, тире, скобками. В большинстве случаев команды могут быть написаны как заглавными, так и строчными буквами.

При передаче модему команд они сначала заносятся во внутренний буфер, который, как правило, имеет размер 40 символов. Команды, записанные в буфер модема, исполняются после поступления символа возврата каретки. Вследствие ограниченности размера буфера не следует передавать модему слишком длинные команды (больше размера буфера). Длинные команды можно разбивать на части и передавать в несколько заходов. При этом каждая часть должна начинаться префиксом AT и заканчиваться символом возврата каретки. При ошибочном наборе команды ее можно исправить, используя клавишу BackSpace.

После выполнения каждой команды модем посылает обратно компьютеру ответ в виде числа или слова. Этот ответ означает, выполнена ли команда (чаще всего Ок) или произошла ошибка (в зависимости от настроек).

Внешний модем на лицевой панели содержит восемь световых индикаторов. Хотя их расположение на различных моделях может меняться, их обозначения являются стандартными:

MR Modem Ready – модем готов к обмену данными. Если этот индикатор не горит, то надо проверить линию питания модема.

TR Terminal Ready – компьютер готов к обмену данными с модемом. Этот индикатор горит, когда модем получил от компьютера сигнал DTR.

CD Carrier Detect – индикатор зажигается, когда модем обнаружил несущую частоту на линии. Индикатор должен гореть на протяжении всего сеанса связи и гаснуть, когда один из модемов освободит линию.

SD Send Data – индикатор мигает, когда модем получает данные от компьютера.

RD Receave Data – индикатор мигает, когда модем передает данные к компьютеру

HS High Speed – модем работает на максимально возможной для него скорости.

AA Auto Answer – модем находится в режиме автоответа. Т. е. модем автоматически будет отвечать на приходящие звонки. Когда модем обнаружит звонок на телефонной линии, этот индикатор замигает.

OH Off-Hook – этот индикатор горит, когда модем снял трубку (занимает линию).

Основные команды модема

AT – начало (префикс) командной строки. После получения этой команды модем автоматически подстраивает скорость передачи и формат данных к параметрам компьютера.

A – автоответ. Если режим автоматического ответа выключен (S0 = 0), команда используется для ответа на звонок от удаленного модема. Команда заставляет модем снять трубку (подключиться к линии) и установить связь с удаленным модемом.

A/ – модем повторяет последнюю введенную команду. Команда передается на модем без префикса AT и исполняется модемом немедленно, не ожидая прихода символа возврата каретки. Если вы передадите модему строку AT A/ <CR>, то модем укажет на ошибку и вернет слово ERROR.

Bn – команда производит выбор стандарта, согласно которому будет происходить обмен данными между модемами. При скорости передачи 300 бит/ с происходит выбор между стандартами BELL 103 и CCITT V.21, при скорости 1 200 bps – между BELL 212A и CCITT V.22bis. При скорости 2400 bps эта команда игнорируется и используется стандарт CCITT V.22. Если n = 0, устанавливаются стандарты CCITT V.21/V.22, а если n = 1 – стандарты BELL 103/212A.

Ds – команда используется для набора номера. После получения этой команды модем начинает набор номера и при установлении связи переходит в режим передачи данных. Команда состоит из префикса AT, символа D и телефонного номера, в состав которого могут входить следующие управляющие модификаторы: P или T. Эти модификаторы производят выбор между импульсной и тоновой системой набора (в нашей стране используется импульсная система).

, – символ запятой вызывает паузу при наборе номера. Длительность паузы определяется содержимым регистра S8.

; – символ точки с запятой, если он находится в конце командной строки, переводит модем после набора номера в командный режим.

@ – модем ожидает пятисекундной тишины на линии в течение заданного промежутка времени. Промежуток времени, в течение которого модем ожидает тишины, задается в регистре S7. Если в течение этого времени паузы тишины не было, модем отключается и отвечает NO ANSWER.

! – если знак ! стоит перед знаками последовательности набора, модем переходит в состояние ON HOOK (кладет трубку) на 1/2 с, а затем снова переходит в состояние OFF HOOK (снимает трубку).

S – модем набирает телефонный номер, записанный в его памяти. Эта команда выполняется только для модемов, имеющих встроенную энергонезависимую память и возможность записи в нее номеров телефонов.

R – после набора номера переводит модем в режим автоответа. Этот модификатор должен находиться в конце набираемого номера.

W – перед дальнейшим набором телефонного номера модем ожидает длинный гудок из линии. Причем время ожидания гудка содержится в регистре S7. Если в отведенное время гудок не появился, модем прекращает набор номера и возвращает сообщение NO DIALTONE. Этот параметр может быть полезен при наборе междугородних номеров.

En – управление эхо-выводом команд, передаваемых модему. После команды E1 модем возвращает каждый знак, передаваемый ему, обратно компьютеру, что позволяет узнать, как работает связь модема и компьютера. Команда E0 запрещает эхо-вывод.

Fn – переключение между дуплексным/ полудуплексным режимами. При n = 0 переход в полудуплексный режим, а при n = 1 – в дуплексный.

Hn – эта команда используется для управления телефонной линией. Если n = 0, то происходит отключение модема от линии, если n = 1, модем подключается к линии.

In – выдает идентификационный код модема и контрольную сумму содержимого памяти модема. Если n = 0, модем сообщает свой идентификационный код; n = 1 – модем проводит подсчет контрольной суммы EPROM и передает ее компьютеру; n = 2 – модем проверяет состояние внутренней памяти ROM и возвращает сообщение OK или CHECKSUM ERROR (ошибка контрольной суммы). При n = 3 выдается состояние модема.

Ln – установка громкости сигнала внутреннего динамика: n = 0,1 соответствует низкой громкости; n = 2 – средней и n = 3 – максимальной.

Mn – управление внутренним динамиком. При n = 0 динамик выключен. При n = 1 динамик включен только во время набора номера и выключен после обнаружения несущей. При n = 2 динамик включен все время. При n = 3 динамик включается после набора последней цифры номера и выключается после обнаружения несущей отвечающего модема.

Qn – управление ответом модема на АТ-команды. При n = 0 ответ разрешен, при n = 1 ответ запрещен. Независимо от состояния Q0 или Q1 модем всегда сообщает содержание S-регистров, свой идентификационный код, контрольную сумму памяти и результаты теста.

On – команда переводит модем из командного режима в режим передачи данных. При этом модем отвечает CONNECT (табл. 5.2). Команда O и O0 переводят модем в режим передачи данных без инициирования последовательности сигналов проверки линии связи. Команда O1 переводит модем в режим передачи данных и заставляет модем передать последовательности сигналов проверки линии связи, т. е. производить повторное квитирование с удаленным модемом.

Sr? – чтение содержимого регистра модема, имеющего номер r. Sr = n – запись в регистр модема с номером r числа n. Число n может иметь значения от 0 до 255. Все команды модифицируют содержимое одного или более S-регистров. Некоторые S-регистры содержат временные параметры, которые можно поменять только командой S.

Vn – производит выбор вида ответа модема на АТ-команды. При n = 0 ответ происходит цифровым кодом, а при n = 1 модем отвечает в символьном виде на английском языке. Использование цифровой формы ответа облегчает обработку результатов выполнения команды при написании собственных программ управления модемом.

Таблица 5.2

Стандартный набор ответов модема

0	OK	Команда выполнена без ошибок
1	CONNECT	Установлена связь с удаленным модемом
2	RING	Поступил входящий звонок
3	NO CARRIER	Нет соединения с удаленным модемом
4	ERROR	Команда выполнена с ошибкой
5	CONNECT 1 200	Установлена связь с удаленным модемом на 1200 бод
6	NO DIAL TONE	Нет гудка в линии
7	BUSY	Вызываемый номер занят
8	NO ANSWER	Вызываемый номер не отвечает
9	CONNECT 0 600	Установлена связь с удаленным модемом на 600 бод
10	CONNECT 2 400	Установлена связь с удаленным модемом на 2 400 бод
11	CONNECT 4 800	Установлена связь с удаленным модемом на 4 800 бод
12	CONNECT 9 600	Установлена связь с удаленным модемом на 9 600 бод

Xn – определяет набор сообщений модема, управляет определением сигнала «занято» и наличием гудков на линии.

X0 – сообщение модема об установлении связи приводится в короткой форме – CONNECT – при всех скоростях. Номер набирается модемом после паузы, вне зависимости от присутствия гудка на линии. Состояние «занято» не определяется.

X1 – сообщение модема об установлении связи приводится в полной форме. Номер набирается модемом после паузы, вне зависимости от присутствия гудка на линии. Состояние «занято» не определяется.

X2 – сообщение модема об установлении связи приводится в полной форме. Номер набирается только при наличии гудка на линии. Состояние «занято» не определяется.

X3 – сообщение модема об установлении связи приводится в полной форме. Номер набирается модемом после паузы, вне зависимости от присутствия гудка на линии. Состояние «занято» определяется.

X4 – сообщение модема об установлении связи приводится в полной форме. Номер набирается модемом после паузы при наличии гудка на линии. Состояние «занято» определяется.

Yn – способ отключения модема от линии. Существуют два способа отключения модема от линии: стандартный, когда модем получает неактивный сигнал DTR от компьютера, и принудительный, когда модем получает от удаленного модема сигнал перерыва BREAK. Команда ATH0 направляет удаленному модему сигнал прерывания BREAK, который длится 4с. При n = 0 модем отключается стандартно, при n = 1 модем отключается после получения из линии сигнала BREAK.

Z – сбрасывает конфигурацию модема. При этом во все регистры загружаются значения, принятые по умолчанию. Значения регистров, принятые по умолчанию берутся из энергонезависимой памяти модема или, если модем такой памяти не имеет, из постоянной памяти или определяется исходя из переключателей на плате модема.

+++ – Escape-последовательность, используемая для перехода в командный режим работы модема. Благодаря этой команде можно перейти из режима передачи данных модемом в командный режим работы без разрыва связи. Модем требует тишины перед и после направления этой Escape-последовательности. Величина этого промежутка тишины определена в регистре S12.

&Cn – данная команда управляет сигналом DCD порта RS-232-C. При n = 0 сигнал DCD всегда активен, а при n = 1 сигнал DCD устанавливается только тогда, когда модем обнаруживает несущую частоту от удаленного модема.

&Dn – управление сигналом DTR. При n = 0 модем игнорирует DTR; n = 1 – при потере сигнала DTR модем переходит в командный режим работы; n = 2 – при потере сигнала DTR модем прекращает связь, отключается от линии, отключает режим автоответа и переходит в командный режим работы; n = 3 – при потере сигнала DTR автоматически сбрасывается конфигурация модема, как при выполнении команды ATZ. Модем обнаруживает потерю сигнала DTR, если сигнал DTR отсутствует дольше времени, определенного в регистре модема S25.

&F – модем устанавливает конфигурацию, записанную в постоянную память.

&Gn – включение/ выключение защитной частоты. n = 0 – защитная частота выключена; n = 1 – модем генерирует защитную частоту 550 Hz; n = 2 – модем генерирует защитную частоту 1800 Hz. Использование данной команды зависит от особенностей телефонной линии.

&Ln – вид линии связи. При n = 0 передача по обычным (коммутируемым) линиям связи; n = 1 – передача по выделенным каналам.

&Mn – установка асинхронно/ синхронного режима работы. При n = 0 устанавливается асинхронный режим; при n = 1, 2, 3 – синхронный режим.

&Pn – установка импульсного коэффициента набора номера в соответствии с различными стандартами. При n = 0 – коэффициент заполнения замыкание/ интервал 39/ 61 (Америка); при n = 1 – 33/ 67 (Англия).

&Rn – управление сигналом CTS: n = 0 – сигнал переходит в активное состояние после получения сигнала RTS. Данные, передаваемые модему до поступления сигнала RTS, игнорируются. Если n = 1 модем игнорирует RTS.

&SN – управление сигналом DSR порта RS-232-C. При n = 0 сигнал DSR активен всегда, а при n = 1 сигнал DSR активизируется только после окончания этапа установления связи между модемами.

&Tn – тестирование модема. От n зависит вид теста.

С помощью команды &T0 можно прервать выполнение теста модема в любой момент. Если модем выполняет локальный аналоговый тест или удаленный цифровой тест, то перед передачей команды &T0 надо с помощью Escape-последовательности перевести модем в командный режим.

По команде *&T1* модем начинает выполнять локальный аналоговый тест. Продолжительность теста определяется регистром S18. В ходе локального аналогового теста проверяется и модем и компьютер.

По команде *&T3* модем выполняет локальный цифровой тест. Этот тест используется для проверки линии связи и удаленного модема. Во время локального цифрового теста модем направляет поступающие ему данные обратно на удаленный компьютер. Для выполнения теста необходимо соединиться с удаленным модемом, затем переключить модем в командный режим и выполнить команду &T3. Оператор удаленного модема должен передать несколько проверочных сообщений. Поступив на тестируемый модем, они будут отправлены обратно удаленному модему. Если принятое удаленным модемом сообщение эквивалентно переданному, значит, линия и удаленный модем исправны.

Команда *&T4* дает согласие на начало удаленного цифрового теста, который запрашивает удаленный модем.

Команда *&T5* не дает согласия удаленному модему на начало удаленного цифрового теста.

Команда *&T6* вызывает выполнение удаленного цифрового теста. При этом происходит проверка локального компьютера, локального модема удаленного модема и линии связи.

По команде *&T7* модем выполняет удаленный цифровой тест с самодиагностикой. Модем сам генерирует тестовые сообщения и подсчитывает число ошибок.

По команде *&T8* – локальный аналоговый тест с самодиагностикой. При этом модем сам генерирует тестовые сообщения и подсчитывает число ошибок.

&V – модем показывает свою текущую конфигурацию и телефонные номера, записанные в энергонезависимой памяти.

&W – модем записывает свою текущую конфигурацию в энергонезависимую память. При сбросе модема будет загружена именно эта конфигурация.

&Zn – используется для записи телефонного номера в энергонезависимую память модема. Количество телефонов зависит от модели модема.

Примечание. В связи с отсутствием строгой стандартизации команд практически каждая из фирм-производителей добавляет свои собственные команды к существующим At-командам или может не обрабатывать отдельные из них.

Регистры модемов

Hayes-совместимые модемы имеют набор регистров, определяющих различные характеристики модема. Содержимое большинства этих регистров можно считывать и изменять программным способом. Для чтения и записи регистров модема можно использовать соответственно AT-команды ATSr? и ATSr = n, где r – номер регистра, а n – число, которое в него записывается.

Далее приводится описание 28 регистров модема, диапазон возможных значений и значение, записываемое в регистр по умолчанию.

S0 (0..255) – регистр управляет режимом автоответа. Регистр задает количество звонков, которое модем ждет до ответа на вызов. Если S0 = 0, то режим автоответа выключен. Когда режим автоответа выключен и приходит звонок, чтобы снять трубку, надо специально передать модему команду ATA. Содержимое регистра сохраняется в энергонезависимой памяти (если, конечно, она есть у вашего модема).

S1 – счетчик сигналов звонка. Значение регистра увеличивается каждый раз, когда модему поступает сигнал звонка из телефонной линии. По истечении восьми секунд с момента последнего звонка содержимое регистра сбрасывается. Значение регистра не сохраняется в энергонезависимой памяти.

S2m (0..255) – данный регистр содержит ASCII-код Escape-символа, используемого в последовательности перехода в командный режим («+++»). Обычно он имеет значение 43, что соответствует ASCII символу «+». Можно переопределить Escape-символ, записав в этот регистр ASCII-код другого символа. В случае если значение регистра S2 больше чем 127, происходит блокировка последовательности возврата. При этом будет не возможно переключиться из режима передачи данных в командный режим без потери связи с удаленным модемом. Содержимое регистра не сохраняется в энергонезависимой памяти.

S3 (0..127) – регистр содержит ASCII-код символа возврата каретки – <CR>. По умолчанию регистр содержит ASCII-код 13 (Ctrl-M). Можно переопределить этот символ, записав в регистр новое значение. Содержимое регистра не сохраняется в энергонезависимой памяти. Это гарантирует, что после выключения питания можно снова использовать символ с ASCII-кодом, равным 13, для ввода команд.

S4 (0..127) – регистр содержит ASCII-код символа перевода строки - <LF>. По умолчанию регистр содержит ASCII-код 10 (Ctrl-J). Можно переопределить этот символ, записав в регистр другое значение. При задании S4 = 0 символ <LF> не используется. Содержимое регистра не сохраняется в энергонезависимой памяти.

S5 (0..127) – регистр содержит ASCII-код символа Backspace (возврат на один символ назад). По умолчанию регистр содержит ASCII-код 8 (Ctrl-H). Можно переопределить этот символ, записав в регистр новое значение. Если записать в регистр ASCII-код от 31 до 127, то нельзя будет использовать символ с этим кодом в командном режиме. Содержимое регистра не сохраняется в энергонезависимой памяти.

S6 (2..255) – определяет время в секундах, в течение которого при снятии трубки на линии должен появиться гудок . По умолчанию регистр содержит значение два. В энергонезависимой памяти регистр не сохраняется.

S7 (1..255) – определяет время в секундах после набора номера, в течение которого модем должен выполнить соединение (обнаружить несущую частоту от удаленного модема). По умолчанию регистр содержит значение 30. Если в течение этого времени модем установит связь, он выдает сообщение CONNECT согласно команде ATXn. Если связь не будет установлена, модем отвечает NO CARRIER. В энергонезависимой памяти регистр не сохраняется.

S8 (0..255) – время задержки при наборе номера (в секундах), которая происходит по модификатору «,» команды ATD. По умолчанию время задержки две секунды. В энергонезависимой памяти регистр не сохраняется.

S9 (1..255) определяет время, в течение которого для установки связи должна приниматься несущая частота от удаленного модема. Если несущая принималась в течение этого времени, модем передает компьютеру сигнал DCD. Содержимое регистра задает время в десятых долях секунды. По умолчанию для установки DCD модем должен принимать несущую 0,6 с, т. е. регистр содержит число 6. В энергонезависимой памяти значение регистра не сохраняется.

S10 (1..255) определяет промежуток времени, в течение которого может отсутствовать несущая от удаленного модема и при этом не произойдет разрыв связи. Содержимое регистра задает время в десятых долях секунды. По умолчанию несущая частота может отсутствовать 0,7 с, т. е. регистр содержит число 7. В энергонезависимой памяти значение регистра не сохраняется. Значение регистра S10 должно быть больше значения регистра S9, иначе связь будет невозможно установить.

S11 (50..255) используется при наборе номера по тоновой системе. Регистр определяет длительность (в миллисекундах) передачи одной цифры номера и промежутка между ними. Значение регистра не оказывает влияния на набор номера в импульсном режиме. В импульсном режиме модем набирает номер со скоростью 10 импульсов в секунду.

S12 (20..255) – регистр определяет задержку, которую необходимо выдержать при передаче модему Escape-последовательности "+++" для перевода модема из режима передачи данных в командный режим. Регистр задает временной промежуток в 0,02 сотых секунды. По умолчанию регистр содержит 50.

S13 – регистр не используется.

Далее рассматриваются битовые регистры модемов. Эти регистры можно использовать для определения текущего состояния модема. Все они сохраняются в энергонезависимой памяти модема, если она установлена в модеме.

S14 – состояние модема:

- 0 бит – не используется;
- 1 бит – эхо-вывод. При = 0 эхо-вывод не выполняется, при = 1 выполняется;
- 2 бит – управление ответом модема. При = 0 вывод разрешен, при = 1 – запрещен;
- 3 бит – управление формой ответа модема. Если данный бит равен нулю, ответ выполняется в сокращенной (цифровой) форме. В противном случае модем отвечает словами на английском языке;
- 4 бит – не используется;

- 5 бит – система набора номера. При = 0 модем использует тоновую систему, а при = 1 – импульсную;
- 6 бит – не используется;
- 7 бит – данный бит отражает текущее состояние модема. При = 0 модем находится в состоянии ответа, а при = 1 – в состоянии вызова другого модема.

S15 – не используется.

S16 – параметры тестирования модема.

S17 – не используется.

S18 (0..255) – данный регистр задает длительность теста модема в секундах.

S19 – не используется.

S20 – не используется.

S21 – регистр имеет различные форматы для модемов различных фирм-изготовителей:

- 0 бит – тип используемого для телефонной линии разъема;
- 1 бит – не используется;
- 2 бит – управление сигналами RTS/CTS;
- 3 – 4 бит – управление сигналом DTR;
- 5 бит – управление сигналом DCD;
- 6 бит –- управление сигналом DSR;
- 7 бит – управление режимом разрыва связи по тайм-ауту (по истечении лимита времени при потере несущей).
- *S22* – регистр управляет выбором набора ответов модема и динамиком:
- 1 бит – эти биты устанавливают уровень громкости на встроенном динамике модема: 00 – не используется; 01 – низкий уровень звука; 10 – средний уровень звука; 11 – высокий уровень звука;
- 2 – 3 бит – управление динамиком: 00 – динамик отключен все время; 01 – динамик включен до момента определения несущей; 10 – динамик включен все время; 11 – динамик включается после набора номера до определения несущей;
- 4 – 5 – 6 бит – данные биты определяют набор ответов, используемых модемом: 000 – набор соответствует X0; 100 – набор соответствует X1; 101 – набор соответствует X2; 110 – набор соответствует X3; 111 – набор соответствует X4;
- 7 бит – бит управляет скоростью набора номера при тональной системе вызова: 0 – американский стандарт; 1 – английский (европейский) стандарт.

S23 – регистр определяет различные параметры модема:

- 0 бит – управление удаленным тестированием модема; = 0 – тестирование запрещено; = 1 тестирование разрешено;
- 1 – 2 бит – отражает скорость передачи данных модемом: 00 – скорость 0..300 бит/сек; 01 – скорость 600 бит/сек; 10 – скорость 1 200 бит/сек; 11 – скорость 2 400 бит/сек;
- 3 бит – не используется;
- 4 – 5 бит – управление контролем четности передаваемых данных: 00 – проверка на четность; 01 – бит четности всегда равен 0; 10 – проверка на нечетность; 11 – бит четности всегда равен 1;
- 6 – 7 бит – устанавливает частоту защитного сигнала: 00 – защитный сигнал не генерируется; 01 – защитный сигнал 550 герц; 10 – защитный сигнал 1 800 герц; 11 – не используется.

S24 – не используется.

S25 – регистр задает время задержки сигнала DTR в сотых долях секунды. Так, если регистр содержит число 5, то время задержки равно 5 * 0,001 = 0,005 с.

S26 – регистр задает время между сигналами RTS и CTS в сотых долях секунды. Зависит от команды &R0.

S27 – регистр задает различные параметры режима передачи данных модемом:

- 1 – 2 бит – управляет режимом передачи данных: 00 – асинхронный; 01 – синхронный; 10 – синхронный с набором номера из памяти; 11 – синхронный с набором номера АТ-командой;
- 3 бит – = 0 при передаче по телефонной линии, = 1 при передаче по выделенному каналу;
- бит – не используется;
- 4 – 5 бит – зарезервирован;
- 6 бит – выбор протокола обмена: =0 – CCITT; = 1 – Bell;
- 7 бит – не используется.

Основные принципы программирования модемов

Доступ к модему происходит через последовательный асинхронный порт. При этом для передачи модему команд их необходимо просто записать в регистр данных COM-порта, на котором находится модем. Ответ от модема также поступает через последовательный порт. Передавая модему команды, его можно проинициализировать, перевести в режим автоответа или заставить набрать номер.

Когда модем наберет номер удаленного абонента или когда модему в режиме автоответа придет вызов, он попытается установить связь с уда-

ленным модемом. После установления связи модем передает компьютеру через COM-порт специальное сообщение и переключится из командного режима в режим передачи данных. После этого данные, передаваемые модему, перестают восприниматься им как команды и сразу передаются по телефонной линии на удаленный модем.

Итак, после установления связи с удаленным модемом, коммуникационная программа может начинать обмен данными. Обмен данными так же, как и передача команд, осуществляется через COM-порт. Затем при помощи специальной Escape-последовательности можно переключить модем из режима передачи данных обратно в командный режим и положить трубку, разорвав связь с удаленным модемом.

Последовательность действий для установления связи

1. Инициализация COM-порта. Произвести инициализацию COM-порта, к которому подключен модем. Для этого запрограммировать регистры микросхемы UART, задавая формат данных и скорость обмена. Модем будет проводить соединение с удаленным модемом как раз на этой скорости. Чем скорость выше, тем быстрее будет происходить обмен данными с удаленным модемом. Однако при увеличении скорости на плохих телефонных линиях сильно возрастает количество ошибок.

2. Инициализация модема. Передавая модему AT-команды через COM-порт, произвести его инициализацию. При помощи AT-команд можно установить различные режимы работы модема – выбрать протокол обмена, установить набор диагностических сообщений модема и т. д.

3. Соединение с удаленным модемом. Передать модему команду набора номера (ATD). В этом случае модем набирает номер и пытается установить связь с удаленным модемом. Или передать модему команду AT S0 = 1 для перевода его в режим автоответа. После этого модем ожидает звонка от удаленного модема, а когда он приходит, пытается установить с ним связь.

4. Ожидание ответа от модема. В зависимости от режима, в котором находится модем, он может передавать компьютеру различные сообщения. Например, если модем производит вызов удаленного модема (AT-команда ATD), то модем может выдать следующие сообщения: CONNECT, BUSY, NO DIALTONE, NO ANSWER, NO CARRIER. Когда приходит звонок, модем передает компьютеру сообщение RING, если регистр модема S0 равен нулю. В этом случае для ответа на звонок надо послать модему команду ATA. Если модем находится в режиме автоответа и регистр модема S0 не равен нулю, то модем автоматически пытается ответить на звонок и может выдать следующие сообщения: CONNECT, NO DIALTONE, NO CARRIER.

Если модем передал компьютеру сообщение CONNECT, значит, он успешно произвел соединение и теперь работает в режиме передачи данных. Теперь все данные, которые вы передаются модему через COM-порт, будут преобразованы модемом в форму, пригодную для передачи по телефонным линиям, и переданы удаленному модему. И наоборот, данные, принятые модемом по телефонной линии, переводятся в цифровую форму и могут быть прочитаны через COM-порт, к которому подключен модем.

Если модем передал компьютеру сообщения BUSY, NO DIALTONE, NO ANSWER, NO CARRIER, значит, произвести соединение с удаленным модемом не удалось и надо попытаться повторить соединение.

5. Переключение модема в командный режим. После окончания работы коммуникационная программа должна перевести модем в командный режим и передать ему команду положить трубку (ATH0). Для перевода модема в командный режим можно воспользоваться Escape-последовательностью «+++». После того как модем перешел в командный режим, можно опять передавать ему AT-команды.

6. Сброс сигналов на линиях DTR и RTS. Низкий уровень сигналов DTR и RTS сообщает модему, что компьютер не готов к приему данных через COM-порт.

При работе с асинхронным последовательным адаптером можно использовать механизм прерываний. Т. к. передача и прием данных модемом представляют собой длительный процесс, то применение прерываний от порта позволяет использовать процессорное время для других нужд.

Описание используемых аппаратных и программных средств

Для выполнения лабораторной работы необходимо использовать модемы, поддерживающие систему AT-команд и терминальную программу.

Модем AMT STAR 2442e (внешний) фирмы AMT International Industries Ins (США) и модем COMPLUS 2442e (внешний) фирмы OSMOS (США). Назначение – 2-х проводные коммутируемые и некоммутируемые телефонные каналы, модуляция v.21, v.22, v.22 бис, коррекция ошибок и сжатие данных по протоколам NNP до 5 класса, режим передачи – дуплекс, асинхронный и синхронный режим.

Эмулятор терминала Telemax 3.00 разработан по заказу фирмы Symantec для оболочки Norton Commander 5.00 и предназначен для эмуляции терминала и пересылки файлов. Telemax может набрать телефонный номер и установить связь, эмулировать различные терминалы после установки связи, посылать и принимать файлы с использованием протоколов Xmodem, Zmodem, Kermit и ASCII.

Порядок выполнения работы

1. Выключить компьютеры. Подключить модемы. Включить компьютеры и модемы.
2. Запустить терминальную программу (telemax.exe).
3. Проверить работоспособность модемов и правильность настройки командой AT.
4. Изучить основные AT-команды, назначение S-регистров и ответные коды модемов.
5. Провести инициализацию модемов.
6. Провести запрос результирующего кода и проверочной суммы ROM.
7. Изучить текущее содержимое S-регистров и энергонезависимой памяти.
8. Провести подстройку модемов для текущей телефонной линии. Для этого разрешить эхо; разрешить выдачу сообщений модемом; установить время ожидания несущей после набора равным 50 с; установить минимально необходимое время присутствия на линии несущей удалённого модема для её опознания равным 3 с; установить интервал времени между моментом потери несущей удалённого модема и моментом разрыва связи локальным модемом равным 8 с, установить паузу для передачи ESCAPE последовательности равную 0,5 с, установить длительность теста модема равную 10 с.
9. Записать настройку в энергонезависимую память модема.
10. Просмотреть изменения в памяти.
11. Провести локальный аналоговый тест.
12. Провести локальный аналоговый тест с самотестированием.
13. Провести локальный цифровой тест.
14. Провести удаленный цифровой тест с самодиагностикой.
15. Произвести соединение с удаленным модемом и передать на удаленный терминал файл размером порядка 30 – 50 kb по протоколам ASCII, X-модем, Z-модем и Kermit. В каждом случае замерить время передачи. Проверить возможности протокола Z-модем для передачи файла по частям.
16. Выйти из терминальной программы.
17. Выключить компьютер. Отсоединить модемы от компьютера и телефонной сети и сдать преподавателю.

Контрольные вопросы

1. Опишите основные сигналы модемной связи.
2. Опишите сигналы интерфейса RS-232, необходимые для обмена сигналами между адаптером компьютера и модемом.

3. Какие вы знаете типы модемов?
4. Опишите протоколы обмена данными, используемые при передаче данных по телефонным линиям.
5. Опишите основные принципы программирования модемов.
6. Опишите последовательность действий для установления связи.

Содержание отчета

Отчет должен содержать:

1. титульный лист;
2. тему и цель лабораторной работы;
3. задание на лабораторную работу;
4. результаты и выводы по каждому выполненному пункту;
5. выводы по результатам работы.

Литература

1. Кулаков, В. Программирование на аппаратном уровне: спец. справочник / В. Кулаков. – 2-е изд. – СПб. : Питер, 2003.
2. Деревянко, А. С. Системное программное обеспечение персональных ЭВМ: учеб. пособие / А. С. Деревянко. – Харьков : ХГПУ, 1994.
3. Нортон, П. Программно-аппаратная организация компьютера IBM PC / П. Нортон; пер. с англ. С. Писарева, Б. Шура. – Киев, 1987.
4. Журден, Р. Справочник программиста на персональном компьютере фирмы IBM / Р. Журден. – P.-Prentice-Hall Press, 1986.

Лабораторная работа № 6

ИССЛЕДОВАНИЕ МОНИТОРА НА БАЗЕ ЭЛТ

Цель работы – изучить основные принципы работы ЭЛТ-монитора.

Теоретические сведения

Цвет и цветовое зрение

Из всего спектра существующих в природе электромагнитных волн лишь волны небольшого диапазона (380 – 760 нм) вызывают раздражение зрительного органа человека. Но и в этом интервале глаз воспринимает свет по-разному, поскольку каждой длине световой волны соответствует свой цвет. Фиолетовая область спектра лежит в пределах от 380 до 450 нм, синяя – от 450 до 480, голубая – от 480 до 510, зеленая – от 510 до 570, желтая – от 570 до 590, оранжевая – от 590 до 620 и красная – от 620 до 760 нм. Разделение видимого спектра на семь цветовых областей условно, т. к. между ними не существует четкой границы. В действительности глаз различает в спектре до 180 промежуточных цветовых оттенков.

В основе систем цветного телевидения лежит теория трехкомпонентного цветового зрения, впервые сформулированная еще в 1756 г. М. В. Ломоносовым. Согласно этой теории, в сетчатой оболочке человеческого глаза содержатся три вида колбочек, обладающих различной спектральной чувствительностью. При раздельном возбуждении того или иного вида колбочек создается ощущение красного (R), зеленого (G) или синего (B) цвета. В случае одновременного возбуждения двух видов колбочек, например, чувствительных к красному и зеленому цветам, возникает ощущение желтого цвета. Световые лучи, падающие от наблюдаемого предмета на сетчатку глаза, воздействуют сразу на колбочки всех трех видов. Неодинаковая степень возбуждения различных видов колбочек создает ощущение цветного изображения. При одновременном (в одинаковой степени) возбуждении всех трех видов колбочек возникает впечатление белого цвета. Глаз наиболее чувствителен к зеленому (G) цвету, менее – к красному (R) и еще меньше – к синему (B). Относительная спектральная чувствительность глаза – это зависимость визуальной яркости световых излучений от длины волны.

Окружающая природа имеет широкую гамму цветов, но любой цвет можно воспроизвести, сочетая определенным образом три цвета – красный, синий и зеленый. Способ образования цвета, основанный на сложении (смешении) световых потоков, называется аддитивным (суммирую-

щим). Аддитивное смешение может быть последовательным (поочередным) или одновременным. В вещательном цветном телевидении применяется одновременное аддитивное смешение. При построении системы цветного телевидения были учтены особенности цветового зрения и зрения вообще. Так, многочисленные исследования показали, что разрешающая способность глаза зависит от яркости, контрастности и цветности двух рассматриваемых мелких деталей. Наибольшая разрешающая способность бывает при рассмотрении черно-белых деталей, а также деталей, окрашенных в зеленый цвет. Это объясняется явлением хроматической аберрации в оптической системе глаза. Суть ее заключается в том, что для различных длин волн коэффициент преломления хрусталика глаза неодинаков. Вследствие этого фокус для синих лучей располагается несколько ближе к хрусталику, т. е. перед сетчаткой, для красных лучей – дальше от него, за сетчаткой, и только зеленые лучи фокусируются на сетчатке глаза. Таким образом, из-за хроматической аберрации мы не можем одновременно видеть одинаково четко все элементы цветного изображения. Эта особенность зрения учитывается при выборе методов кодирования сигналов в системах цветного телевидения.

Основные понятия колометрии

Колориметрия занимается вопросами измерения цвета и определения составляющих любой цветной смеси (от лат. колор – цвет, метрон – мера). Любой цвет, видимый глазом, является трехмерной величиной, определяемой яркостью, цветовым тоном и насыщенностью. Яркость – это количественная характеристика цвета, а цветовой тон и насыщенность – качественные. Цвета, качественно одинаковые, но обладающие разной яркостью, вызывают различные зрительные ощущения. Например, белый, светло-серый и темно-серый цвета кажутся белым цветом, имеющим разную яркость. Цвет, который при большой яркости воспринимается глазом как желтый, при малой яркости – как коричневый. Оба качественных параметра – цветовой тон и насыщенность – определяют цветность светового потока независимо от его яркости. Цветовой тон (оттенок) характеризует свойство цвета, отличающее его от белого и серого. Это свойство позволяет оценить данный цвет как красный, голубой, зеленый и др. Насыщенность определяет чистоту цвета, т. е. степень разбавленности его белым цветом (например, синий, светло-синий, голубой и т. д.). Чем больше белого (бледнее цвет), тем меньше его насыщенность, а чем меньше примеси белого в данном цветовом тоне, тем больше насыщенность. Более точно насыщенность представляет собой число цветовых порогов, т. е. едва заметных переходов (изменений), отделяющих данный цвет от белого рав-

ной с ним яркости. Для изучения законов смешения цветов в колориметрии используется диаграмма цветности, которая позволяет производить расчеты, связанные с разложением и синтезом различных цветовых излучений (рис. 6.1).

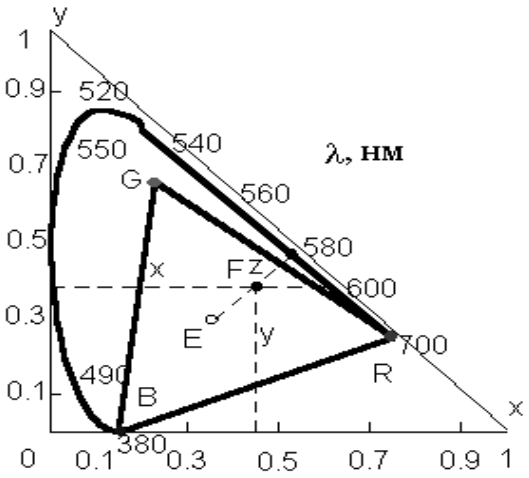

Рис. 6.1. **Диаграмма цветности**

Диаграмма цветности представляет собой прямоугольный треугольник, плоскость которого покрыта координатной сеткой. Внутри треугольника размещена подковообразная фигура, получившая название локус цветов. По периметру локуса располагаются чистые спектральные цвета, обладающие предельной насыщенностью – от сиреневого до красного. Локус является незамкнутой фигурой. Цвета, лежащие на прямой, которая соединяет сиреневый край спектра с красным, и представляющие собой смесь синего и красного в разных пропорциях, являются не спектральными. Это – пурпурные, а также те цвета, которые лежат внутри диаграммы цветности. Локус вместе с линией пурпурных цветов охватывает все цвета, видимые человеческим глазом. Точки, лежащие на диаграмме цветности вне локуса, не соответствуют никаким реальным цветам и поэтому физического смысла не имеют. Вся площадь диаграммы цветности разделена на отдельные участки различного цвета, реальные границы между которыми выражены не резко. На диаграмме обозначен треугольник основных цветов RGB, внутри которого содержатся все цвета, включая белый. Насыщенные основные цвета редко встречаются в повседневной жизни. Чаще – это более бледные, расположенные ближе к центру диаграммы, так называемые пастельные цвета: розовый, светло-зеленый или бледно-синий. В центральной части диаграммы находятся белые цвета, которые имеют слегка голубоватый оттенок произвольной яркости. Если на локусе цветов провести прямую, соединяющую точку белого цвета с любой точкой на кри-

вой, ограничивающей локус, то на прямой будут расположены цвета различной насыщенности, но одного цветового тона. Чем ближе к точке белого цвета, тем менее насыщенным оказывается цвет. Таким образом, любая точка на локусе цветов дает наглядное представление о цветовом тоне и насыщенности, а также о способах получения цвета посредством смешения различных цветов. Для измерения цвета применяют специальные приборы – колориметры.

Устройство ЭЛТ-монитора

Все ЭЛТ-мониторы имеют общий элемент – электронно-лучевую трубку, которая, собственно, и дала такое название мониторам (рис. 6.2). Трубка заполнена вакуумом и в ней содержится несколько элементов. Катод в задней части излучает электроны при нагреве. Электронная пушка «выстреливает» электроны в сторону анода, поэтому поток электронов движется с задней части кинескопа на экран. При этом поток электронов проходит через две катушки, которые направляют луч. Одна катушка отвечает за вертикальное отклонение, другая – за горизонтальное. Итак, как видим, трубка не имеет движущихся частей, что гарантирует долговечность. Если монитор цветной, то в нем используется три электронные пушки, каждая из них отвечает за свой цвет – красный, синий или зеленый. Такую технологию называют аддитивной цветовой технологией. Полутона на экране образуются из трех цветов, в зависимости от их интенсивности. Свечение происходит при попадании электронов на частички люминофора с внутренней поверхности трубки. Частички очень близко расположены друг к другу, так что три частички разных цветов воспринимаются глазом как один пиксель.

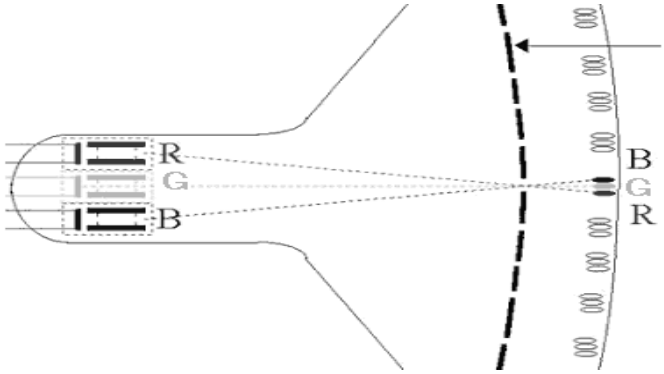

Рис. 6.2. **Электронно-лучевая трубка**

Классификация мониторов по типу маски

Современные мониторы с любой маской имеют практически плоскую форму экрана, благодаря которой существенно снижаются искажения

геометрии, особенно по углам. Поэтому тип маски по форме экрана определить не так просто.

На сегодняшний день в ЭЛТ-дисплеях используются три основные технологии формирования матриц и масок для RGB-триад:

- трехточечная теневая маска (DOT-TRIO SHADOW-MASK CRT);
- щелевая апертурная решетка (APERTURE-GRILLE CRT);
- гнездовая маска (SLOT-MASK CRT).

При создании мониторов помимо масок используются различные отклоняющие системы и другая электроника. Хотя сам экран и является наиболее важным фактором, определяющим эксплуатационные параметры дисплея, отклоняющая система и видеоусилитель также играют важную роль.

Наиболее старая и широко используемая технология с так называемой теневой маской использует перфорированную металлическую пластину, помещаемую перед люминофором (рис. 6.3).

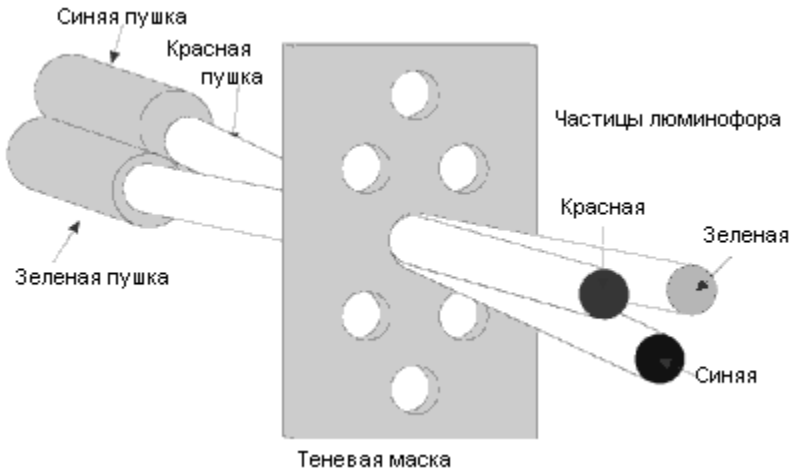

Рис. 6.3. **Теневая маска**

Она маскирует три отдельных луча, каждый из которых управляется собственной электронной пушкой. Маскирование обеспечивает необходимую концентрацию каждого луча и обеспечивает его попадание только на нужный цветовой участок люминофора. Однако практика показывает, что ни один из мониторов не обеспечивает идеального выполнения этой задачи по всей поверхности экрана.

Ранние ЭЛТ-дисплеи с теневой маской имели выраженную криволинейную (сферическую) поверхность. Это позволяло добиваться лучшей фокусировки и уменьшало нежелательные эффекты и отклонения, вызываемые нагревом. В настоящее время большинство профессиональных и

специализированных мониторов имеет практически плоский прямоугольный экран.

Мониторы с теневой маской имеют свои преимущества:

- текст выглядит лучше (особенно при малом размере точек);
- цвета «натуральнее» и точнее (что особенно важно для компьютерной графики и в полиграфии);
- отлаженная технология обеспечивает лучшее соотношение стоимости и эксплуатационных качеств.

Из недостатков можно отметить меньшую яркость таких мониторов, недостаточную контрастность изображения и более короткий срок службы, по сравнению с другими типами дисплеев.

Новую технологию изготовления CRT-дисплеев – с апертурной решеткой вместо традиционной точечной маски – впервые предложила фирма Sony, выпустив мониторы с трубкой Trinitron. В электронных пушках этих трубок используются динамические квадрупольные магнитные линзы, позволяющие формировать очень тонкий и точно направленный пучок электронов.

Благодаря такому решению значительно снижается астигматизм – рассеивание электронного пучка, приводящее к недостаточной резкости и контрастности изображения (особенно по горизонтали). Но главное отличие от технологии с теневой маской состоит в том, что вместо металлической пластины с круглыми отверстиями, выполняющей функции маски, здесь используется вертикальная проволочная сетка (апертурная решетка) и люминофор наносится не в виде точек, а в виде вертикальных полос (рис.6.4).

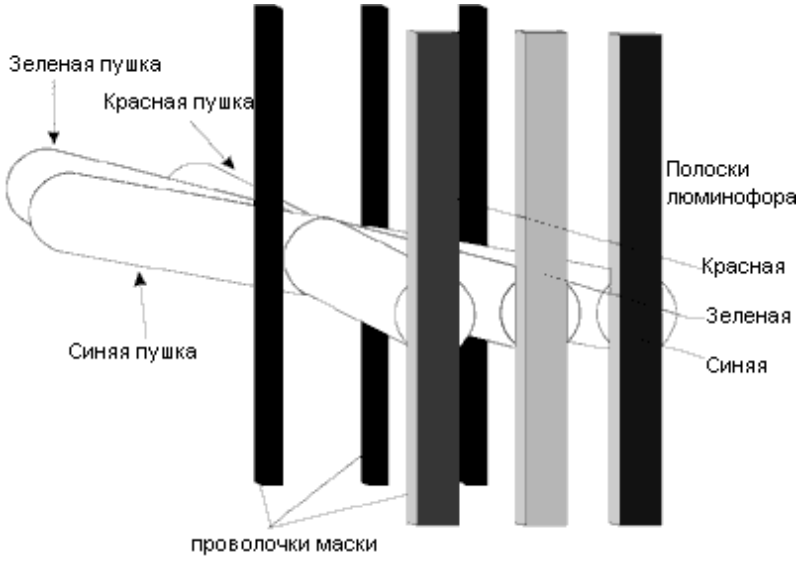

Рис. 6.4. **Апертурная решетка**

Мониторы с апертурной решеткой имеют следующие преимущества:

- в тонкой сетке меньше металла, что позволяет использовать больше энергии электронов на реакцию с люминофором, а значит, меньше рассеивается на решетке и уходит в тепло;
- увеличенная площадь покрытия люминофором позволяет повысить яркость излучения при той же интенсивности пучка электронов;
- в связи со значительным общим повышением яркости можно использовать более темное стекло и получать на экране более контрастное изображение;
- экран монитора с апертурной решеткой более плоский, чем у дисплеев с теневой маской, а в последних моделях даже не цилиндрический, как раньше, а почти абсолютно ровный, что гораздо удобнее в работе и уменьшает количество бликов и отражений.

Из недостатков можно отметить только «неприятные» горизонтальные нити – ограничители, используемые в таких мониторах для придания проволочной сетке дополнительной жесткости. Хотя проволочки в апертурной решетке туго натянуты, в процессе работы они могут вибрировать под воздействием пучков электронов. Демпферная нить (а в экранах больших размеров – две нити) служит для ослабления колебаний и гашения вибрации. По этим нитям мониторы с трубкой Trinitron можно отличить от других моделей. Кроме того, если в процессе работы такого монитора его слегка качнуть, колебания изображения будут видны даже невооруженным глазом. Именно поэтому мониторы с этими трубками не рекомендуется ставить на системные блоки типа desktop.

Остается добавить, что в электронно-лучевых трубках Sony Trinitron используется система трех пучков электронов, излучаемых одной пушкой, а в трубках с подобной апертурной решеткой компании Mitsubishi – Diamondtron – система из трех лучей с тремя пушками.

И, наконец, последний, комбинированный тип электронно-лучевой трубки, так называемый CromaClear/ OptiClear (впервые предложенный фирмой NEC) – это вариант теневой маски, в которой используются не круглые отверстия, а щели, как в апертурной решетке, только короткие – «пунктиром», и люминофор наносится в виде таких же эллиптических полосок, а полученные таким образом гнезда для большей равномерности расположены в «шахматном» порядке.

Такая гибридная технология позволяет сочетать все преимущества вышеописанных типов при отсутствии их недостатков. Четкий и ясный текст, натуральные, но достаточно яркие цвета и высокая контрастность

изображения неизменно привлекают к этим мониторам все группы пользователей.

Разрешение

Разрешающая способность характеризует качество воспроизведения изображения монитором. Для получения высокого разрешения в первую очередь высококачественным должен быть видеосигнал. Электронные цепи должны обработать его таким образом, чтобы обеспечить правильные уровни и сочетания фокусировки, цвета, яркости и контраста. Разрешающая способность характеризуется числом точек, или пикселов (dot), на число строк (line). Например, разрешение монитора 1024 x 768 означает возможность различить до 1024 точек по горизонтали при числе строк до 768.

Горизонтальная развертка

Время горизонтального перемещения луча от левого до правого края экрана называется периодом горизонтальной развертки. Величина, обратно пропорциональная этому периоду, называется частотой горизонтальной развертки, или просто горизонтальной разверткой (иногда встречаются названия «частота строчной развертки», или «строчная частота»), и измеряется в килогерцах (кГц). Например, для монитора с разрешением 1 024 x 768 пикселов горизонтальная развертка обратно пропорциональна времени, за которое луч сканирует 1 024 пиксела. При увеличении разрешающей способности за тот же период времени лучом должно быть отсканировано большее число пикселов. При увеличении частоты кадров частота горизонтальной развертки также должна быть увеличена.

Вертикальная развертка, или частота кадров

Монитор с электронно-лучевой трубкой обновляет изображение на экране десятки раз в секунду. Это число называется частотой вертикальной развертки, или частотой обновления экрана, и измеряется в герцах (Гц).

Монитор с вертикальной разверткой 60 Гц имеет такую частоту мерцания, как лампа дневного света в США (несколько выше, чем в Европе, где частота сети 50 Гц). Обычно при частотах выше 75 Гц мерцание незаметно для глаза (режим без мерцания). Стандарт VESA рекомендует работу на частоте 85 Гц, считая это важным потребительским показателем эргономичности монитора.

Расчет частоты горизонтальной развертки исходя из частоты кадров: Горизонтальная развертка = (число строк) x (вертикальная развертка) x 1,05. Например, требуемая горизонтальная развертка при вертикальной частоте 85 Гц и разрешении 1 024 x 768 составляет: 768 x 85 x 1,05 = 68 500 Гц = 68,5 кГц.

Описание используемых программных и аппаратных средств

Программа DISPLAY предназначена для исследования R G B сигналов ЭЛТ-монитора. В программе предлагается 10 тестов. Выбор теста осуществляется по клавише ввод. Переход между изображениями в тесте осуществляется нажатием любой клавиши на клавиатуре.

Осциллограф С1-76 предназначен для исследования сигналов с амплитудой 0,5мВ – 300 В и длительностью 1мкс – 50с. Отличается высокой чувствительностью канала вертикального отклонения и большим рабочим полем экрана. Установка идентичных режимов работы усилителей в каналах вертикального и горизонтального отклонения луча позволяет наблюдать фигуры Лиссажу и вольтамперные характеристики с разностью фазовых характеристик не более 3 в полосе частот 0 – 100 кГц.

Порядок выполнения работы

1. Включить осциллограф, на компьютере запустить программу DISPLAY.
2. Выбрать клавишу TEST1. В данном режиме необходимо при помощи осциллографа получить формы импульсов входных R G B сигналов, идущих от системного блока, а также выходных R G B сигналов, поступающих после видеоусилителей на катоды электронно-лучевой трубки. Измерить амплитуды входных и выходных сигналов R G B для каждого цвета. Для выходных сигналов амплитуда измеряется при положении ручки CONTRAST в крайнем правом и крайнем левом положении. Данные занести в таблицу.
3. Клавиши TEST2, TEST3, TEST4 – цветные полосы с высокой, низкой и совмещенной интенсивностью соответственно. В данных режимах необходимо зарисовать формы импульсов входных сигналов R G B и определить, каким амплитудам R G B сигналов соответствует определенная цветная полоса на экране. Определить разницу между сигналами R G B с высокой и низкой интенсивностью.
4. Клавиша TEST5. В данном пункте необходимо измерить и вычислить частоту следования R G B импульсов для каждого из 4 изображений на экране. Переход к следующему изображению осуществляется нажатием любой клавиши на клавиатуре.
5. Клавиша TEST6 предназначена для работы с двухканальным осциллографом для наблюдения одновременно двух сигналов RG, BG или

RB на экране осциллографа. В данном пункте необходимо измерить длительность импульсов сигналов RG, BG и RB для каждого пункта теста.

6. Клавиши TEST7, TEST8, TEST9 предназначены для исследования R G B сигналов при выборе разных разрешений монитора. Следует измерить длительность любого из R G B сигналов при переходе от одного изображения к другому для всех трех разрешений.

7. Клавиша TEST10 служит для корректировки изображения на экране при помощи резисторов, находящихся на задней панели монитора. В этом режиме необходимо получить формы строчного и кадрового отклоняющих напряжений на отклоняющих катушках электронно-лучевой трубки.

Контрольные вопросы

1. Опишите основные принципы цветового зрения.
2. Опишите основные понятия колометрии.
3. Опишите устройство ЭЛТ-монитора.
4. Приведите классификацию мониторов по типу маски.
5. Дайте определение следующим понятиям: разрешение, горизонтальная развертка, вертикальная развертка.

Содержание отчета

Отчет должен содержать:

1. титульный лист;
2. тему и цель лабораторной работы;
3. результаты и выводы по каждому выполненному пункту;
4. выводы по результатам работы.

Литература

1. Кулаков, В. Программирование на аппаратном уровне: спец. справочник / В. Кулаков. – 2-е изд. – СПб. : Питер, 2003.
2. Деревянко, А. С. Системное программное обеспечение персональных ЭВМ: учеб. пособие / А. С. Деревянко. – Харьков : ХГПУ, 1994.
3. Нортон, П. Программно-аппаратная организация компьютера IBM PC / П. Нортон; пер. с англ. С. Писарева, Б. Шура. – Киев, 1987.
4. Журден, Р. Справочник программиста на персональном компьютере фирмы IBM / Р. Журден. – Р.-Prentice-Hall Press, 1986.

Лабораторная работа № 7

ПОСЛЕДОВАТЕЛЬНЫЙ ИНТЕРФЕЙС RS-232C

Цель работы – исследовать основные принципы работы адаптера последовательного интерфейса и приобрести навыки его программирования.

Теоретические сведения

Практически каждый компьютер оборудован хотя бы одним последовательным асинхронным адаптером. Обычно он представляет собой отдельную плату или же расположен прямо на материнской плате компьютера. Его называют еще асинхронным адаптером RS-232-C, или портом RS-232-C. Каждый асинхронный адаптер обычно содержит несколько портов RS-232-C, через которые к компьютеру можно подключать внешние устройства. Каждому такому порту соответствует несколько регистров, через которые программа получает к нему доступ, и определенная линия IRQ для сигнализации компьютеру об изменении состояния порта. При выполнении BIOS процедуры начальной загрузки каждому порту RS-232-C присваивается логическое имя COM1 – COM4 (COM-порт номер 1 – 4).

Интерфейс RS-232-C разработан ассоциацией электронной промышленности (Electronic Industries Association – EIA) как стандарт для соединения компьютеров и различных последовательных периферийных устройств.

Компьютер IBM PC поддерживает интерфейс RS-232-C не в полной мере, скорее разъем, обозначенный на корпусе компьютера как порт последовательной передачи данных, содержит некоторые из сигналов, входящих в интерфейс RS-232-C и имеющих соответствующие этому стандарту уровни напряжения.

В настоящее время порт последовательной передачи данных используется очень широко:

- подключение мыши;
- подключение графопостроителей (плоттеров), сканеров, принтеров, дигитайзеров;
- связь двух компьютеров через порты последовательной передачи данных с использованием специального кабеля;
- подключение модемов для передачи данных по телефонным линиям;
- подключение к сети персональных компьютеров.

Основные понятия и термины

Последовательная передача данных означает, что данные передаются по единственной линии. При этом биты байта данных передаются по очереди с использованием одного провода. Для синхронизации группе битов данных обычно предшествует специальный стартовый бит, после группы битов следуют бит проверки на четность и один или два стоповых бита. Иногда бит проверки на четность может отсутствовать.

Сказанное иллюстрируется рис. 7.1.

Рис. 7.1. **Формат последовательной передачи данных**

Из рис. видно, что исходное состояние линии последовательной передачи данных – уровень логической 1. Это состояние линии называют отмеченным – MARK. Когда начинается передача данных, уровень линии переходит в 0. Это состояние линии называют пустым – SPACE. Если линия находится в таком состоянии больше определенного времени, считается, что линия перешла в состояние разрыва связи – BREAK.

Стартовый бит START сигнализирует о начале передачи данных. Далее передаются биты данных, вначале младшие, затем старшие.

Если используется бит четности P, то передается и он. Бит четности имеет такое значение, чтобы в пакете битов общее количество единиц (или нулей) было четно или нечетно, в зависимости от установки регистров порта. Этот бит служит для обнаружения ошибок, которые могут возникнуть при передаче данных из-за помех на линии. Приемное устройство заново вычисляет четность данных и сравнивает результат с принятым битом четности. Если четность не совпала, то считается, что данные переданы с ошибкой. Конечно, такой алгоритм не дает стопроцентной гарантии обнаружения ошибок. Так, если при передаче данных изменилось четное число битов, то четность сохраняется, и ошибка не будет обнаружена. Поэтому на практике применяют более сложные методы обнаружения ошибок.

В самом конце передаются один или два стоповых бита STOP, завершающих передачу байта. Затем до прихода следующего стартового бита линия снова переходит в состояние MARK.

Использование бита четности, стартовых и стоповых битов определяют формат передачи данных. Очевидно, что передатчик и приемник должны использовать один и тот же формат данных, иначе обмен будет невозможен.

Другая важная характеристика – скорость передачи данных. Она также должна быть одинаковой для передатчика и приемника.

Скорость передачи данных обычно измеряется в бодах (по фамилии французского изобретателя телеграфного аппарата Emile Baudot – Э. Бодо). Боды определяют количество передаваемых битов в секунду. При этом учитываются и старт/ стопные биты, и бит четности.

Иногда используется другой термин – биты в секунду (bps). Здесь имеется в виду эффективная скорость передачи данных, без учета служебных битов.

Аппаратная реализация

Компьютер может быть оснащен одним или двумя портами последовательной передачи данных. Эти порты расположены либо на материнской плате, либо на отдельной плате, вставляемой в слоты расширения материнской платы.

Бывают также платы, содержащие четыре или восемь портов последовательной передачи данных. Их часто используют для подключения нескольких компьютеров или терминалов к одному, центральному компьютеру. Эти платы имеют название «мультипорт».

В основе последовательного порта передачи данных лежит микросхема Intel 8 250 или ее современные аналоги – Intel 16 450, 16 550, 16 550A. Эта микросхема является универсальным асинхронным приемопередатчиком (UART – Universal Asynchronous Receiver Transmitter). Микросхема содержит несколько внутренних регистров, доступных через команды ввода/ вывода.

Микросхема 8 250 содержит регистры передатчика и приемника данных. При передаче байта он записывается в буферный регистр передатчика, откуда затем переписывается в сдвиговый регистр передатчика. Байт «выдвигается» из сдвигового регистра по битам. Аналогично имеются сдвиговый и буферный регистры приемника.

Программа имеет доступ только к буферным регистрам, копирование информации в сдвиговые регистры, и процесс сдвига выполняется микросхемой UART автоматически.

К внешним устройствам асинхронный последовательный порт подключается через специальный разъем. Существует два стандарта на разъемы интерфейса RS-232-C, это DB25 и DB9. Первый разъем имеет 25 контактов (табл. 7.1).

Таблица 7.1

Разводка разъема последовательной передачи данных DB25

Номер контакта	Назначение контакта	Вход или выход
1	Защитное заземление(Frame Ground, FG)	–
2	Передаваемые данные (Transmitted Data, TD)	Выход
3	Принимаемые данные (Received Data, RD)	Вход
4	Запрос для передачи (Request to send, RTS)	Выход
5	Сброс для передачи (Clear to Send, CTS)	Вход
6	Готовность данных (Data Set Ready, DSR)	Вход
7	Сигнальное заземление (Signal Ground, SG)	–
8	Детектор принимаемого с линии сигнала (Data Carrier Detect, DCD)	Вход
9 – 19	Не используются	
20	Готовность выходных данных (Data Terminal Ready, DTR)	Выход
21	Не используется	
22	Индикатор вызова (Ring Indicator, RI)	Вход
23 – 25	Не используются	

Наряду с 25-контактным разъемом часто используется 9-контактный разъем (табл. 7.2).

Таблица 7.2

Разводка разъема последовательной передачи данных DB9

Номер контакта	Назначение контакта	Вход или выход
1	Детектор принимаемого с линии сигнала(Data Carrier Detect, DCD)	Вход
2	Принимаемые данные (Received Data, RD)	Вход
3	Передаваемые данные (Transmitted Data, TD)	Выход
4	Готовность выходных данных (Data Terminal Ready, DTR)	Выход
5	Сигнальное заземление (Signal Ground, SG)	–
6	Готовность данных(Data Set Ready, DSR)	Вход
7	Запрос для передачи (Request to send, RTS)	Выход
8	Сброс для передачи (Clear to Send, CTS)	Вход
9	Индикатор вызова (Ring Indicator, RI)	Вход

Только два вывода этих разъемов используются для передачи и приема данных. Остальные передают различные вспомогательные и

управляющие сигналы. На практике для подсоединения того или иного устройства может понадобиться различное количество сигналов.

Интерфейс RS-232-C определяет обмен между устройствами двух типов: DTE (Data Terminal Equipment – терминальное устройство) и DCE (Data Communication Equipment – устройство связи). В большинстве случаев, но не всегда, компьютер является терминальным устройством. Модемы, принтеры, графопостроители всегда являются устройствами связи.

Сигналы интерфейса RS-232-C. Входы TD и RD используются устройствами DTE и DCE по-разному. Устройство DTE использует вход TD для передачи данных, а вход RD для приема данных. И наоборот, устройство DCE использует вход TD для приема, а вход RD для передачи данных. Поэтому для соединения терминального устройства и устройства связи выводы их разъемов необходимо соединить напрямую.

Процесс подтверждения связи между компьютером и модемом происходит следующим образом. В начале сеанса связи компьютер должен удостовериться, что модем может произвести вызов (находится в рабочем состоянии). Затем, после вызова абонента, модем должен сообщить компьютеру, что он произвел соединение с удаленной системой. Подробнее это происходит следующим образом. Компьютер подает сигнал по линии DTR, чтобы показать модему, что он готов к проведению сеанса связи. В ответ модем подает сигнал по линии DSR. Когда модем произвел соединение с другим, удаленным модемом, он подает сигнал по линии DCD, чтобы сообщить об этом компьютеру.

Если напряжение на линии DTR падает, это сообщает модему, что компьютер не может далее продолжать сеанс связи, например, из-за того, что выключено питание компьютера. В этом случае модем прервет связь. Если напряжение на линии DCD падает, это сообщает компьютеру, что модем потерял связь и не может больше продолжать соединение. В обоих случаях эти сигналы дают ответ на наличие связи между модемом и компьютером. Это самый низкий уровень управлением связи – подтверждение связи. Существует более высокий уровень, который используется для управления скоростью обмена данными, но он также реализуется аппаратно. Практически управление скоростью обмена данными (управление потоком) необходимо, если производится передача больших объемов данных с высокой скоростью. Когда одна система пытается передать данные с большей скоростью, чем они могут быть обработаны принимающей системой, результатом может стать потеря части передаваемых данных. Чтобы предотвратить передачу большего числа данных, чем то, которое может быть обработано, используют управление связью, называемое «управление

потоком» (flow-controll handshake). Стандарт RS-232-C определяет возможность управления потоком только для полудуплексного соединения. Полудуплексным называется соединение, при котором в каждый момент времени данные могут передаваться только в одну сторону. Однако фактически этот механизм используется и для дуплексных соединений, когда данные передаются по линии связи одновременно в двух направлениях.

В полудуплексных соединениях устройство DTE подает сигнал RTS, когда оно желает передать данные. DCE отвечает сигналом по линии CTS, когда оно готово, и DTE начинает передачу данных. До тех пор, пока оба сигнала RTS и CTS не примут активное состояние, только DCE может передавать данные. При дуплексных соединениях сигналы RTS/ CTS имеют противоположные значения по сравнению с теми, которые они имели для полудуплексных соединений. Когда DTE может принять данные, он подает сигнал по линии RTS. Если при этом DCE готово для принятия данных, оно возвращает сигнал CTS. Если напряжение на линиях RTS или CTS падает, то это сообщает передающей системе, что получающая система не готова для приема данных. Однако в большинстве случаев этого недостаточно, т. к. для устройств DTE и DCE функции, выполняемые линиями DSR, DTR, DCD, CTS и RTS, асимметричны. Устройство DTE подает сигнал DTR и ожидает получения сигналов DSR и DCD. В свою очередь, устройство DCE подает сигналы DSR, DCD и ожидает получения сигнала DTR. Таким образом, соединить вместе два устройства DTE кабелем, который используется для соединения устройств DTE и DCE, то они не смогут договориться друг с другом. Не выполнится процесс подтверждения связи.

Иногда для соединения двух устройств DTE линии RTS и CTS соединяют вместе на каждом конце кабеля. В результате получается, что другое устройство всегда готово для получения данных. Поэтому, если при большой скорости передачи принимающее устройство не успевает принимать и обрабатывать данные, возможна потеря данных.

Чтобы решить все эти проблемы для соединения двух устройств типа DTE используется специальный кабель, в обиходе называемый нуль-модемом (рис. 7.2). Нуль-модемный кабель, представленный в левой части рисунка, содержит значительно меньше проводов, чем нуль-модемный кабель, изображенный справа. За счет того, что на каждом конце кабеля линии RTS, CTS и DSR, DCD, DTR соединены вместе, процедуры подтверждения связи и управления потоком всегда будут заканчиваться успешно. На больших скоростях это может привести к потере информации, поэтому лучше использовать вторую схему.

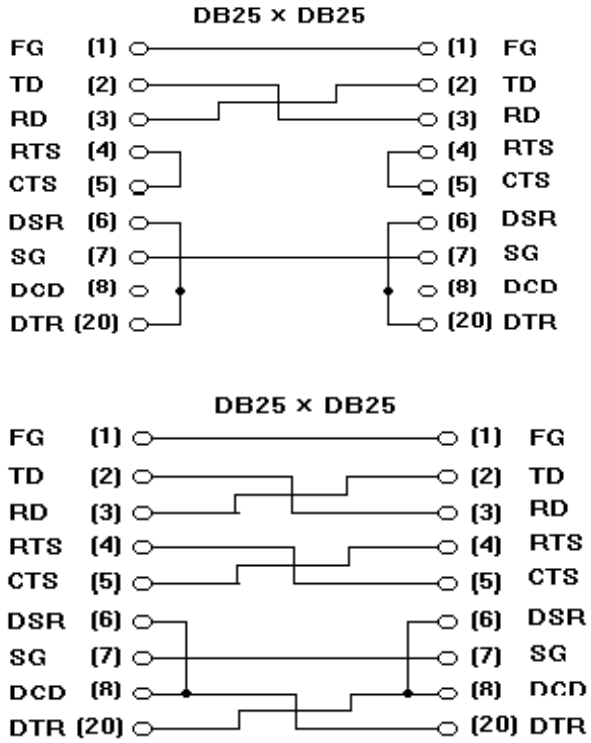

Рис. 7.2. **Схема нуль-модемного кабеля**

Технические параметры интерфейса RS-232-C. При передаче данных на большие расстояния без использования специальной аппаратуры из-за помех, наводимых электромагнитными полями, возможно возникновение ошибок. Вследствие этого накладываются ограничения на длину соединительного кабеля между устройствами DTR-DTR и DTR-DCE. Официальное ограничение по длине для соединительного кабеля по стандарту RS-232-C составляет 15,24 м. Однако на практике это расстояние может быть значительно больше. Оно непосредственно зависит от скорости передачи данных. Согласно McNamara (Technical Aspects of Data Communications, Digital Press, 1982) определены следующие значения (табл. 7.3).

Таблица 7.3

Длина соединительного кабеля для RS-232C

Скорость передачи, бод	Максимальная длина для экранированного кабеля, м	Максимальная длина для неэкранированного кабеля, м
110	1 524,0	914,4
300	1 524,0	914,4
1 200	914,4	914,4
2 400	304,8	152,4
4 800	304,8	76,2
9 600	76,2	76,2

Уровни напряжения на линиях разъема составляют для логического нуля –15..–3 В, для логической единицы +3..+15 В. Промежуток от –3 до +3 В соответствует неопределенному значению. При подключении устройства к разъему интерфейса RS-232-C (а также при соединении двух компьютеров нуль-модемом) необходимо предварительно выключить компьютер, а также снять статический заряд (подсоединив заземление). В противном случае можно вывести из строя асинхронный адаптер. Земля компьютера и земля внешнего устройства должны быть соединены вместе.

Программирование асинхронного адаптера

Подключение внешних устройств к системной шине, для связи с процессором, выполняется через дополнительные переходные устройства – адаптеры. В архитектуре IBM PC регистры ввода/ вывода соединяются с системной шиной через порты. Порт рассматривается как байтовая ячейка в пространстве адресов ввода/ вывода; диапазон адресов – от 0 до 0ffff. Поскольку общая размерность регистров устройства, как правило, превышает один байт, для соединения с устройством используется массив портов. Начальный адрес массива портов при соединении с некоторым устройством называют базовым адресом этого устройства.

Имеются функции MS-DOS и BIOS, предназначенные для работы с адаптером. Однако они не отличаются функциональной полнотой. Например, используя только функции MS-DOS нельзя проанализировать ошибочные ситуации и изменить режим работы асинхронного адаптера. Функции BIOS, обслуживающие адаптер, более разнообразны. Однако и им присущи недостатки. Например, нельзя установить скорость передачи более 9 600 бод или использовать режим фиксации четности. Нет возможности узнать текущий режим асинхронного адаптера, отсутствует поддержка модема. Поэтому для программирования асинхронного адаптера лучше использовать порты ввода/ вывода микросхемы 8 250.

Порты асинхронного адаптера. Способ доступа к регистрам устройства определяется схемой их подключения к портам ввода/ вывода. Если количество программно доступных регистров устройства превышает 6 – 7, то к одному порту обычно подсоединяют несколько регистров через схему коммутации. Регистр данных передатчика и регистр данных приемника последовательного канала связи подключены к общему порту, расположенному по смещению 0 от базового адреса устройства. При чтении этого порта к нему подключается регистр данных приемника последовательного канала связи, а при записи – регистр данных передатчика.

На этапе инициализации системы, модуль POST BIOS тестирует имеющиеся асинхронные порты RS-232-C и инициализирует их. В зависимости от версии BIOS инициализирует первые два или четыре порта. Их

базовые адреса располагаются в области данных BIOS, начиная с адреса 0000:0400h.

Первый адаптер COM1 имеет базовый адрес 3F8h и занимает диапазон адресов от 3F8h до 3FFh. Второй адаптер COM2 имеет базовый адрес 2F8h и занимает адреса 2F8h...2FFh. Третий адаптер COM3 имеет базовый адрес 3E8h и занимает диапазон адресов от 3E8h до 3EFh. Четвертый адаптер COM4 имеет базовый адрес 2E8h и занимает адреса 2E8h...2EFh.

Асинхронные адаптеры могут вырабатывать прерывания:

COM1, COM3 – IRQ4 (соответствует INT 0Ch);

COM2, COM4 – IRQ3 (соответствует INT 0Bh).

Регистр данных (base_adr). Регистр данных расположен непосредственно по базовому адресу порта RS-232-C и используется для обмена данными и для задания скорости обмена. Для передачи данных в этот регистр необходимо записать передаваемый байт данных. После приема данных от внешнего устройства принятый байт можно прочитать из этого же регистра. В зависимости от состояния старшего бита управляющего регистра (расположенного по адресу base_adr + 3, где base_adr соответствует базовому адресу порта RS-232-C), назначение этого регистра может изменяться. Если старший бит равен нулю, регистр используется для записи передаваемых данных. Если же старший бит равен единице, регистр используется для ввода значения младшего байта делителя частоты тактового генератора. Изменяя содержимое делителя, можно изменять скорость передачи данных. Старший байт делителя записывается в регистр управления прерываниями по адресу base_adr + 1. Зависимость скорости передачи данных от значения делителя частоты представлена в табл. 7.4.

Таблица 7.4

Зависимость скорости передачи данных от значения делителя частоты

Делитель, десятичная форма	Делитель, шестнадцатеричная форма	Скорость передачи, бод
1 040	600h	110
768	300h	150
384	180h	300
192	0C0h	600
96	60h	1 200
48	30h	2 400
24	18h	4 800
12	0Ch	9 600
6	6h	19 200
3	3h	38 400
2	2h	57 600
1	1h	115 200

Как следует из таблицы, максимальная скорость обмена информацией, которую можно достичь при использовании асинхронного адаптера, достигает 115 200 бод, что примерно соответствует 14 кбайт/ с.

Регистр управления прерываниями (base_adr + 1). Этот регистр используется либо для управления прерываниями от асинхронного адаптера, либо (после вывода в управляющий регистр байта с установленным в 1 старшим битом) для ввода значения старшего байта делителя частоты тактового генератора. В режиме управления прерываниями регистр имеет следующий формат (табл. 7.5).

Таблица 5

Назначение битов регистра управления прерываниями

Номер бита	Назначение
7	не используются, должны быть равны 0
6	
5	
4	
3	= 1 – разрешение прерывания по изменению состояния входных линий на разъеме RS-232-C (CTS, DSR, RI, DCD)
2	= 1 – разрешение прерывания по обнаружению состояния BREAK или по ошибке
1	= 1 – разрешение прерывания после передачи байта (когда выходной буфер передачи пуст)
0	= 1 – разрешение прерывания при готовности принимаемых данных

Регистр идентификации прерывания (base_adr + 2). Считывая содержимое регистра идентификации прерывания, программа может определить причину прерывания. Назначение битов регистра идентификации прерывания представлено в табл. 7.6.

Таблица 7.6

Назначение битов регистра идентификации прерывания

Номер бита	Назначение
7	должны быть равны 0
6	
5	
4	
3	
2	= 00 – состояние модема. Устанавливается при изменении состояния входных линий CTS, RI, DCD, DSR. Сбрасывается после чтения состояния модема из регистра состояния модема = 01 – буфер передатчика пуст. Сбрасывается при записи новых данных в регистр данных = 10 – данные приняты и доступны для чтения. Сбрасывается после чтения данных из регистра данных = 11 – прерывание по линии состояния приемника, возникает при переполнении приемника, ошибках четности или формата данных или при состоянии BREAK. Сбрасывается после чтения состояния линии из регистра состояния линии
1	
0	= 1 – нет прерываний, ожидающих обслуживания

Управляющий регистр (base_adr + 3). Управляющий регистр доступен по записи и чтению. Этот регистр управляет различными характеристиками UART: скоростью передачи данных, контролем четности, передачей сигнала BREAK, длиной передаваемых слов (символов). Назначение битов управляющего регистра представлено в табл. 7.7.

Таблица 7.7

Назначение битов управляющего регистра

Номер бита	Назначение
7	бит используется для доступа к регистру установки скорости: = 1 – регистр данных и регистр управления прерываниями используются для загрузки делителя частоты тактового генератора; = 0 – регистр данных и регистр управления прерываниями используются как обычно
6	=1 – установка перерыва. Вызывает вывод строки нулей в качестве сигнала BREAK для подключенного устройства
5	фиксация четности. При установке этого бита бит четности всегда принимает значение 0 (если биты 3 – 4 равны 11) или 1 (если биты 3 – 4 равны 01)
4	четность: =x0 – контроль на четность не выполняется; =01 – выполняется проверка на нечетность; =11 – выполняется проверка на четность
3	
2	Количество стоповых бит: =0 – 1 бит; =1 – 2 бита
1	Данные биты определяют длину передаваемых слов, бит: =00 – 5; =01 – 6; =10 – 7; =11 – 8
0	

Регистр управления модемом (base_adr + 4). Регистр управления модемом управляет состоянием выходных линий DTR, RTS и линий, специфических для модемов – OUT1 и OUT2, а также запуском диагностики присоединенных вместе входа и выхода асинхронного адаптера. Назначение битов регистра управления модема представлено в табл. 7.8.

Таблица 7.8

Назначение битов регистра управления модемом

Номер бита	Назначение
7	Должны быть равны 0
6	
5	
4	Запуск диагностики при входе асинхронного адаптера, замкнутом на его выход (Digital Loopback test). Эта возможность реализована только для асинхронных портов, использующих микросхему UART 8250, или полностью совместимых с ней

Окончание табл.

3	Линия OUT2 (запасная). Если бит равен единице, то UART может вырабатывать прерывания, а если нулю – не может
2	Линия OUT1 (запасная). Для некоторых модемов при установке этого бита в единицу происходит его аппаратный сброс
1	Линия RTS. Сигнал подтверждения связи. Используется модемами для разрешения передачи данных между компьютером и микросхемой UART
0	Линия DTR. Сигнал подтверждения связи. Используется модемами для разрешения передачи данных между компьютером и микросхемой UART

Регистр состояния линии (base_adr + 5). Регистр состояния линии определяет причину ошибок, которые могут произойти при передаче данных между компьютером и микросхемой UART. Назначение битов регистра состояния линии представлено в табл. 7.9.

Таблица 7.9

Назначение битов регистра состояния линии

Номер бита	Назначение
7	=1 – таймаут (устройство не связано с компьютером)
6	=1 – регистр сдвига передатчика пуст. Этот регистр получает данные из регистра хранения и преобразует их в последовательный вид для передачи. Если этот бит равен единице, то UART может принять очередной символ от компьютера
5	=1 – регистр хранения передатчика пуст, в него можно записывать новый байт для передачи
4	=1 – обнаружен запрос на прерывание передачи BREAK – длинная строка нулей
3	=1 – ошибка синхронизации. Возникает, например, при отсутствии стоп-битов
2	=1 – ошибка четности, сбрасывается после чтения состояния линии
1	=1 – ошибка переполнения. Был принят новый байт данных, а предыдущий еще не был считан программой. В результате предыдущий байт потерян
0	=1 – данные получены и готовы для чтения, при чтении данных бит сбрасывается

В табл. 7.9 упоминаются два регистра передатчика – буферный и сдвиговый. То, что называется регистром хранения – это буферный регистр, в который записываются данные через порт со смещением ноль. Данные, записанные в буферный регистр, поступают в сдвиговый регистр, как только закончена передача очередного байта. Байт из сдвигового регистра передается в линию по битам. Пока идет передача, сдвиговый регистр занят; а в освободившийся буферный регистр тем временем можно записать следующий байт для передачи.

Регистр состояния модема (base_adr + 6). Регистр состояния модема определяет состояние управляющих сигналов, передаваемых модемом асинхронному порту компьютера. Назначение битов регистра состояния модема представлено в табл. 7.10.

Таблица 7.10

Назначение битов регистра состояния модема

Номер бита	Назначение
7	состояние линии DCD. Единица означает, что модемом получена несущая частота. Заметим, что при выполнении аналогового теста (Analog test) этот бит должен содержать единицу. Если это не так, то возможно, что модем исправен (для внешних модемов), но кабель, соединяющий модем и компьютер, не полностью соответствует стандарту RS-232
6	состояние линии RI. Единица означает, что модем обнаружил звонок на телефонной линии
5	состояние линии DSR. Эта линия используется совместно с линией DTR при аппаратной реализации подтверждения связи
4	состояние линии CTS. Эта линия используется совместно с линией RTS при реализации аппаратного управления потоком данных
3	=1 – линия DCD изменила свое состояние. Некоторые коммуникационные программы определяют по состоянию этого бита, детектировал ли модем несущую частоту на телефонной линии
2	=1 – линия RI изменила состояние. Некоторые коммуникационные программы определяют по состоянию этого бита наличие звонка на телефонной линии
1	=1 – линия DSR изменила состояние
0	=1 – линия CTS изменила состояние

Инициализация асинхронного адаптера. Первое, что должна сделать программа, работающая с асинхронным адаптером, – установить формат и скорость передачи данных. После загрузки операционной системы для асинхронных адаптеров устанавливается скорость 2400 бод, не выполняется проверка на четность, используются один стоповый бит и восьмибитовая длина передаваемого символа. Выполнив чтение из управляющего регистра, программа может получить текущий режим адаптера. Для установки нового режима необходимо изменить нужные поля и записать новый байт режима обратно в управляющий регистр.

Если надо задать новое значение скорости обмена данными, перед записью байта режима необходимо установить старший бит этого байта в 1, при этом регистр данных и управляющий регистр используются для задания скорости обмена. Затем последовательно двумя командами загрузить делитель частоты тактового генератора. Младший байт записать в регистр данных, а старший – в регистр управления прерываниями.

Перед началом работы необходимо также проинициализировать регистр управления прерываниями (порт 3F9h), даже если в программе не используются прерывания от асинхронного адаптера. Для этого сначала надо перевести регистр данных и регистр управления прерываниями в обычный режим, записав ноль в старший бит управляющего регистра. Затем можно устанавливать регистр управления прерываниями. Если прерывания не нужны, необходимо записать в этот порт нулевое значение.

Передача данных. Перед записью байта данных в регистр передатчика необходимо убедиться в том, что регистр хранения передатчика свободен, т. е. убедиться в том, что передача предыдущего символа завершена. Признаком того, что регистр передатчика свободен, является установленный в 1 бит 5 регистра состояния линии с адресом base_adr + 5.

Прием данных. Аналогично передаче данных перед чтением символа из регистра данных (адрес base_adr) необходимо убедиться в том, что бит 0 регистра состояния линии (адрес base_adr + 5) установлен в 1. Это означает, что символ принят из линии и находится в буферном регистре приемника.

Использование прерываний. Т. к. процесс последовательной передачи данных протекает достаточно медленно, имеет смысл выполнять его в фоновом режиме, используя прерывания по окончании передачи или приема символа. Порту COM1 соответствует аппаратное прерывание INT 0Ch, а COM2 – INT 0Bh. Для разрешения прерываний необходимо установить в 1 биты регистра управления прерываниями, соответствующие тем прерываниям, которые необходимо обрабатывать.

Когда произошло прерывание, программа-обработчик прерывания должна проанализировать причину прерывания, прочитав содержимое регистра идентификации прерывания с адресом base_adr + 2.

В конце обработчика аппаратного прерывания должна находиться последовательность команд:

```
mov   al, 20h
out   20h, al
iret
```

для сброса контроллера прерываний.

Может случиться так, что одновременно произойдет несколько прерываний. В этом случае бит 0 регистра идентификации прерывания будет установлен в 1. Если такая ситуация имеет место, перед завершением обработки прерывания надо снова прочитать регистр идентификации прерывания и обработать следующее прерывание. Так следует поступать до тех пор, пока бит 0 регистра идентификации прерывания не станет равным нулю.

Последовательная передача в диагностическом режиме. Для наблюдения за результатами и процессом приема/ передачи можно воспользоваться диагностическим режимом адаптера. В этом режиме выход передатчика замкнут внутри адаптера на вход приемника. Для включения диагностического режима необходимо установить в единицу бит 4 регистра управления модемом, который подключен к порту со смещением четыре. После этого запись в регистр данных автоматически запускает вывод (передачу) записанного значения. При работе вручную передача кажется мгновенной; при начальной загрузке BIOS настраивает последовательные адаптеры на скорость 2400 бит/ с, поэтому передача байта вместе со служебными битами занимает не более 5 мсек. После этого бит готовности приемника установлен – передача завершена. Готовность приемника сохраняется до тех пор, пока принятые данные не будут прочитаны.

Функции языка Си для работы с портами ввода/ вывода. Для чтения данных из порта можно использовать функции:

```
int inport(int port);
unsigned char inportb(int port);
```

Для записи данных в порт можно использовать функции:

```
void outport(int port, int val);
void outportb(int port, unsigned char val);
```

Аргумент этих функций port – номер порта ввода-вывода. Функции inport и inportb возвращают прочитанное из порта значение, а в функциях outport и outportb записываемое в порт значение задается аргументом val. Функции inportb и outportb работают с однобайтными, а inport и outport – с двухбайтными портами.

Описание используемых программных средств

Программа comport предназначена для работы с портами ввода/ вывода адаптера последовательного интерфейса. Работает под управлением ОС DOS. При запуске в качестве параметра необходимо указать адаптер последовательного интерфейса, не занятый какими-либо устройствами (com1 или com2).

Порядок выполнения работы

1. Загрузить ОС DOS и программу «comport» на компьютере, указав в качестве параметра адаптер последовательного интерфейса, не занятый какими-либо устройствами (com1 или com2).

2. Изучить назначение регистров адаптера последовательного интерфейса.

3. Установить скорость обмена равной 9 600 бод.

4. Инициализировать управляющий регистр по контролю на четность, числу стоп-битов, числу информационных битов.

5. Установить диагностический режим работы, позволяющий выходной сигнал адаптера посылать на его вход. Для этого необходимо установить в единицу 4 бит регистра управления модемом. В процессе проведения работы необходимо следить, чтобы значение этого бита не изменялось.

6. Записать в регистр данных любое число (например, 10101010). Проанализировать содержимое регистра состояния линии на предмет наличия ошибок и сбоев в линии, проверить, принят ли символ адаптером. Затем считать из буфера приемника компьютера принятое число, проверить правильность переданного символа.

7. Определить максимально и минимально возможные скорости передачи данных для используемых адаптеров. Для этого, устанавливая скорости из табл. 7.4 передавать в адаптер символ до тех пор, пока не произойдет ошибка.

8. Провести проверку поведения адаптера при переполнении, для чего переслать в линию сразу 2 байта данных. Программа «comport» выполняет чтение из буфера приемника автоматически, т.е. после записи в буфер передатчика данных некоторого значения из буфера приемника считывается предыдущий полученный символ, и переполнения не происходит.

9. Исследовать основные возможности адаптера по обработке прерываний. Для этого записать в регистр управления прерываниями «00001111» для разрешения всех прерываний. Затем переслать в регистр данных символ и проанализировать содержимое регистра идентификации прерывания, и регистр состояния линии.

10. Проверить возможности по определению обрыва линии, для чего установить бит 6 управляющего регистра (имитации обрыва) в «1». Затем переслать символ и проанализировать содержимое регистра идентификации прерывания и регистра состояния линии.

11. Ознакомиться с основными средствами адаптера для работы с модемом. Для этого поочередно устанавливать в «1» биты 0 – 3 регистра управления модемом и анализировать содержимое регистра идентификации прерывания, регистра состояния модема и регистра состояния линии.

12. Написать и испытать программу на языке Си или Ассемблере, выполняющую следующие действия: инициализацию адаптера для переда-

чи символов в 8(7, 6, 5) бит со скоростью в 9600(2400, 3600, 7200) бод с использованием 1(2) стоп-битов и контролем на четность (нечетность) и осуществляющую передачу и прием данных.

Контрольные вопросы

1. Опишите порядок последовательной передачи данных.
2. Опишите аппаратную реализацию интерфейса RS-232C.
3. Опишите назначение сигналов интерфейса RS-232C.
4. Опишите технические характеристики интерфейса RS-232C.
5. Опишите порядок инициализации асинхронного адаптера для приема/ передачи данных.

Содержание отчета

Отчет должен содержать:

1. титульный лист;
2. тему и цель лабораторной работы;
3. задание на лабораторную работу;
4. результаты и выводы по каждому выполненному пункту;
5. описание алгоритма программы (блок-схема или текстовое описание);
6. прокомментированный листинг программы;
7. выводы по результатам работы.

Литература

1. Кулаков, В. Программирование на аппаратном уровне: спец. справочник / В. Кулаков. – 2-е изд. – СПб. : Питер, 2003.
2. Деревянко, А. С. Системное программное обеспечение персональных ЭВМ: учеб. пособие / А. С. Деревянко. – Харьков : ХГПУ, 1994.
3. Нортон, П. Программно-аппаратная организация компьютера IBM PC / П. Нортон; пер. с англ. С. Писарева, Б. Шура. – Киев, 1987.
4. Журден, Р. Справочник программиста на персональном компьютере фирмы IBM / Р. Журден. – P.-Prentice-Hall Press, 1986.

Лабораторная работа № 8

МАТРИЧНЫЕ ПРИНТЕРЫ. РАСШИРЕННЫЕ ВОЗМОЖНОСТИ ПЕЧАТИ

Цель работы – изучить расширенные возможности печати матричных принтеров.

Теоретические сведения

Принтер дает широкий выбор эффектов при печати – от применения числа знаков на дюйм до использования специальных эффектов или выделения выбранных символов или фраз. Эта лабораторная работа показывает дополнительные возможности печати, которые можно выбрать с помощью программного обеспечения.

Для установки различных спецификаций, относящихся к формату страницы, стилю шрифта и т.п., на принтер посылаются специальные управляющие коды. Эти коды посылаются на принтер, как и любые другие данные. Некоторые из них это простые однобайтные коды из числа первых 32-х набора кодов ASCII. Эти управляющие коды инициируют такие простые действия принтера, как перевод строки или перевод формата (прогон страницы). Однако большинство спецификаций печати устанавливается посылкой ESC-последовательностей, в которых один или более кодовых байтов следует за символом ESC, код которого ASCII 27. Начальный код ESC информирует принтер, что символ(ы), который следует за ним следует интерпретировать как команду, а не как данные. Такие ESC-последовательности обычно не имеют символа-ограничителя, поскольку принтер «знает» длину каждой последовательности. Только в некоторых случаях, когда последовательность может иметь разную длину, требуется ограничивающий символ, в качестве которого всегда используется код ASCII 0.

Почти во всех случаях спецификации, установленные этими кодами действуют до тех пор, пока они не будут явно отменены. Как только будет получен код, например, подчеркивания, то оно будет осуществляться до тех пор, пока не будет послан код отмены подчеркивания. Буфер принтера может быть очищен без отмены установленных спецификаций. Но если произошла ошибка на принтере, и принтер был выключен и включен, то необходимо снова устанавливать все спецификации.

Большинство кодов устанавливающих спецификации принтера перемешаны с данными, на которые они действуют. Например, данные для слова, которое должно быть выделено жирным шрифтом, должны предваряться ESC-последовательностью, включающей жирный шрифт, и завершаться ESC-последовательностью, выключающей его. Поскольку универсальный стандарт на эти коды отсутствует, то печать с использованием мощных возможностей требует, чтобы для каждого поддерживаемого принтера были написаны драйверы. Каждый драйвер преобразует инструкции, генерируемые процедурой печати, в протокол, используемый данным принтером.

Обсуждения и примеры последующих страниц в основном относятся к графическому принтеру IBM. Коды, используемые этим принтером, настолько же «стандартны», насколько и любой другой протокол. В большей степени это связано с тем, что этот протокол используется в эпсоновских принтерах (первые принтеры для IBM PC были фирмы Epson), которые составляют треть всех используемых принтеров. Хотя приведенная далее информация может быть неприменима ко всем принтерам, но большинство общих принципов применимо.

Средства BIOS для работы с принтером. BIOS использует для работы с принтером функции 0, 1, 2 прерывания INT 17h.

Функция 00h предназначена для печати одного символа:

На входе:

AH = 00h;

AL = ASCII-код символа для печати;

DX = номер принтера: 0, 1 или 2.

На выходе:

AH = слово состояния принтера.

Эта функция выводит на принтер один символ, заданный в регистре AL. В регистр DX необходимо записать номер используемого принтера, для LPT1 это 0, для LPT2 – 1 и т. д. После выполнения прерывания регистр AH будет содержать слово состояния, имеющее следующий формат (табл. 8.1).

Вызвав функцию 0 прерывания INT 17h, программа должна проверить отдельные биты слова состояния и убедиться в том, что вывод байта произошел без ошибок. Наиболее часто оператор забывает перевести принтер в состояние ONLINE, либо вставить бумагу, либо вообще включить принтер. В этом случае целесообразно напомнить оператору о необходимости выполнения этих действий и затем повторить печать символа.

Таблица 8.1

Слово состояния принтера

Биты	Значение
0	=1 – таймаут слишком большая задержка при выполнении операции печати, возможно, что принтер неисправен
1	не используется
2	не используется
3	ошибка ввода/ вывода;
4	=1 – принтер выбран для работы; =0 - принтер в состоянии offline;
5	конец бумаги
6	=1 – подтверждение
7	=1 – принтер готов; = 0 – принтер занят

Если принтер неисправен, программа должна предоставить оператору возможность отменить печать текста. Бит 1 байта состояния – таймаут. Если принтер находится в состоянии OFFLINE, функция 0 прерывания INT 17h ожидает некоторое время готовности принтера, после чего, если принтер так и не перешел в состояние готовности, устанавливает бит 1 в байте состояния. Область данных BIOS по адресу 0000h:0478h содержит четыре байта, которые используются в качестве счетчиков времени при ожидании готовности принтера.

Прерывание INT 17h имеет еще две функции, выполняющие инициализацию принтера и получающую текущее состояние принтера.

```
//вывод символа на стандартный принтер с помощью
//функции 00h   прерывания BIOS INT 17h
//вывод символа на стандартный принтер с помощью
//функции 00h   прерывания BIOS INT 17h
union REGS rg;
int chr;
rg.h.ah = 0;
rg.h.al = chr;
rg.x.dx = 0;
int86(0x17, &rg, &rg);
```

Функция 01h инициализирует принтер:

На входе:

AH = 01h;

DX = номер принтера: 0, 1 или 2.

На выходе:

AH = слово состояния принтера.

Эта функция выполняет аппаратный сброс принтера. Если в принтер загружен какой-либо шрифт (например, кириллица), то после сброса за-

грузку шрифта придется выполнять заново. Поэтому не следует выполнять сброс принтера, если это действительно не требуется. Обычно принтер приходится сбрасывать либо перед настройкой его на заданный режим работы, которая выполняется один раз, либо при изменении этого режима.

Слово состояния принтера может быть получено с помощью *функции 02h:*

На входе:

AH = 02h;

DX = номер принтера: 0, 1 или 2.

На выходе:

AH = слово состояния принтера.

Эту функцию удобно использовать перед началом печати для определения готовности принтера к работе.

Средства MS-DOS для работы с принтером. Для печати символа на стандартном печатающем устройстве LPT1 (PRN) можно использовать функцию 05h прерывания MS-DOS INT 21h:

На входе:

AH = 05h;

DL = ASCII-код символа для печати.

На выходе:

AH = слово состояния принтера.

```
//вывод символа на стандартный принтер с помощью
//функции 05h   прерывания MS-DOS INT 21h
union REGS rg;
int chr;
rg.h.ah = 5;
rg.h.dl = chr;
int86(0x21, &rg, &rg);
```

Функция 05h прерывания INT 21h не возвращает состояния принтера при ошибке ввода/ вывода. Вместо этого вызывается стандартный обработчик критических ошибок MS-DOS, который выводит на экран сообщение:

Write fault error writing device PRN

Abort, Retry, Ignore, Fail?

При ответе Retry (нажатие клавиши «R») MS-DOS выполнит попытку повторить печать символа. Если ответить Abort (нажав клавишу «A»), MS-DOS завершит работу программы. Если не устраивают действия, выполняемые стандартным обработчиком критических ошибок MS-DOS, можно определить собственный обработчик.

Набор команд EPSON. Фирма EPSON разработала для своих матричных принтеров набор команд ESC/P, фактически ставший впоследствии международным стандартом – фирмы, выпускающие матричные принтеры, в обязательном порядке включают в свои изделия поддержку набора команд ESC/P. В набор команд ESC/P входят команды для печати в текстовом режиме и режиме битового образа. В усовершенствованный вариант этого командного языка, получивший название ESC/P2, были включены также команды для печати в растровом режиме. На основе ESC/P2 в свою очередь был разработан набор команд для струйных принтеров «Epson raster», специально ориентированный на использование растрового режима – команды для печати в текстовом режиме и режиме битового образа из него изъяты, зато добавлены новые растровые команды.

Управление печатающей головкой и перемещением бумаги. Имеется ряд команд для управления печатающей головкой и перемещением бумаги.

Команда ESC @ (1Bh 40h) – инициализация принтера. Данная команда производит сброс принтера в исходное состояние.

Команда *07h* – генерация звукового сигнала. Если послать этот байт, принтер издаст звуковой сигнал. Сигнал удобно использовать для привлечения внимания оператора, например, когда кончилась бумага.

Команда *0Dh* – возврат каретки. Распечатываются все символы из буфера принтера, затем каретка (печатающая головка) возвращается к началу строки.

Команда *0Ah* – перевод строки. Когда этот символ посылается на принтер, все символы, находящиеся во внутреннем буфере принтера, распечатываются, затем каретка возвращается к началу строки и происходит подача листа вперед на одну строку.

Команда *0Ch* – перевод страницы. Принтер распечатывает все символы, находившиеся в буфере, затем выполняет прогон одного листа бумаги.

Команда *ESC C n (1Bh 43h n)* – установить длину листа бумаги в строках. Команда устанавливает длину листа бумаги, равной n строкам. Допустимые значения параметра *n* лежат в пределах 1...127 строк.

Команда *ESC N n (1Bh 4Eh n)* – установить режим пропуска перфорации. В этой команде параметр n – это количество строк, пропускаемых принтером между последней строкой страницы и первой строкой следующей страницы. Значение n должно находиться в пределах 1...127 строк.

Команда *ESC O (1Bh 4Fh)* – отмена режима пропуска перфорации. Эта команда отменяет режим пропуска перфорации, установленный командой «ESC N n».

Команда *ESC 0 (1Bh 30h)* – выбор межстрочного интервала, равного 1/8 дюйма. Расстояние между текстовыми строками устанавливается равным 1/8 дюйма.

Команда *ESC 2 (1Bh 32h)* – выбор межстрочного интервала, равного 1/6 дюйма. Расстояние между текстовыми строками устанавливается равным 1/6 дюйма. Это значение используется по умолчанию при включении питания принтера.

Команда *ESC 3 n (1Bh 32h***) –** выбор межстрочного интервала, равного n/180 дюйма. Расстояние между текстовыми строками устанавливается равным n/180 дюймов. Значение n должно находиться в пределах 0...255.

Команда *ESC + n (1Bh 2Bh n)* – выбор межстрочного интервала, равного n/360 дюйма. Расстояние между текстовыми строками устанавливается равным n/360 дюймов. Значение n должно находиться в пределах 0...255.

Команда *ESC A n (1Bh 41h n)* – выбор межстрочного интервала, равного n/60 дюйма. Расстояние между текстовыми строками устанавливается равным n/60 дюймов. Значение n должно находиться в пределах 0...85.

Команда *ESC J n (1Bh 4Ah n)* – проброс бумаги на расстояние n/180 дюймов. Бумага продвигается вперед на расстояние, равное n/180 дюймов. Команда выполняется немедленно и не вызывает перемещений печатающей головки.

Команда *11 (0Bh)* – вертикальная табуляция. Бумага продвигается до следующего символа табуляции в канале, выбранном командой «ESC /». Если не выбран никакой канал, по умолчанию используется нулевой. Если вертикальная табуляция не установлена, бумага продвигается вперед на одну строку.

Команда *ESC B n1 n2 ... 0 (1Bh 42h n1 n2 ... 00h)* – установка вертикальной табуляции. Команда позволяет задать до 16 положений для вертикальной табуляции. Параметры n1, n2, ... задают позиции для табуляции. Они должны указываться в порядке возрастания. Последний параметр всегда должен быть равен 0 – это признак конца последовательности параметров. Табуляция устанавливается в нулевом канале.

Команда *ESC b c n1 n2 ... 0 (1Bh 62h c 1n n2 ... 00h)* **–** установка вертикальной табуляции в канале. Команда аналогична команде «ESC B», за исключением того, что необходимо указывать параметр c – номер выбираемого канала для вертикальной табуляции. Значение параметра c должно находиться в пределах 0...7.

Команда *ESC / c (1Bh 2Fh c)* – выбор канала для вертикальной табуляции. Команда выбирает канал c для работы с командами вертикальной табуляции. Значение параметра c должно находиться в пределах 0...7.

Команда *ESC l n (1Bh 6Ch n)* **–** установка левой границы. Устанавливается левая граница листа. В левой части листа оставляется n пустых столбцов символов текущей ширины. Если команда выдается для пропорционального набора символов, в качестве ширины для установки левой границы берется значение 10 символов на дюйм (10 pitch).

Команда *ESC Q n (1Bh 51h n)* – установка правой границы. Устанавливается правая граница листа. В правой части листа оставляется n пустых столбцов символов текущей ширины. Если команда выдается для пропорционального набора символов, в качестве ширины для установки правой границы берется значение 10 символов на дюйм (10 pitch).

Команда *ESC $ n1 n2 (1Bh 24h n1 n2)* – установка абсолютной позиции для печати. Команда задает расстояние от левой границы листа до того места, откуда будет продолжена печать символов. Для вычисления расстояния используется следующая формула: n1 + (n2 * 256). Расстояние задается в единицах, эквивалентных 1/60 доли дюйма.

Команда *ESC \ n1 n2(1Bh 5Ch n1 n2)* – установка относительной позиции для печати. Команда задает расстояние от текущей позиции печатающей головки до того места, откуда будет продолжена печать символов. Для вычисления расстояния используется следующая формула: n1 + (n2 * 256). Расстояние задается в единицах, эквивалентных 1/120 доли дюйма для чернового режима и 1/180 доли дюйма для качественного и пропорционального.

Команда *9(09h)* – горизонтальная табуляция. Печатающая головка продвигается до следующего символа горизонтальной табуляции. По умолчанию для одного символа табуляции используется интервал в 8 символов текущего размера.

Команда *ESC D n1 n2 ... 0 (1Bh 44h n1 n2 ... 00h)* – установка горизонтальной табуляции. Команда позволяет задать до 32 положений для горизонтальной табуляции. Параметры n1, n2, ... задают позиции для табуляции. Они должны указываться в порядке возрастания. Последний параметр всегда должен быть равен 0 – это признак конца последовательности параметров. Команда «ESC D 0» сбрасывает все позиции горизонтальной табуляции.

Расширение возможностей печати текста. Принтер дает широкий выбор эффектов при печати – от изменения числа знаков на дюйм до использования специальных эффектов или выделения выбранных символов или фраз.

Команда *ESC x n (1Bh 78h n)* – выбор черновой или качественной печати. Параметр n определяет режим печати следующим образом: 0 – черновой режим печати; 1 – качественный (LQ) режим печати.

Команда *ESC k n (1Bh 6Bh n)* – выбор стиля печати. Команда действительна только для режима качественной печати. Параметр n задает стиль: 0 – Roman; 1 – Sans Serif; 2 – Courier; 3 – Prestige; 4 – Script; 5 – OCR-B; 6 – OCR-A.

Команда *ESC ! n (1Bh 21h n)* – выбор режима работы принтера. Команда позволяет задать комбинацию различных режимов работы принтера. Можно по отдельности задавать размер символов (10 или 12 символов на дюйм), набор символов (пропорциональный, сжатый, выделенный и т. д.). Отдельные биты байта параметра n задают режим работы принтера следующим образом (табл. 8.2).

Таблица 8.2

Назначение битов параметра n

Биты	Назначение
0	=0 – размер символа 10 pitch; =1 – размер символа 12 pitch
1	=1 – пропорциональный шрифт
2	=1 – сжатый шрифт
3	=1 – выделенный шрифт
4	=1 – использование двух проходов
5	=1 – двойная высота
6	=1 – использование курсива
7	=1 – использование подчеркивания

Команда *ESC P (1Bh 50h)* – выбор размера символа в 10 pitch. Эта команда задает размер символа, равный 10 pitch, или 10 символов на дюйм. Такой размер устанавливается по умолчанию при инициализации принтера.

Команда *ESC M (1Bh 4Dh)* – выбор размера символа в 12 pitch. Эта команда задает размер символа, равный 12 pitch, или 12 символов на дюйм.

Команда *ESC g (1Bh 67h)* – выбор размера символа в 15 pitch. Эта команда задает размер символа, равный 15 pitch, или 15 символов на дюйм. Этот режим несовместим с режимом сжатой печати.

Команда *ESC p n (1Bh 70h n)* – включение/выключение пропорционального режима. В пропорциональном режиме разные символы имеют различную ширину, что благоприятно сказывается на читаемости текста. Например, букава «i» уже, чем «W». Параметр n может принимать следующие значения: 0 – выключение пропорционального режима; 1 – включение пропорционального режима.

Команда *15(0Fh)* – выбор режима сжатой печати (SI). В этом режиме символы имеют примерно на 60 процентов меньшую ширину, чем в нормальном режиме. Режим сжатой печати не совместим с пропорциональным режимом.

Команда *ESC SI (1Bh 0Fh)* – выбор режима сжатой печати. Команда полностью аналогична предыдущей команде «SI».

Команда *18(12h)* – отмена режима сжатой печати (DC2). Отменяется режим сжатой печати, установленный ранее командами ESC SI или SI.

Команда *14 (0Eh)* – печать с двойной шириной (SO). В этом режиме ширина каждого распечатываемого символа увеличивается в два раза. Режим печати с двойной шириной отменяется командой возврата каретки или командой DC4.

Команда *ESC SO(1Bh 0Eh)* – печать с двойной шириной. Команда полностью аналогична предыдущей команде «SO».

Команда *ESC W n (1Bh 77h n)* – включение/выключение режима печати с двойной высотой. В режиме печати с двойной высотой высота каждого распечатываемого символа увеличивается в два раза. Возможные значения параметра n: 0 – выключение режима печати с двойной высотой; 1 – включение режима печати с двойной высотой.

Команда *20(14h)* – отмена режима печати с двойной шириной (DC4). Команда отменяет действие команд «ESC SO» или «SO», но не действует, если режим печати с двойной шириной задан командами «ESC W» или «ESC!».

Команда *ESC W n (1Bh 57h n)* – включение/выключение режима печати с двойной шириной. В режиме печати с двойной шириной ширина каждого распечатываемого символа увеличивается в два раза. Возможные значения параметра n: 0 – выключение режима печати с двойной шириной; 1 – включение режима печати с двойной шириной.

Команда *ESC E (1Bh 45h)* – установка режима печати с выделением. Распечатываемые символы выглядят «толще» за счет того, что каждая точка печатается дважды.

Команда *ESC F (1Bh 46h)* – отмена режима печати с выделением. Команда отменяет действие команды «ESC E».

Команда *ESC G (1Bh 47h)* – установка режима двойной печати. В режиме двойной печати каждая строка печатается дважды, поэтому текст выглядит ярче. Скорость печати уменьшается в два раза.

Команда *ESC H (1Bh 48h)* – отмена режима двойной печати. Команда отменяет действие команды «ESC G».

Команда *ESC S 0 (1Bh 53h 00h)* – печать верхнего индекса. Символы распечатываются выше обычного уровня, занимая верхние две трети сетки.

Команда *ESC S 1 (1Bh 53h 01h)* – печать нижнего индекса. Символы распечатываются ниже обычного уровня, занимая нижние две трети сетки.

Команда *ESC T (1Bh 54h)* – отмена печати верхнего или нижнего индекса. Команда отменяет действие любой из команд, задающих режим печати индекса – «ESC S 0» или «ESC S 1».

Команда *ESC - n (1Bh 2Dh n)* – включение/выключение режима подчеркивания. В зависимости от значения параметра n все символы (и пробелы тоже) печатаются с подчеркиванием или без подчеркивания: 0 – выключение режима подчеркивания; 1 – включение режима подчеркивания.

Команда *ESC q n (1Bh 71h n)* – выбор стиля распечатываемых символов. В зависимости от значения параметра n все символы, кроме имеющих коды от B0h до DFh и символа с кодом F5h распечатываются с использованием следующих стилей: 0 – обычный стиль; 1 – контурное (outline) начертание символов; 2 – использование тени (стиль shadow); 3 – комбинация контурного начертания и тени.

Команда *ESC a n (1Bh 61h n)* – выравнивание для качественного (LQ) набора символов. Параметр n может принимать следующие значения: 0 – выравнивание влево; 1 – выравнивание по центру; 2 – выравнивание вправо; 3 – полное выравнивание. По умолчанию при инициализации принтера выбирается режим выравнивания влево. Полное выравнивание выполняется после заполнения буфера печати. При этом распечатываемый текст может содержать символы горизонтальной табуляции HT и возврата на одну позицию BS только тогда, когда задан режим выравнивания влево (n=0). Если используется полное выравнивание, параграфы текста не должны содержать символы возврата каретки.

Команда *ESC SP n (1Bh 20h n)* – выбор расстояния между символами. Команда позволяет увеличить расстояние между символами по сравнению с тем, которое было задано в сетке при разработке начертания символов. Параметр n, значение которого должно лежать в пределах 0...127, задает количество точек, добавляемых справа к каждому символу. Одна точка соответствует 1/120 дюйма в черновом режиме и 1/180 дюйма в качественном и пропорциональном режимах.

Графический режим

Принтер всегда находится в текстовом режиме, до тех пор, пока он специально не переведен в графический режим. Команда, устанавливающая графический режим, должна сообщать какое число байтов графических данных будет передано (но не больше одной строки) и после того, как это число байтов будет интерпретировано как графическое изображение, принтер вернется в текстовый режим. По этой причине нет команды, которая переводит принтер в текстовый режим.

Печатающая головка принтера имеет девять иголок. При ее перемещении вдоль страницы электрические импульсы вызывают быстрое перемещение иголок. Всякий раз, когда иголка перемещается, она ударяет по ленте с краской и прижимает ее к бумаге, на которой остается маленькая точка. При движении головки вдоль бумаги иголки раз за разом перемещаются, формируя буквы, цифры или символы. Печатающая головка принтера может печатать графику и дополнение к текстам, потому что графическое изображение формируется принтером таким же способом, как печатаются иллюстрации в газетах и журналах. Принтер также формирует такие изображения рисунком из точек с плотностью 240 точек на дюйм по горизонтали и 72 точки по вертикали. Поэтому изображения, печатаемые принтером, могут иметь такие же мелкие детали, как и на рисунке. В своем основном режиме графики принтер печатает одну колонку точек для получаемого кода и при этом используется только восемь первых иголок из девяти. Поэтому программа графики должна посылать коды для точечных рисунков по одному для каждой колонки в строке. В каждой из этих колонок печатающая головка печатает определенный рисунок из точек. Для печати рисунков выше, чем восемь точек, печатающая головка делает более одного прохода. Принтер печатает одну строку, затем перемещает бумагу и печатает следующую, также как он делает это с текстом. Чтобы печатающая головка не делала пропусков между строками графики, как между строками текста, расстояние между строками должно изменяться до устранения пропуска между строками. При изменении расстояния между строками принтер может печатать графические изображения с мелкими деталями так, что не будет видно, что они состоят из отдельных линий, каждая высотой не более 8/72 дюйма.

Чтобы сообщить принтеру, какие иголки должны перемещаться в каждой колонке, необходимо сначала разделить каждую вертикальную колонку на три секции по восемь колонок в каждой и рассматривать каждую секцию отдельно. Так как имеется 256 возможных комбинаций восьми иголок, вам необходима система нумерации, которая показана на рис. 8.1.

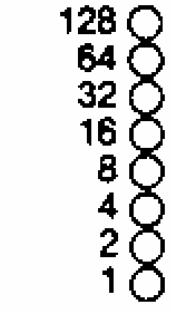

Рис. 1. **Система нумерации иголок**

Для перемещения любой иголки, необходимо посылать на принтер ее номер. Для перемещения одновременно более одной иголки, необходимо сложить номера иголок и послать сумму принтеру. Поэтому при этой маркировке иголок,

перемещаете верхнюю иголку, посылая число 128. Для перемещения нижней иголки необходимо послать 1. Если необходимо переместить только верхнюю и нижнюю иголки, то нужно сложить 128 и 1 и посылать 129.

Складывая соответствующие маркировочные номера вместе, можно перемешать любую комбинацию иголок. На рис. 8.2 показаны три примера вычисления чисел, которые будут создавать конкретный рисунок иголок.

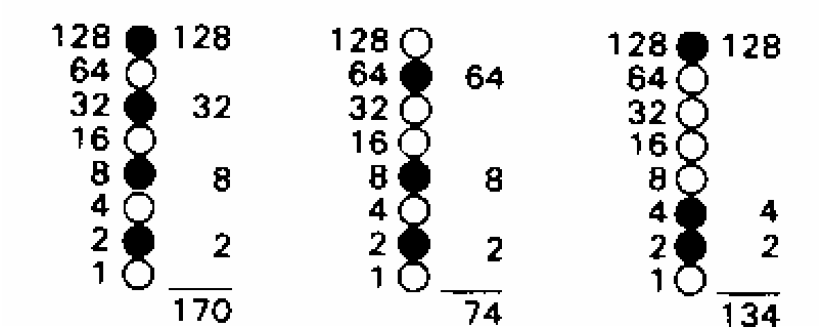

Рис. 8.2. **Задание конкретного рисунка иголок**

При этой системе нумерации любая комбинация сложения для восьми иголок представляется без дублирования десятичным числом от 0 до 255.

Команда *ESC K n1 n2 (1Bh 4Bh n1 n2)* – выбор графического режима с одинарной плотностью. Команда устанавливает графический 8-битовый режим одинарной плотности. Общее количество столбцов в графической строке при этом составит n1 + (n2 * 256).

Команда *ESC L n1 n2 (1Bh 4Ch n1 n2)* – выбор графического режима с двойной плотностью. Команда устанавливает графический 8-битовый режим двойной плотности, печать будет выполняться с низкой скоростью. Общее количество столбцов в графической строке при этом составит n1 + (n2 * 256).

Команда *ESC Y n1 n2 (1Bh 59h n1 n2)* – выбор скоростного графического режима с двойной плотностью. Команда устанавливает графический 8-битовый режим двойной плотности, печать будет выполняться с высокой скоростью. Общее количество столбцов в графической строке при этом составит n1 + (n2 * 256).

Команда *ESC Z n1 n2 (1Bh 5Ah n1 n2)* – выбор графического режима с учетверенной плотностью. Команда устанавливает графический 8-битовый режим учетверенной плотности. Общее количество столбцов в графической строке при этом составит n1 + (n2 * 256).

Команда *ESC ? s n (1Bh 3Fh s n)* – переназначение графических режимов. Команда позволяет заменить один графический режим на другой.

Параметр s – это символ (K, L, Y, Z), который назначается графическому режиму, заданному параметром n (0...6).

Команда *ESC * (1Bh 2Ah m n1 n2)* – печать в графическом режиме. Полный формат команды графической печати: *ESC * m n1 n2 data*. В этой команде параметр m задает режим печати:

0 – одинарная плотность, 60 точек на дюйм,8-битовая графика;

1 – двойная плотность, 120 точек на дюйм, 8-битовая графика;

2 – двойная плотность, печать с высокой скоростью, 120 точек на дюйм, 8-битовая графика;

3 – учетверенная плотность, 240 точек на дюйм,8-битовая графика;

4 – режим CRT I, плотность 80 точек на дюйм, 8-битовая графика;

6 – режим CRT II, плотность 90 точек на дюйм, 8-битовая графика;

32 – одинарная плотность, 60 точек на дюйм, 24-битовая графика;

33 – двойная плотность, 120 точек на дюйм, 24-битовая графика;

38 – режим CRT III, плотность 90 точек на дюйм, 24-битовая графика;

39 – тройная плотность, 180 точек на дюйм, 24-битовая графика; 40-шестикратное увеличение плотности, 360 точек на дюйм, 24-битовая графика.

Параметры n1 и n2 определяют длину печатаемой графической строки в точках. При определении длины графической строки необходимо учитывать, что в режиме одинарной плотности на строке длиной 8 дюймов можно разместить 480 точек, в режиме учетверенной плотности – около 2000.

Т. к. передача данных в принтер выполняется по байтам, для представления длины строки приходится использовать два байта информации. Для вычисления параметров n1 и n2 можно использоваться следующей схемой:

- разделить длину строки на 256, целочисленный результат деления использовать в качестве параметра n2;
- остаток от деления использовать в качестве n1.

Например, пусть надо распечатать строку из 1234 точек. Тогда параметр n2 будет равен 1234 / 256 = 4. Остаток от деления составит 1234 – 256 * 4 = 210. Это и есть параметр n1. Проверка: 4 * 256 + 210 = 1234.

Команда должна всегда содержать два параметра, даже если параметр n2 получился равным нулю. Вслед за параметрами n1 и n2 должны следовать байты графических данных, предназначенные для печати. Должно быть передано точно n2 * 256 + n1 байтов. Если будет передано меньше графических данных, чем это было определено в команде ESC *, следующие вводимые в принтер команды или данные будут интерпрети-

роваться как графические данные. Если будет передано больше графических данных, чем нужно, лишние данные будут напечатаны как обычный текст.

Для представления одного восьми точечного столбца графической строки используется один байт данных, причем верхней точке в столбце соответствует старший разряд байта, а нижней – младший (рис. 8.3):

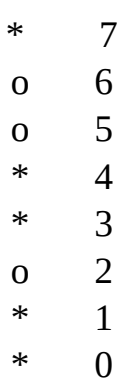

Рис. 8.3. **Восьмиточечный столбец графической строки**

Этому столбцу соответствует байт 10011011b или 9Bh. На рис. «*» означает точку в столбце графической строки, «o» – отсутствие точки. Необходимо подготовить, таким образом, байты для всех столбцов, входящих в распечатываемую графическую строку.

24-игольчатый принтер может работать в описанном выше 8-бито-вом графическом режиме. Это сделано для обеспечения совместимости со старым программным обеспечением, рассчитанным на принтеры серий FX, RX, LX и EX. Однако все возможности этого принтера раскрываются только при использовании всех его 24 иголок. В этом случае графическое изображение печатается отдельными строчками, высота которых составляет 24 точки. При этом для представления одного столбца графической строки требуется три байта данных. Каждый байт должен готовиться отдельно, при этом можно считать, что 24-битовая графическая строка состоит из трех 8-битовых.

Формат команды графической печати для этого принтера расширен по сравнению с описанным выше:

*ESC * m n1 n2 data* – печать в графическом режиме. В этой команде m, как и раньше, задает режим печати. Однако для этого параметра определено больше значений:

0 – одинарная плотность, 60 точек на дюйм, 8-битовая графика;

1 – двойная плотность, 120 точек на дюйм, 8-битовая графика;

2 – двойная плотность, печать с высокой скоростью, 120 точек на дюйм, 8-битовая графика;

3 – учетверенная плотность, 240 точек на дюйм, 8-битовая графика;

4 – режим CRT I, плотность 80 точек на дюйм, 8-битовая графика,

6 – режим CRT II, плотность 90 точек на дюйм, 8-битовая графика;

32 – одинарная плотность, 60 точек на дюйм,24-битовая графика;

33 – двойная плотность, 120 точек на дюйм, 24-битовая графика;

38 – режим CRT III, плотность 90 точек на дюйм, 24-битовая графика;

39 – тройная плотность, 180 точек на дюйм, 24-битовая графика;

40 – шестикратное увеличение плотности, 360 точек на дюйм, 24-битовая графика.

Для команды 24-битового графического вывода требуется массив графических данных в три раза больше по размеру, чем для 8-битовой команды.

```
//пример программы, выводящей на принтер в режиме 8-битовой
//графики строку, состоящую из 40 столбцов

#include <dos.h>
#include <stdio.h>

union REGS rg;

int main() {

int i;

// Переводим строку
printchar(0x0d);
printchar(0x0a);

// Выдаем на принтер команду графической
// печати (ESC "*" m n1 n2 data)
printchar(27);
printchar('*');

// Выбираем режим 0 - 8-битовая графика,
// одинарная плотность
printchar(0);

// Задаем длину графической строки:
//    n1 = 40; n2 = 0
printchar(40);
printchar(0);

// Выводим в цикле 40 раз байт DBh
 for(i=0; i<40; i++)
  printchar(0xdb);

// Переводим строку
 printchar(0x0d);
 printchar(0x0a);
}

// -----------------------------------
// Эта функция выводит один символ
```

```
// на стандартный принтер (LPT1)
// ----------------------------------

int printchar(int chr) {

// Вызываем функцию 5 прерывания INT 21h -
// распечатка символа на принтере.
 rg.h.ah = 5;
 rg.h.dl = chr;
int86(0x21, &rg, &rg);
}

//Аналогичная программа, использующая 24-битовую графику
#include <dos.h>
#include <stdio.h>

union REGS rg;

int main() {

int i;

// Переводим строку
printchar(0x0d);
printchar(0x0a);

// Выдаем на принтер команду графической
// печати (ESC "*" m n1 n2 data)
printchar(27);
printchar('*');

// Выбираем режим 32 - 24-битовая графика,
// одинарная плотность
printchar(32);

// Задаем длину графической строки:
//    n1 = 40; n2 = 0
printchar(40);
printchar(0);

// Выводим в цикле 120 раз байт DBh. Для вывода
// строки из 40 столбцов в 24-битовом режиме
// требуется в три раза больше графических данных,
// чем для 8-битового режима.
for(i=0; i<120; i++)
 printchar(0xdb);

// Переводим строку
printchar(0x0d);
printchar(0x0a);
}
```

Итак, для вывода на принтер сложных графических изображений программа должна сначала подготовить массив данных для построчной 8-битовой или 24-битовой печати. Затем готовый массив можно вывести на принтер, используя несколько команд графического вывода (по одной команде на одну графическую строку).

Разработка своих символов. Для разработки собственных символов используется сетка. В 9-игольчатых принтерах эта сетка имеет 11 столбцов и девять строк (рис. 8.4).

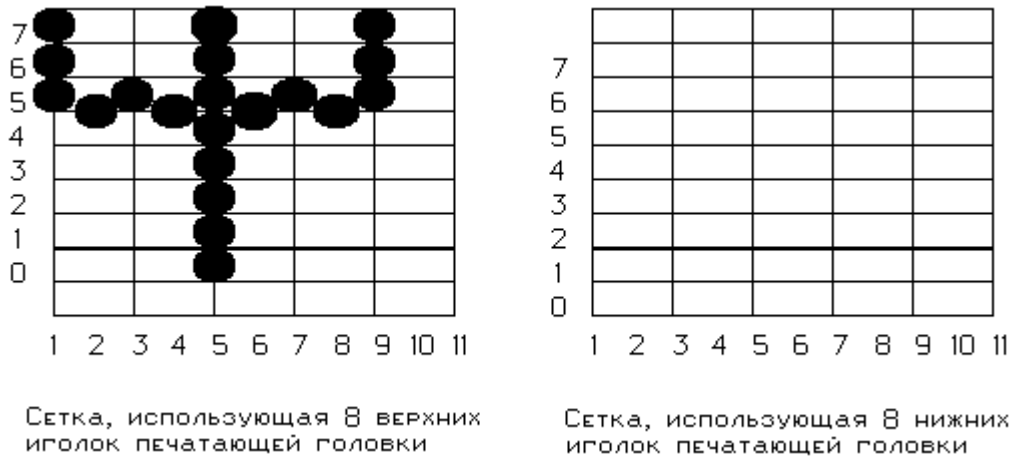

Рис. 8.4. **Сетка для разработки символов 9-игольчатого принтера**

Из девяти строк может использоваться только 8 верхних или 8 нижних (это показано на левом и правом рисунках соответственно). Обычно символ располагается выше утолщенной линии, то есть в строках с номерами от 1 до 7. Исключение составляют такие буквы, как «у», «ц» и т. п. Нижние «хвостики» этих букв должны находиться на строке с номером 0.

Есть ограничение на расположение отдельных точек определяемого символа в строке - эти точки не должны находиться рядом, то есть между точками в строке должна находиться одна свободная ячейка.

Для переопределения символов 9-игольчатый принтер использует команду ESC &:

ESC & 0 n1 n2 a1 d1 d2 ... dn – определить символы. Параметры n1 и n2 задают диапазон кодов ASCII символов, начертание которых необходимо переопределить. Если вы переопределяете только один символ, эти два параметра должны быть одинаковыми. Параметр a1 определяет ширину символа в точках и его положение в сетке (использует ли символ верхние восемь линий, либо нижние восемь линий). Ширина определяемого симво-

ла требуется для печати в пропорциональном режиме, когда место, занимаемое каждой буквой в строке распечатки, зависит от ее ширины. Например, буква «Ш» шире, чем буква «И». Старший бит параметра a1 задает расположение символа в сетке. Если этот бит равен 1, используются восемь верхних линий сетки, если 0 – восемь нижних. Младшие семь битов задают ширину символа и представляют собой число, определяемое по следующей схеме:

- в качестве начального значения для ширины символа необходимо взять число 8;
- для каждого пустого столбца в сетке с правой стороны символа надо вычесть из начального значения единицу;
- для каждого пустого столбца в сетке с левой стороны символа надо прибавить к начальному значению число 16.

Пусть определяемый символ располагается в верхней части сетки (использует восемь верхних строк). Пусть этот символ начинается в третьем столбце и заканчивается в 7 столбце. Тогда десятичное значение параметра a1 вычисляется следующим образом:

a1 = 8 (начальное значение) –
– 2 (два пустых столбца справа) +
+ 32 (два пустых столбца слева) +
+ 128 (старший бит равен 1) = 166

Если ваш символ использует верхние восемь строк сетки, начинается в первом столбце и заканчивается в девятом, в качестве параметра a1 подходит значение 136. При этом символы будут печататься верхними восемью иголками печатающей головки. Для использования нижних восьми иголок и такой же ширины символа задайте значение a1 равное 8.

Параметры d1...dn – образцы столбцов точек для определяемого символа. Их должно быть всегда 11, даже если символ содержит пустые столбцы. Для пустых столбцов в качестве образца надо задать 0.

Для включения определенного программой набора символов в работу необходимо выдать команду *ESC % 0*, для использования набора символов из внутреннего ПЗУ принтера необходимо использовать команду *ESC % 1*.

Для определения начертания в 24-иголочном принтере используется сетка высотой 24 точки – соответственно, по одной точке для каждой иголки в печатающей головке. 24-иголочный принтер использует несколько наборов символов:

- черновой набор символов (Draft);
- качественный набор символов (Letter Quality);
- пропорциональный набор символов (Proportional).

В зависимости от используемого набора символов ширина сетки может быть либо 9 точек (для чернового набора символов), либо 29 точек (для качественного набора символов), либо 37 точек (для пропорционального набора символов). Кроме того, для последних двух наборов столбцы сетки расположены ближе друг к другу, чем для чернового набора.

На рис. 8.5 показаны сетки для чернового и качественного/ пропорционального наборов символов 24-игольчатого принтера.

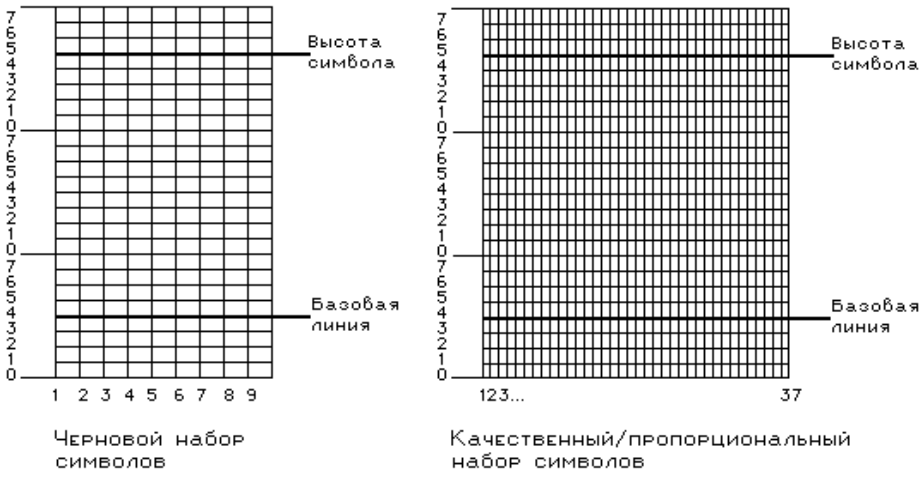

Рис. 8.5. **Сетка для разработки символов 24-игольчатого принтера**

Обычные символы располагаются между жирными линиями. Нижние линии сетки используются для подчеркивания и изображения «хвостиков» таких букв, как «у», «ц» и т. п. Так же, как и для 9-игольчатых принтеров, существует ограничение на расположение точек в узлах сетки: справа и слева от каждой точки должны располагаться пустые позиции.

Формат команды для переопределения символов в 24-игольчатом принтере: *ESC & 0 n1 n2 d0 d1 d2 data* – определить символы.

Параметры n1 и n2 задают диапазон кодов ASCII символов, начертание которых необходимо переопределить. Их назначение такое же, как и для 9-игольчатых принтеров. Если переопределяется только один символ, эти два параметра должны быть одинаковыми. Далее следуют три байта данных, которые задают ширину символа и размер свободного пространства вокруг символа. Параметр d0 задает количество свободных столбцов слева от символа, параметр d2 определяет количество свободных столбцов справа от символа. Параметр d1 определяет ширину символа в столбцах сетки. Изменяя ширину символа и размер свободного пространства вокруг него можно формировать пропорциональные наборы символов. В табл. 8.3 приведены максимальные значения для параметров d0, d1, d2 для различных наборов символов.

Таблица 8.3

Максимальные значения для параметров d0, d1, d2 для различных наборов символов

Набор	d1	d0+d1+d2
Черновой	9	12
Качественный, 10 символов на дюйм	29	36
Качественный, 12 символов на дюйм	23	30
Пропорциональный	37	42

После параметра d2 следует последовательность байтов, описывающих символ, т.е. образец для символа. Для задания одного столбца сетки требуется три байта, поэтому для определения одного символа необходимо задать (d1 * 3) байтов данных.

```
//пример программы, изменяющей начертание символа "@" в 9-
//игольчатом принтере

#include <dos.h>
#include <stdio.h>

main(){

char buffer[] = {

0x1b, '@',   // Сбрасываем принтер в исходное
// состояние.

// Определяем вместо "@" новый символ:
0x1b, '&', 0,
'@', '@', 136,
 32,80,168,84,42,84,168,80,32,0,0,

// Выдаем строку символов, используем начертание,
// заданное в ПЗУ принтера.

'@', '@', '@', '@', '@', 0x0a,

// Используем новое начертание:
0x1b, '%', 1,
'@', '@', '@', '@', '@', 0x0a,

// Возвращаемся опять к старому начертанию:
0x1b, '%', 0,
'@', '@', '@', '@', '@', 0x0a,
'$'  // Признак конца массива данных
};
```

```
        char *p;

// Выводим строку символов на принтер
for(p = buffer; *p != '$'; p++)
 bdos(0x05, *p, 0);
}
```

Описание используемых аппаратных средств

Для выполнения лабораторной работы необходимо использовать матричный принтер, поддерживающий систему команд EPSON ESC/P.

Принтер EPSON LX-1050+ – монохромный матричный притер формата А3. Параметры печати: 9-игольчатая печатающая головка. Скорость печати 200/240cps (10/12cpi) в черновом и 40/48 cps (10/12 cpi) в качественном режимах, максимальная скорость 240 cps (12 cpi). Максимальное разрешение 240x216 dpi. Буфер данных 4кБ. Шумы не более 53 дБ. Функции: 3 встроенных шрифта. Система команд EPSON ESC/P. Интерфейс Centronics. Загрузчик бумаги на 150 листов А3, печать на отдельных листах и перфорированной бумаге. Совместим с MS-DOS и Windows.

Порядок выполнения работы

Согласно одному из следующих вариантов задания написать и отладить программу на языке Си или Ассемблер, работающую с принтером, используя средства BIOS или MS-DOS:

1. Отформатировать и вывести на печатающее устройство текстовый файл в кодах ASCII с количеством символов в строке M и количеством строк на листе N. Текст отформатировать по ширине. В начале каждой страницы вставить колонтитул, содержащий дату, время, имя файла и номер страницы, колонтитулы выделить утолщенным шрифтом.
2. Создать свой символ «сердце» и распечатать его в режиме двойной ширины и двойным проходом.
3. Создать символ  и распечатать его вместо прописных букв «a» и «w».

Контрольные вопросы

1. Какие Вы знаете средства BIOS для работы с принтером?
2. Какие Вы знаете средства MS-DOS для работы с принтером?
3. Опишите принцип печати матричного принтера.
4. Для чего используются ESC-последовательности?
5. Опишите основные возможности принтера при печати текста.
6. Опишите команду графического режима принтера.

Содержание отчета

Отчет должен содержать:

1. титульный лист;
2. тему и цель лабораторной работы;
3. задание на лабораторную работу;
4. описание алгоритма программы (блок-схема или текстовое описание);
5. прокомментированный листинг программы;
6. выводы по результатам работы.

Литература

1. Кулаков, В. Программирование на аппаратном уровне: спец. справочник / В. Кулаков. – 2-е изд. – СПб. : Питер, 2003.
2. Деревянко, А. С. Системное программное обеспечение персональных ЭВМ: учеб. пособие / А. С. Деревянко. – Харьков : ХГПУ, 1994.
3. Нортон, П. Программно-аппаратная организация компьютера IBM PC / П. Нортон; пер. с англ. С. Писарева, Б. Шура. – Киев, 1987.
4. Журден, Р. Справочник программиста на персональном компьютере фирмы IBM / Р. Журден. – P.-Prentice-Hall Press, 1986.