САРОВСКИЙ ГОСУДАРСТВЕННЫЙ
ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет информационных технологий и электроники

Кафедра вычислительной и информационной техники

В.А. Павлов

Система ввода-вывода ПК.

Подсистемы клавиатуры и манипуляторов типа мышь и джойстик.

Учебно-методическое пособие

г. Саров 2008г.

САРОВСКИЙ ГОСУДАРСТВЕННЫЙ
ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет информационных технологий и электроники

Кафедра вычислительной и информационной техники

В.А. Павлов

Система ввода-вывода ПК.

Подсистемы клавиатуры и манипуляторов типа мышь и джойстик.

.

Учебное пособие и практикум

Утверждено:

На заседании кафедры ВИТ

_____ 2008г

Научно-методическим Советом факультета

_____ 2008г

Научно-методическим Советом СарФТИ

_____ 2008г.

г. Саров 2008г.

УДК 681.3.06

П12

Одобрено Научно-методическим советом
Саровского государственного физико-технического
института

Павлов Виктор Александрович
Система ввода-вывода ПК. Подсистемы клавиатуры и манипуляторов типа мышь и джойстик. Учебно-методическое пособие и практикум по циклу "Система ввода-вывода ПК". СарФТИ, Саров, 2008. - 287 с.: ил.

Настоящее учебное пособие является дальнейшим расширением методической поддержки лабораторных практикумов цикла "Система ввода-вывода ПК". В СарФТИ этот цикл реализуется в спецкурсе "Интерфейсы периферийных устройств" и курсе специализации "Адаптеры и контроллеры ЭВМ" в рамках специальности 230101 "Вычислительные машины, комплексы, системы и сети".

В пособии рассмотрены принципы работы и программирования адаптера клавиатуры и PS/2 Mouse, контроллера COM-порта и контроллера игрового порта, электронной клавиатуры, манипуляторов PS/2 Mouse и манипуляторов Serial Mouse. Приводятся программы и программные модули, иллюстрирующие правила программного взаимодействия с перечисленными устройствами ПК. В практикуме приведено описание семи лабораторных работ, посвященных изучению принципов работы и программирования перечисленных выше устройств и исследованию их интерфейсов на физическом и канальном уровнях.

В приложении приведено краткое описание специальных программных средств (отладчика AFD), используемых для отладки и проверки программ, приведенных в пособии и практикуме. Приведено также описание системы команд процессора i8086 и подсистемы аппаратных прерываний ПК.

© В. А. Павлов, 2008г.

СОДЕРЖАНИЕ

Стр.

ВВЕДЕНИЕ -- -- -- -- -- -- -- -- -- 7
1. Подсистема клавиатуры и PS/2 Mouse -- -- -- -- -- -- 8
1.1. Адаптер клавиатуры и манипулятора PS/2 Mouse -- -- -- -- -- 8
1.1.1. Микроконтроллер i8042 -- -- -- -- -- -- -- -- 9
1.1.2. Интерфейс: адаптер клавиатуры - система -- -- -- -- -- -- 11
1.1.3. Интерфейсы: адаптер - клавиатура и адаптер - PS/2 Mouse -- -- -- -- 11
1.1.4. Программно доступные регистры адаптера -- -- -- -- -- -- 12
1.1.4 1. Регистр состояния адаптера -- -- -- -- -- -- -- 12
1.1.4.2. Регистр командного байта адаптера -- -- -- -- -- -- 13
1.1.4.3. Регистр команд -- -- -- -- -- -- -- -- -- 13
1.1.4.4. Регистр данных -- -- -- -- -- -- -- -- -- 13
1.1.5. Команды адаптера-- -- -- -- -- -- -- -- -- 13
1.1.6. Порты ввода и вывода микроконтроллера i8042. -- -- -- -- -- 15
1.1.7. Функционирование адаптера -- -- -- -- -- -- -- 15
1.1.8. Программирование адаптера клавиатуры и PS/2 Mouse -- -- -- -- 16
1.1.8.1. Программирование портов микроконтроллера i8042 -- -- -- -- 17
1.1.8.2. Примеры программирования на Ассемблере -- -- -- -- -- 18
1.1.8.3. Примеры программирования адаптера в среде AFD -- -- -- -- 19
1.1.8.3.1. Программа CKLD0.bin, позволяющая получить общие характеристики команд адаптера 19
1.1.8.3.2. Программа CKLA1C.BIN циклического выполнения команд, возвращающих результаты 21
1.1.8.3.3. Программа CKLA8.BIN циклического выполнения однобайтовых команд -- 22
1.1.8.3.4. Программа CKLAA2.BIN циклического выполнения команды AAh -- -- 23
1.1.8.3.5. Программа CKL60M.BIN циклического выполнения двухбайтовых команд -- 24
1.1.8.3.6. Программа CKLD34.BIN циклического выполнения команд D3h и D4h `-- -- 25
1.1.8.3.7. Программа CKLAFT.BIN циклического выполнения трехбайтовой команды AFh -- 27
1.1.9. Проверка адаптера клавиатуры процедурой POST -- -- -- -- -- 29
Контрольные вопросы -- -- -- -- -- -- -- -- 28
1.2. Клавиатура -- -- -- -- -- -- -- -- 30
1.2.1. Конструктивы 101-клавишной клавиатуры -- -- -- -- -- -- 31
1.2.2. Блок-схема клавиатуры -- -- -- -- -- -- -- -- 33
1.2.3. Схемотехника клавиатуры -- -- -- -- -- -- -- -- 34
1.2.4. Интерфейс клавиатуры -- -- -- -- -- -- -- -- 36
1.2.5. Функционирование клавиатуры -- -- -- -- -- -- -- 37
1.2.5.1. Базовый тест -- -- -- -- -- -- -- -- -- 37
1.2.5.2. Синхросигналы и сигналы данных -- -- -- -- -- -- 37
1.2.5.3. Поток данных -- -- -- -- -- -- -- -- -- 38
1.2.5.4. Обмен данными -- -- -- -- -- -- -- -- -- 38
1.2.5.4.1. Ввод данных с клавиатуры -- -- -- -- -- -- -- 38
1.2.5.4.2. Вывод данных в клавиатуру -- -- -- -- -- -- -- 38
1.2.5.5. Режимы работы клавиатуры -- -- -- -- -- -- -- 41
1.2.5.6. Скан-коды -- -- -- -- -- -- -- -- -- 42
1.2.6. Программирование клавиатуры -- -- -- -- -- -- -- 45
1.2.6.1. Обработка нажатия и отпускания клавиш -- -- -- -- -- -- 46
1.2.6.1.1. Обработка специальных клавиш -- -- -- -- -- -- 46
1.2.6.2. Программное взаимодействие с клавиатурой -- -- -- -- -- 46
1.2.6.2.1. Работа с клавиатурой через порты ввода-вывода-- -- -- -- -- 46
1.2.6.2.2. Команды клавиатуры и их описание -- -- -- -- -- -- 48
1.2.6.2.3. Коды ответов клавиатуры -- -- -- -- -- -- -- 51
1.2.6.3. Работа с клавиатурой через прерывание INT 09h -- -- -- -- -- 51
1.2.6.4. Работа с клавиатурой через прерывание INT 16h (драйвер клавиатуры) -- -- 52
1.2.6.4.1. Функции прерывания INT 16h -- -- -- -- -- -- -- 53
1.2.6.4.2 ASCII-коды -- -- -- -- -- -- -- -- -- 54
1.2.6.5. Функции DOS обработки клавиатуры -- -- -- -- -- -- 57
1.2.7. Примеры программирования клавиатуры -- -- -- -- -- -- 58
1.2.7.1. Примеры использования функций BIOS -- -- -- -- -- -- 58
1.2.7.1.1. Программа тестирования функции 00h -- -- -- -- -- -- 58
1.2.7.1.2. Процедуры вывода данных на экран и ввода информации с клавиатуры -- -- 59
1.2.7.1.3. Мнемоническое обозначение управляющих клавиш и цветовых кодов -- -- 63
1.2.7.1.4. Макрокоманды для вывода данных на экран -- -- -- -- -- 64
1.2.7.1.5. Программа тестирования функции 01h прерывания Int16h, обрабатываемого BIOS 65

1.2.7.1.6. Программные модули, выполняемые в среде отладчика AFD -- -- -- 65
1.2.7.2. Примеры работы с клавиатурой непосредственно через адаптер клавиатуры -- 66
1.2.7.2.1. Программа отображения потока данных от клавиатуры в шестнадцатеричных кодах 66
1.2.7.2.2. Программа упрощенного драйвера клавиатуры -- -- -- -- -- 68
1.2.7.2.3. Программа чтения скан-кодов, сориентированная на выполнение в среде отладчика AFD 70
Контрольные вопросы -- -- -- -- -- -- -- -- 72
1.3. PS/2 MOUSE -- -- -- -- -- -- -- -- -- 75
1.3.1. Конструктивы PS/2 Mouse -- -- -- -- -- -- -- -- 75
1.3.2. Схемотехника манипуляторов 3D PS/2 Mouse -- -- -- -- -- 78
1.3.3. Интерфейс PS/2 Mouse -- -- -- -- -- -- -- -- 82
1.3.4. Функционирование PS/2 Mouse -- -- -- -- -- -- -- 84
1.3.4.1. Группа форматов PS/2 Mouse -- -- -- -- -- -- -- 84
1.3.4.2. Непосредственная работа с PS/2 Mouse на уровне ее команд -- -- -- 85
1.3.4.3. Работа с PS/2 Mouse через функции BIOS прерывания INT 15h -- -- -- 92
1.3.4.4. Примеры программирования PS/2 Mouse -- -- -- -- -- -- 95
1.3.4.4.1. Программы, выполняемые в среде DOS -- -- -- -- -- -- 95
1.3.4.4.2. Программные модули, выполняемые в среде отладчика AFD -- -- -- 101
Контрольные вопросы -- -- -- -- -- -- -- -- 103
2. Манипуляторы-указатели — мышь. -- -- -- -- -- -- 105
2.1. Манипуляторы мышь, подключаемые к COM-портам -- -- -- -- 106
2.1.1. Схемотехника манипуляторов Serial Mouse -- -- -- -- -- -- 107
2.1.2. Интерфейс Serial Mouse -- -- -- -- -- -- -- -- 109
2.1.2.1. Интерфейс RS-232C -- -- -- -- -- -- -- -- 109
2.1.2.2. Родственные интерфейсы и преобразователи уровней -- -- -- -- 113
2.1.2.3. Асинхронный режим передачи -- -- -- -- -- -- -- 116
2.1.2.4. Управление потоком данных -- -- -- -- -- -- -- 117
2.1.2.5. Микросхемы асинхронных приемопередатчиков -- -- -- -- -- 118
2.1.2.5.1. Микросхемы УАПП 8250 и 16450 -- -- -- -- -- -- 119
2.1.2.5.1.1. Характеристики и функциональные возможности -- -- -- -- 119
2.1.2.5.1.2. Блок-схема -- -- -- -- -- -- -- -- -- 119
2.1.2.5.2. УАПП 16550 -- -- -- -- -- -- -- -- -- 120
2.1.2.5.2.1. Характеристики и функциональные возможности -- -- -- -- 121
2.1.2.5.2.2. Блок-схема -- -- -- -- -- -- -- -- -- 121
2.1.2.5.3. Регистры УАПП 8250, 16450 и 16550 -- -- -- -- -- -- 121
2.1.2.6. Системная поддержка COM портов -- -- -- -- -- -- 127
2.1.2.6.1. Драйвер COM-порта Int 14h -- -- -- -- -- -- -- 128
2.1.2.7. Конфигурирование COM-портов -- -- -- -- -- -- -- 129
2.1.2.8. Использование COM портов -- -- -- -- -- -- -- 129
2.1.2.9. COM-порт и PnP-- -- -- -- -- -- -- -- -- 130
2.1.2.10. Неисправности и тестирование COM портов -- -- -- -- -- 131
2.1.2.10.1. Проверка конфигурирования -- -- -- -- -- -- -- 131
2.1.2.10.2. Функциональное тестирование -- -- -- -- -- -- 132
2.1.2.10.3. Питание от интерфейса, или причины неработоспособности мыши -- -- 133
2.1.3.Форматы передачи данных манипуляторами типа Serial Mouse -- -- -- 133
2.1.4. Работа с манипуляторами мышь -- -- -- -- -- -- -- 135
2.1.4.1. Функции DOS, предназначенные для работы с мышью -- -- -- 135
2.1.4.1.1. Прерывание 33h, функция 0000h: проверить наличие драйвера мыши и произвести сброс (Инициализация мыши) -- -- -- -- -- -- -- 136
2.1.4.1.2. Прерывание 33h, функция 0001 h: включить курсор мыши -- -- -- 136
2.1.4.1.3. Прерывание 33h, функция 0002h: выключить курсор мыши -- -- -- 136
2.1.4.1.3.1. Программа MSCURSOR -- -- -- -- -- -- -- 137
2.1.4.1.4. Прерывание 33h, функция 0003h: получить информацию о положении курсора и состоянии кнопок мыши -- -- -- -- -- -- -- -- -- 137
2.1.4.1.4.1. Программа MSGCURS -- -- -- -- -- -- -- 138
2.1.4.1.5. Прерывание 33h, функция 0004h: установить новое положение курсора -- -- 138
2.1.4.1.6. Прерывание 33h, функция 0005h: определить положение курсора при нажатии клавиши 139
2.1.4.1.7. Прерывание 33h, функция 0006h: определить положение курсора при отпускании клавиши 139
2.1.4.1.8. Прерывание 33h, функция 0007h: задать диапазон движения курсора по горизонтали 139
2.1.4.1.9. Прерывание 33h, функция 0008h: задать диапазон движения курсора по вертикали 140
2.1.4.1.10. Прерывание 33h, функция 0009h: задать форму курсора в графическом режиме -- 140
2.1.4.1.10.1. Программа MSGFORM -- -- -- -- -- -- -- 140
2.1.4.1.11. Прерывание 33h, функция 000Ah: задать форму курсора в текстовом режиме -- 141

2.1.4.1.11.1. Программа MSTFORM -- -- -- -- -- -- -- 141
2.1.4.1.12. Прерывание 33h, функция 000Bh: определить содержимое счетчиков перемещения 142
2.1.4.1.13. Прерывание 33h, функция 000Ch: установить драйвер событий -- -- -- 142
2.1.4.1.13.1. Программа MSDRIVER -- -- -- -- -- -- -- 143
2.1.4.1.14. Прерывание 33h, функция 000Dh: включить эмуляцию светового пера -- -- 144
2.1.4.1.15. Прерывание 33h, функция 000Eh: выключить эмуляцию светового пера -- -- 144
2.1.4.1.16. Прерывание 33h, функция 000Fh: задать скорость перемещения курсора мыши -- 144
2.1.4.1.17. Прерывание 33h, функция 00010h: установить область исключения для курсора -- 144
2.1.4.1.18. Прерывание 33h, функция 00012h: задать увеличенный графический курсор -- 145
2.1.4.1.19. Прерывание 33h, функция 00013h определить порог удвоения скорости -- -- 145
2.1.4.1.20. Прерывание 33h, функция 00014h: заменить драйвер событий -- -- -- 145
2.1.4.1.21. Прерывание 33h, функция 00015h: определить размер буфера состояния драйвера 145
2.1.4.1.22. Прерывание 33h, функция 00016h: сохранить состояние драйвера -- -- 145
2.1.4.1.23. Прерывание 33h, функция 00017h: восстановить состояние драйвера -- -- 145
2.1.4.1.24. Прерывание 33h, функция 00018h: установить альтернативный драйвер событий-- 146
2.1.4.1.25. Прерывание 33h, функция 00019h: получить адрес альтернативного драйвера событий 146
2.1.4.1.26. Прерывание 33h, функция 0001Ah: установить чувствительность мыши -- -- 146
2.1.4.1.27. Прерывание 33h, функция 0001Bh: определить чувствительность мыши -- -- 146
2.1.4.1.28. Прерывание 33h, функция 0001Ch: установить частоту прерываний для Inport Mouse 146
2.1.4.1.29. Прерывание 33h, функция 0001Dh: установить номер страницы видеопамяти -- 147
2.1.4.1.30. Прерывание 33h, функция 0001Eh: пределить номер страницы видеопамяти -- 147
2.1.4.1.31. Прерывание 33h, функция 0001Fh: отключить драйвер мыши -- -- -- 147
2.1.4.1.32. Прерывание 33h, функция 00020h: восстановить драйвер мыши-- -- -- 147
2.1.4.1.33. Прерывание 33h, функция 00021h: сбросить драйвер мыши -- -- -- 147
2.1.4.1.34. Прерывание 33h, функция 00024h: определить тип мыши -- -- -- 147
2.1.4.2. Непосредственная работа с мышью типа MS Mouse -- -- -- -- 148
2.1.4.3. Непосредственная работа с мышью типа MS Mouse в среде AFD -- -- .. 152
Контрольные вопросы. -- -- -- -- -- -- -- -- 153
3. Игровой порт -- -- -- -- -- -- -- -- -- 156
3.1. Разъем -- -- -- -- -- -- -- -- -- -- 156
3.2. Внутреннее устройство игрового порта -- -- -- -- -- -- 156
3.3. Программное управление -- -- -- -- -- -- -- -- 158
3.3.1. QBASIC -- -- -- -- -- -- -- -- -- -- 158
3.3.2. Turbo Pascal 6. -- -- -- -- -- -- -- -- -- 158
3.3.3. Assembler -- -- -- -- -- -- -- -- -- -- 160
3.3.3.1. Определение сопротивления потенциометров джойстика -- -- -- -- 162
3.3.3.2. Перевод чисел из десятичного кода в двоичный и наоборот -- -- -- 163
3.3.4. Borland C/C^{++} -- -- -- -- -- -- -- -- -- 171
Контрольные вопросы -- -- -- -- -- -- -- -- 173
ПРАКТИКУМ -- -- -- -- -- -- -- -- -- 174
1. Общие указания по выполнению и оформлению лабораторных работ -- -- 174
2. Лабораторная работа 1 (описание) "Адаптер клавиатуры и PS/2 Mouse" -- -- 175
1. Цель работы -- -- -- -- -- -- -- -- -- 175
2. Технические и программные средства, используемые в работе -- -- -- 175
3. Краткое описание адаптера клавиатуры и мыши PS/2 -- -- -- -- 175
4. Указания по подготовке к лабораторной работе -- -- -- -- -- 179
5. Порядок выполнения лабораторной работы -- -- -- -- -- 179
6. Примеры программ -- -- -- -- -- -- -- -- 179
7. Рекомендуемая литература -- -- -- -- -- -- -- 183
8. Контрольные вопросы -- -- -- -- -- -- -- -- 183
3. Лабораторная работа 2 (описание) "Электронная клавиатура и ее интерфейс" -- -- 185
1. Цель работы -- -- -- -- -- -- -- -- -- 185
2. Технические и программные средства, используемые в работе -- -- -- 185
3. Краткое описание клавиатуры и ее интерфейса -- -- -- -- -- 185
4. Указания по подготовке к лабораторной работе -- -- -- -- -- 191
5. Порядок выполнения лабораторной работы -- -- -- -- -- 192
6. Примеры программ -- -- -- -- -- -- -- -- 193
7. Рекомендуемая литература -- -- -- -- -- -- -- 195
8. Контрольные вопросы -- -- -- -- -- -- -- -- 195
4. Лабораторная работа 3 (описание) "Программное взаимодействие с клавиатурой" -- 196
1. Цель работы -- -- -- -- -- -- -- -- -- 196
2. Технические и программные средства, используемые в работе -- -- -- 196
3. Краткое описание средств и правил программного взаимодействия с клавиатурой -- 196

4. Указания по подготовке к лабораторной работе -- -- -- -- -- 204
5. Порядок выполнения лабораторной работы -- -- -- -- -- 204
6. Примеры программ -- -- -- -- -- -- -- -- 204
7. Рекомендуемая литература -- -- -- -- -- -- -- 208
8. Контрольные вопросы -- -- -- -- -- -- -- -- 208
5. Лабораторная работа 4 (описание) "Манипулятор PS/2 Mouse" -- -- -- 210
1. Цель работы -- -- -- -- -- -- -- -- -- 210
2. Технические и программные средства, используемые в работе -- -- -- 210
3. Краткое описание оптического манипулятора 3D USB/PS/2 Mouse и его интерфейса 210
4. Указания по подготовке к лабораторной работе -- -- -- -- -- 217
5. Порядок выполнения лабораторной работы -- -- -- -- -- 217
6. Примеры программ -- -- -- -- -- -- -- -- 218
7. Рекомендуемая литература -- -- -- -- -- -- -- 219
8. Контрольные вопросы -- -- -- -- -- -- -- -- 219
6. Лабораторная работа 5 (описание) "Программное взаимодействие с PS/2 Mouse" -- 221
1. Цель работы -- -- -- -- -- -- -- -- -- 221
2. Технические и программные средства, используемые в работе -- -- -- 221
3. Краткое описание средств и правил программного взаимодействия с PS/2 Mouse -- 221
4. Указания по подготовке к лабораторной работе -- -- -- -- -- 225
5. Порядок выполнения лабораторной работы -- -- -- -- -- 225
6. Примеры программ -- -- -- -- -- -- -- -- 226
7. Рекомендуемая литература -- -- -- -- -- -- -- 231
8. Контрольные вопросы -- -- -- -- -- -- -- -- 231
7. Лабораторная работа 6 (описание) "Последовательный асинхронный порт ПК (COM-порт)" 232
1. Цель работы -- -- -- -- -- -- -- -- -- 232
2. Технические и программные средства, используемые в работе -- -- -- 232
3. Краткое описание интерфейса RS-232C, контроллера COM-порта и функций драйвера COM-порта INT 14h -- -- -- -- -- -- -- -- 232
4. Указания по подготовке к лабораторной работе -- -- -- -- -- 245
5. Порядок выполнения лабораторной работы -- -- -- -- -- 245
6. Примеры программ -- -- -- -- -- -- -- -- 247
7. Рекомендуемая литература -- -- -- -- -- -- -- 254
8. Контрольные вопросы -- -- -- -- -- -- -- -- 254
8. Лабораторная работа 7 (описание) "Манипулятор Serial Mouse" -- -- -- 255
1. Цель работы -- -- -- -- -- -- -- -- -- 255
2. Технические и программные средства, используемые в работе -- -- -- 255
3. Краткое описание манипулятора Serial Mouse и его интерфейса -- -- -- 255
4. Указания по подготовке к лабораторной работе -- -- -- -- -- 259
5. Порядок выполнения лабораторной работы -- -- -- -- -- 260
6. Примеры программ -- -- -- -- -- -- -- -- 261
7. Рекомендуемая литература -- -- -- -- -- -- -- 266
8. Контрольные вопросы -- -- -- -- -- -- -- -- 266
ПРИЛОЖЕНИЕ -- -- -- -- -- -- -- -- -- 268
1. Контроллер прерываний -- -- -- -- -- -- -- -- 268
Контрольные вопросы -- -- -- -- -- -- -- -- 270
2. Полноэкранный отладчик реального режима -- -- -- -- -- -- 271
3. Описание команд отладчика DEBUG -- -- -- -- -- -- 276
4. Система команд микропроцессора i8086 -- -- -- -- -- -- 281
СПИСОК ЛИТЕРАТУРЫ -- -- -- -- -- -- -- -- 287

ВВЕДЕНИЕ

Подсистемы клавиатуры и манипуляторов являются составной частью системы ввода-вывода (СВВ) персональных компьютеров (ПК) и входит в состав интерфейса пользователя, обеспечивая, совместно с видео подсистемой, интерактивное взаимодействие пользователя с ПК.

Клавиатура и манипулятор типа PS-Mouse подключаются к системе через адаптер, выполненный на базе однокристальной микро-ЭВМ (микроконтроллера) типа i8042. Они имеют системную поддержку на уровне базовой системы ввода-вывода (BIOS - Basic Input-Output System). Для клавиатуры, как для основного компонента интерфейса пользователя, в BIOS реализован как обработчик прерывания клавиатуры INT 09h (линия IRQ 1 подсистемы аппаратных прерываний, адрес вектора прерывания 0000:0024h), так и драйвер клавиатуры, представленный функциями программного прерывания INT 16h (драйвера клавиатуры). Манипулятор PS-Mouse в рамках BIOS поддерживается только на уровне драйвера, представленного подфункциями 00 - 07 функции C2h программного прерывания INT 15h. Обработчик прерывания INT 74h (линия IRQ 12 подсистемы аппаратных прерываний, вектор прерывания 0000:01D0h) на уровне BIOS не поддерживается и реализуется в рамках операционных систем (ОС), или в рамках приложений. На уровне регистров (на уровне команд ввода-вывода процессора) клавиатура и PS-Mouse доступны по адресам 60h и 64h адресного пространства ввода-вывода (порты пространства ввода-вывода), закрепленным за адаптером клавиатуры и PS-Mouse.

Манипуляторы типа COM-Mouse (Serial Mouse) подключаются к ПК через его COM-порты и никакой поддержки на уровне BIOS не имеют. Их работа поддерживается соответствующим программным обеспечением в рамках ОС или приложений. На регистровом уровне COM-Mouse доступна по адресам, закрепленным за контроллерами COM-портов. Системными соглашениями за контроллерами COM-портов закреплены базовые адреса ввода-вывода: COM 1 - 03F8h, COM 2 - 02F8h. За COM-портами закреплены также и линии аппаратных прерываний: COM 1 - IRQ 4 (INT 0Ch с вектором 0000:0030h); COM 2 - IRQ 3 (INT 0Bh с вектором 0000:002Ch).

Манипулятор джойстик, подключаемый к игровому порту ПК, на уровне BIOS поддерживается функцией 84h программного прерывания INT 15h, используемой для опроса состояния кнопок и чтения координат. В системе аппаратных прерываний ПК за игровым портом никаких линий прерываний не закреплено. На регистровом уровне джойстик доступен системе через контроллер игрового порта по адресу 0201h.

С целью обеспечения совместимости нового аппаратного обеспечения ПК со старым программным обеспечением все современные чипсеты (элементы системной логики) повторяют в своей структуре особенности микроконтроллеров адаптера клавиатуры и PS-Mouse (i8042, i8042B или i8242) а также контроллера COM-порта (i8250 и 16550A). Некоторые чипсеты реализуют функции и контроллера игрового порта.

1. Подсистема клавиатуры и PS-mouse

Подсистема клавиатуры ПК тира IBM PC/AT была представлена контроллером интерфейса клавиатуры (адаптером клавиатуры), выполненным на базе микроконтроллера i8042 (однокристальная микро-ЭВМ) и электронной клавиатурой, содержащей устройство управления, выполненное, как правило, на базе того же микроконтроллера i8042 или его аналога. Микроконтроллеры настраивались на выполнение функций адаптера клавиатуры или устройства управления клавиатурой (контроллера клавиатуры) соответствующим набором программных модулей, хранящихся в постоянном запоминающем устройстве (ПЗУ) соответствующего микроконтроллера.

Для подключения клавиатуры к ПК на аппаратном уровне адаптер клавиатуры и контроллер клавиатуры реализуют двунаправленный асинхронный интерфейс с раздельными двунаправленными линиями передачи байтов данных в последовательном формате и синхронизирующей эту передачу последовательности синхросигналов. По этим же линиям реализуется сигнальный протокол интерфейса, обеспечивающий асинхронный обмен с квитированием приема и запросом передачи байтов данных, а также управлением потоком.

В ПК типа PS/2, а позже и в ПК тира PC (в рамках стандарта ATX [1, 2]), адаптер клавиатуры стал использоваться и для подключения к системе манипулятора PS-Mouse через интерфейс, аналогичный интерфейсу клавиатуры.

Упрощенная схема подключения к компьютеру клавиатуры и мыши типа PS/2 (PS-Mouse) показана на рис. 1.1.

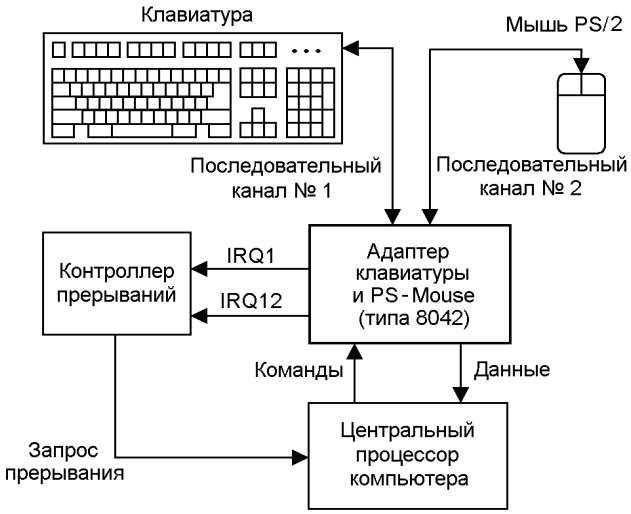

Рис. 1.1. Упрощенная схема подключения клавиатуры и манипулятора PS-Mouse к компьютеру

1.1. Адаптер клавиатуры и манипулятора PS-Mouse.

На системной плате PC/XT адаптер клавиатуры был реализован аппаратной логикой — регистром сдвига, параллельный выход которого подключался к входам системного порта A, реализованного на контроллере параллельного интерфейса i8255. По приему байта от клавиатуры вырабатывалось аппаратное прерывание IRQ1, обработчик которого мог прочитать принятый байт из порта 60h (порт A). С помощью бит 7 и 6 порта 61h (порт B системного интерфейса i8255) была возможна программная блокировка и сброс клавиатуры PC/XT соответственно. Сброс клавиатуры XT осуществлялся обнулением линии KB-Clock.

В архитектуре PC/AT адаптер клавиатуры (адаптер клавиатуры) приобрел новое качество в сравнении с PC/XT. Он стал выполняться на базе однокристального микроконтроллера (однокристальная микро-ЭВМ) типа Intel 8042, который стал обеспечивать выполнение следующих функций:

- управление определенными системными функциями и двунаправленным интерфейсом клавиатуры;
- чтение информации о конфигурации системы для процедур BIOS.

В ПК типа PS/2, а позже и в ПК тира PC (в рамках стандарта ATX [1, 2]), адаптер клавиатуры стал использоваться и для подключения к системе манипулятора PS Mouse через интерфейс, аналогичный интерфейсу клавиатуры (модификации i8042B или i8242 и др.).

В микроконтроллере i8042 постоянно исполняется внутренняя микропрограмма, реагирующая на сигналы интерфейса клавиатуры и манипулятора, а также на команды и данные, поступающие от процессора. Эта микропрограмма хранится во внутреннем масочном ПЗУ микроконтроллера; внешне она недоступна, и микроконтроллер можно рассматривать как устройство с заданными свойствами (адаптер). Поскольку логика работы адаптера реализована программно, его реакция на команды процессора и сигналы интерфейсов относительно медленная — время отклика измеряется десятками микросекунд. В современных ПК адаптер может работать в одном из двух режимов: в режиме AT или режиме PS/2. При инициализации (по аппаратному сбросу) адаптер устанавливается в режим PS/2 или AT, в зависимости от соответствующей

настройки BIOS Setup. В режиме AT адаптер не выполняет функции, связанные с обслуживанием интерфейса мыши и игнорирует все команды, относящиеся к мыши. В режиме PS/2 адаптером поддерживается работа, как с клавиатурой, так и с манипулятором мышь. В этом режиме модифицируются некоторые разряды регистра состояния, регистра командного байта, а также изменяется назначение некоторых выводов портов адаптера (см. ниже).

1.1.1. Микроконтроллер i8042.

На рис. 1.2 приведена блок-схема адаптера клавиатуры, выполненного на базе микроконтроллера i8042, с распределением функций выводов портов, соответствующих режиму AT.

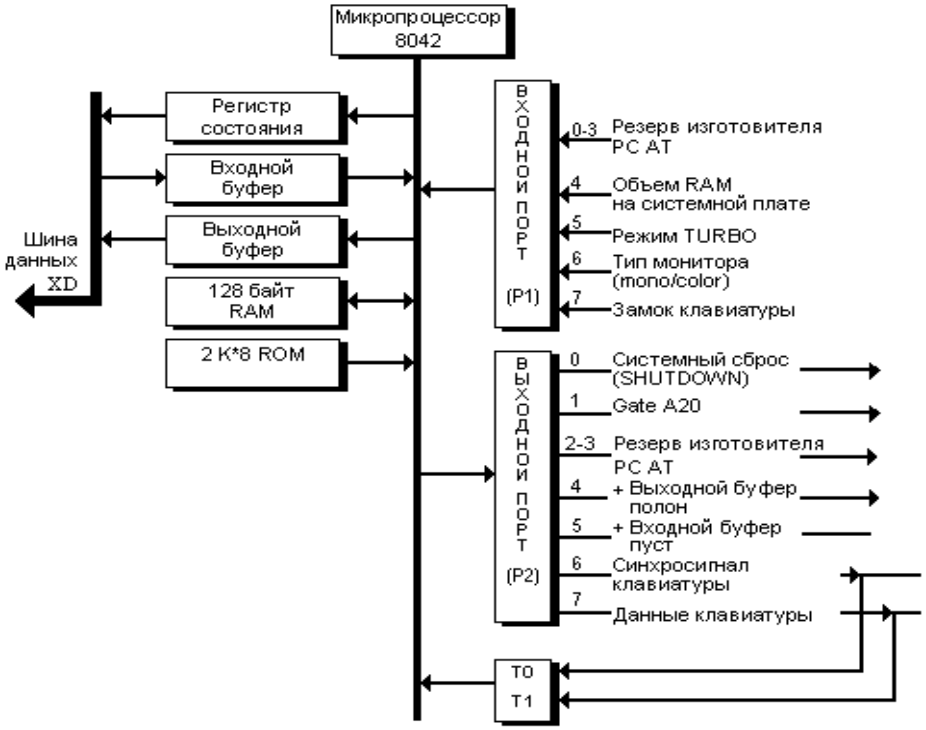

Рис 1.2. Блок-схема адаптера клавиатуры PC/AT, выполненного на базе микроконтроллера i8042.

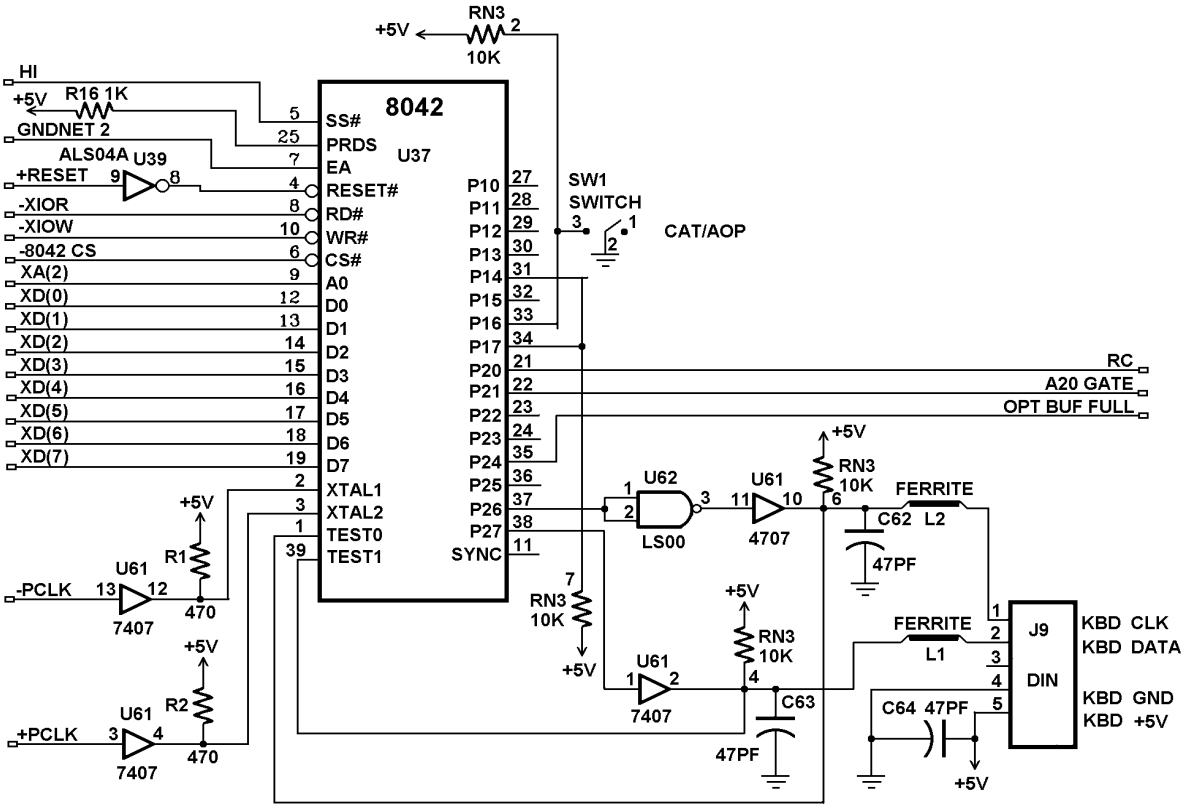

Рис 1.3. Подключение микроконтроллера i8042 в системе персонального компьютера PC/AT.

На рис 1.3 представлен пример подключения микроконтроллера i8042 в системе компьютера PC/AT. Из всех выводов входного порта P1 в данном конкретном схемотехническом решении используется только вывод 6, информирующий о типе используемого в системе видеомонитора. Выводы 4 (объем RAM) и вывод 7 (замок клавиатуры) подключены через резисторы к шине питания +5V, остальные имеют режим холостого хода. Синхронизация микроконтроллера осуществляется синхросигналами серии PCLK. Линии интерфейса клавиатуры KBD CLK и KBD DATA подтягиваются к уровню шины питания +5V через резисторы с сопротивлением 10 Ком и имеют в своих цепях LC-фильтры, защищающие от высокочастотных помех.

Описание назначения выводов микроконтроллера i8042 приведено в табл. 1.1. Описание выводов приводится в общем виде, без указания их конкретного использования в архитектуре PC/AT.

Таблица 1.1. Описание выводов микроконтроллера i8042.

Обозначение вывода	Тип вывода	Номер вывода	Функциональное назначение
TEST 0, TEST 1	Вход	1 39	КОНТРОЛЬНЫЕ ВХОДЫ: входы, которые могут опрашиваться командами условного перехода. ЭТАЛОН ЧАСТОТЫ: TEST 1 (T1) действует так же, как вход таймера событий (под управлением программы). TEST 0 (T0) используется при программировании PROM и проверке ROM/EPROM. Он также используется в режиме SYNC для сброса состояния команды в S1 и синхронизации внутреннего генератора с PH1
XTAL 1, XTAL 2	Вход	2 3	ВХОДЫ: входы, используемые для подключения внешнего сигнала синхронизации
RESET#	Вход	4	СБРОС: этот вход используется для сброса триггеров состояния и установки счетчика команд в нуль. RESET используется также при программировании и проверки EPROM
SS#	Вход	5	ШАГ: этот вход используется совместно с выходом SYNC для пошагового выполнения команд в программе (EPROM). Когда этот вход не используется, он должен быть подключен к питанию 5 В. При подаче на этот выход 12,5 В, он используется также для установки устройства в режим SYNC
CS#	Вход	6	ВЫБОР МИКРОСХЕМЫ: вход используется для выбора одного из нескольких микрокомпьютеров UPI, подключенных к общей шине данных
EA	Вход	7	ВНЕШНИЙ ДОСТУП: вывод внешнего доступа служит для эмуляции, тестирования, а также для проверки ROM/EPROM. Когда этот вход не используется, на него должен подаваться сигнал низкого уровня
RD#	Вход	8	ЧТЕНИЕ: вход, разрешающий читать данные из выходного буфера или регистра состояния
A0	Вход	9	ВЫБОР КОМАНДА/ДАННЫЕ: вход адреса используется процессором для выдачи UPI данных (A0=0, F1 сброшен) или команды (A0=1, F1 установлен). При программировании и проверке A0=0
WR#	Вход	10	ЗАПИСЬ: вход, разрешающий записывать данные и команды во входной буфер контроллера
SYNC	Выход	11	ВЫХОДНОЙ СИНХРОИМПУЛЬС: выходной сигнал, который вырабатывается в каждом командном цикле 8042. SYNC может использоваться в качестве строба для внешних схем; он используется также для синхронизации пошаговой работы
D0 - D7 (BUS)	Вход-выход	12 - 19	ШИНА ДАННЫХ: трех стабильные двунаправленные линии буфера шины данных, которые используются для интерфейса между 8042 и 8 разрядной системной шиной данных
P10 P17	Вход-выход	27 - 34	ПОРТ 1: 8 разрядные квазидвунаправленные линии ввода-вывода порта 1. P10 - P17 обращаются к строке шифра пользователя и разряду защиты

PRDS	Вход-выход	25	ПРОГРАММА: многофункциональный вывод, который используется в качестве импульсного входа программы при программировании PROM. При доступе расширителя ввода вывода вывод PROG выполняет функцию строба адреса/днные для 8243. Когда этот вывод не используется, на него должен быть подан сигнал высокого уровня
P20 P27,	Вход-выход	21 - 24 35 - 38	ПОРТ 2: 8 разрядные квази двунаправленные линии ввода-вывода порта 2. Младшие четыре разряда (P20 - P23) сопрягаются непосредственно с расширителем ввода вывода 8243 и содержат адрес и данные во время доступа в порты 4 - 7. Старшие четыре разряда (P24 - P27) могут быть запрограммированы для обеспечения запроса прерывания и квитирования контроллера DMA. Под управлением программы P24 может быть конфигурирован как прерывание по переполнению выходного буфера (OBF), P25 как прерывание по переполнению входного буфера (IBF), P26 как запрос DMA (DRQ), а P27 как подтверждение DMA (DACK)
VCC		40	ПИТАНИЕ: вывод основного питания (5В)
VDD		26	ПИТАНИЕ: 5 В при обычной работе, 12,5 В при программировании. Вывод резервного питания низкого напряжения
VSS		20	"ЗЕМЛЯ"

1.1.2. Интерфейс: адаптер клавиатуры - система.

Микроконтроллер i8042 (адаптер клавиатуры) подключается к системе через шину периферийных устройств XD при помощи программно-доступных входного и выходного буферов, которые представляют собой два буфера шины данных: один для ввода и один для вывода. На рис.1.4 изображена схема подключения адаптера к шине XD.

Рис. 1.4. Схема подключения адаптера клавиатуры к шине XD

Выходной буфер представляет собой 8-разрядный регистр, который доступен системе только как читаемый порт с адресом 60h. Адаптер использует выходной буфер для передачи системе кодов сканирования, полученных от клавиатуры, и байтов данных, прочитанных по командам, посылаемым системой адаптеру клавиатуры, клавиатуре и манипулятору PS-Mouse.

Выходной буфер должен быть прочитан только после того, как установлен признак "выходной буфер полон" (разряд 0 регистра состояния).

Входной буфер представляет собой 8-разрядный регистр, который доступен системе только для записи по адресам 60h или 64h. Операция записи по адресу 60h устанавливает флажок, который указывает на запись данных. Запись по адресу 64h устанавливает флажок, который указывает на запись команды для адаптера клавиатуры. Данные, записанные по адресу 60h, отсылаются клавиатуре, если не являются атрибутами команды для адаптера. Команда, записанная по адресу 64h, выполняется адаптером. Данные или команды могут быть записаны во входной буфер только после того, как сброшен признак "входной буфер полон" (разряд 1 регистра состояния).

Регистр состояния, доступный для чтения по адресу 64h, несет информацию о состоянии адаптера и интерфейса, в частности о том, когда можно читать данные из выходного буфера, или производить запись во входной буфер адаптера.

1.1.3. Интерфейсы: адаптер - клавиатура и адаптер - PS Mouse.

Адаптер клавиатуры и клавиатура связаны четырех проводным экранированным кабелем, включающим линию питания (+5 В), линию заземления, линии сигнала данных и сигнала синхронизации. Адаптер и клавиатура взаимодействуют с помощью механизма квитирования, используя двунаправленные линии данных и синхронизации, как для синхронизированной последовательной двунаправленной передачи байтов данных, так и для реализации механизма запроса-квитирования этих передач.

Упрощенные схемы цепей данных и синхронизации приведены на рис. 1.5.

Клавиатура PC/AT передает и принимает данные в 11-разрядном формате. Адаптер автоматически определяет тип клавиатуры по формату данных. Первый разряд - стартовый, за ним следуют восемь информационных разрядов, разряд паритета и стоповый разряд. Посылка данных синхронизируется

клавиатурой. В конце передачи адаптер клавиатуры блокирует интерфейс до тех пор, пока система не примет полученный байт. Если байт данных получен с ошибкой паритета, в клавиатуру автоматически отсылается команда RESEND (код FEh).

На рис. 1.6 показаны упрощенный 11-разрядный формат и пример передачи данных.

Более подробное описание интерфейса адаптер - клавиатура представлено в разделах 1.2.4 и 1.2.5.

Адаптер и манипулятор PS-Mouse связаны между собой интерфейсом, аналогичным интерфейсу подключения клавиатуры.

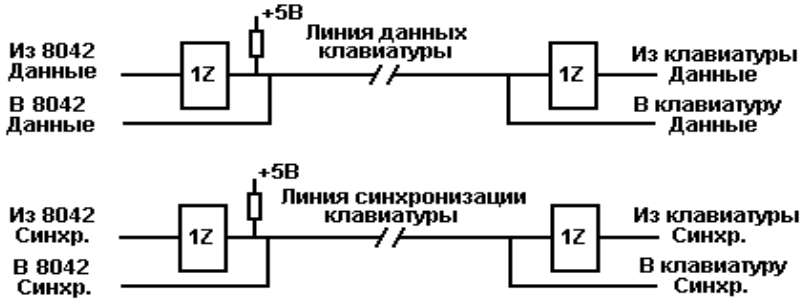

Рис. 1.5. Упрощенные схемы цепей данных и синхронизации

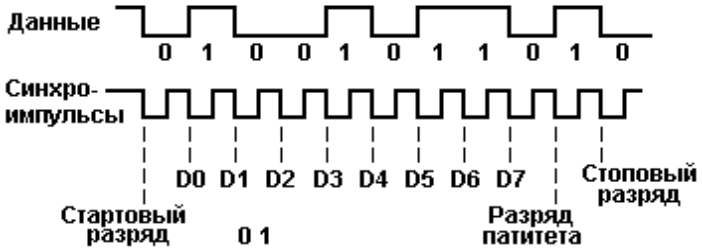

Рис. 1.6. 11-разрядные форматы данных.

Примечание: клавиатура сбрасывает стоповый разряд линии данных в конце их посылки для подтверждения передачи.

1.1.4. Программно доступные регистры адаптера.

К программно доступным регистрам адаптера клавиатуры и PS-Mouse относятся: регистр состояния, регистр команд, регистр данных и регистр командного байта. Первые три регистра доступны центральному процессору ПК непосредственно по адресам 64h и 60h соответственно. Регистр командного байта доступен по чтению и записи через выходной и входной буфер соответственно с помощью специальных команд, посылаемых адаптеру (команды 20h и 60h соответственно).

1.1.4 1. Регистр состояния адаптера.

Регистр состояния представляет собой 8-разрядный регистр, который доступен системе только как читаемый порт с адресом 64h. Он содержит информацию о состоянии адаптера и интерфейсов клавиатуры и мыши. Может быть прочитан в любое время.

Значение разрядов регистра следующее:

- бит 0 — признак "выходной буфере заполнен" (OBF) (0 — буфер пуст, 1 — буфер заполнен);
- бит 1 — признак "входной буфер заполнен" (0 — буфер пуст, 1 — буфер заполнен);
- бит 2 — признак типа последнего общесистемного сброса (0 — сброс по включению питания, 1 — программный сброс);
- бит 3 — признак записи команды (0 — последняя операция записи являлась операцией записи данных, 1 — записи команды);
- бит 4 — состояние «замка» клавиатуры (0 — клавиатура блокирована, 1 — не блокирована);
- бит 5 — в AT-режиме: признак "ошибка тайм-аута передачи" (0 — нормальное завершение передачи, 1 — произошла ошибка); в PS/2-режиме: признак "выходной буфер мыши заполнен" (Mouse_OBF) (0 — буфер пуст, 1 — буфер заполнен);
- бит 6 — в AT-режиме: признак "ошибка тайм-аута приема" (0 — нормальное завершение приема, 1 — произошла ошибка); в PS/2-режиме: общий признак "ошибка тайм-аута при приеме или передаче данных" (0 — нормальное завершение операции, 1 — произошла ошибка);
- бит 7 — признак: "ошибка паритета при приеме или передаче данных" (0 — нет ошибки, 1 — обнаружена ошибка по четности).

1.1.4.2. Регистр командного байта адаптера.

Командный байт адаптера клавиатуры хранится во внутреннем регистре адаптера и определяет режимы работы адаптера и интерфейса. Прочитать и записать командный байт можно с помощью команд адаптера 20h и 60h соответственно (подробнее см. 1.1.5. Команды адаптера).

Разряды регистра имеют следующее назначение:

- бит 0 — управление выдачей сигнала прерывания по готовности данных в выходном буфере клавиатуры (выходной буфер адаптера) (0 — генерация прерывания запрещена, 1 — разрешена);
- бит 1 — управление выдачей сигнала прерывания по готовности данных в выходном буфере мыши (0 — генерация прерывания запрещена, 1 — разрешена);
- бит 2 — установка признака системного сброса (значение, записанное в этот разряд, переносится в разряд 2 регистра состояния);
- бит 3 — управление функцией блокировки клавиатуры (0 — функция блокировки разрешена, 1 — функция запрещена и блокировка игнорируется);
- бит 4 — управление интерфейсом клавиатуры (0 — обмен данными с клавиатурой разрешен, 1 — запрещен);
- бит 5 — признак типа протокола передачи данных в AT-режиме (0 — протокол IBM); управление интерфейсом мыши в PS/2-режиме (0 — обмен данными с мышью разрешен, 1 — запрещен);
- бит 6 — преобразование скан-кодов в PC-совместимые (0 — выключено, 1 — включено); по умолчанию имеет значение 1, то есть скан-коды преобразуются в PC-совместимые;
- бит 7 — не используется (зарезервирован).

1.1.4.3. Регистр команд.

Регистр команд адаптера доступен только по записи через порт 64h. В него заносятся коды команд, которые должен выполнять адаптер. Команды можно посылать в порт 64h только при условии, что входной буфер пуст (бит 1 регистра состояния =0).

1.1.4.4. Регистр данных

Регистр данных доступен для записи и считывания через порт 60h. В режиме считывания он служит для приема информации от клавиатуры, адаптера и мыши. В режиме записи регистр данных служит для передачи команд и их атрибутов клавиатуре, координатному устройству (мыши PS/2-типа) и атрибутов команд самому адаптеру. Какого-либо особого формата данный регистр не имеет.

1.1.5. Команды адаптера

Управление адаптером клавиатуры и манипулятора мышь осуществляется одно– или двухбайтными командами. Первый, возможно единственный, байт команды записывается в порт 64h, а следующий (атрибут команды), если он есть, записывается в порт 60h.

Коды команд адаптера приведены в табл. 1.2.

Некоторые адаптеры в режиме PS/2 могут выполнять дополнительный набор команд, использовавшийся в компьютерах PS/2. В основном эти команды расширяют диагностические возможности и позволяют организовывать парольную защиту. Ниже приводится описание этого набора команд.

Считать байт памяти (20h-3Fh). Данные команды предназначены для считывания памяти контроллера адаптера клавиатуры. Биты 5-0 команды определяют шестнадцатеричный адрес внутренней памяти 8042, т.е. сам код команды несет в себе параметр. Команда с кодом 20h (считать управляющий (командный) байт), таким образом, является всего лишь частным случаем этой группы команд. Запрошенный байт памяти помещается адаптером в выходной буфер.

Табл. 1.2. Коды команд адаптера клавиатуры и мыши

Код	Функция
20h	Прочитать текущий командный байт. Адаптер помещает значение командного байта в порт 060h.
60h	Загрузить новый командный байт. Это часть 2-байтовой операции. Чтобы записать новый командный байт, необходимо 1) записать 60h в порт 064h, 2) записать командный байт в порт 060h
A1h	Чтение номера версии адаптера
A4h	Проверка пароля (признака подключения клавиатуры). При успешном завершение команды через выходной буфер возвращается код F1h (PS/2 адаптер).
A5h	Специальное чтение - 8042 помещает в выходной буфер истинное значение порта 2 за исключением разрядов 4 и 5, которым дается новое определение. Не генерируется состояние «Буфер полон». Если разряд 5 = 0, используется 9-разрядная клавиатура (XT);

	5 = 1, используется 11-разрядная клавиатура; 4 = 0, запрещается прерывание по заполнению выходного буфера; 4 = 1, разрешается прерывание по заполнению выходного буфера. В PS/2 адаптерах данная команда инициирует процедуру установки пароля.
A7h	Запретить работу с мышью. Выполняется только в режиме PS/2.
A8h	Разрешить работу с мышью. Выполняется только в режиме PS/2.
A9h	Тест интерфейса мыши. При успешном выполнение команды в выходной буфер помещается код 00h.
AAh	Инициализация – 8042 инициализирует порты 1 и 2, запрещает клавиатуру и очищает указатели буфера. Затем он помещает код 55h в выходной буфер.
ABh	Контроль интерфейса клавиатуры - направляет 8042 на тестирование линий данных и синхронизации интерфейса клавиатуры. Выходной буфер (порт ввода 60h) получает результаты тестирования: 00h - ошибок не обнаружено; 01h - линия синхронизации клавиатуры «остается нулевой»; 02h - линия синхронизации клавиатуры «остается единичной»; 03h - линия данных клавиатуры «остается нулевой»; 04h - линия данных клавиатуры «остается единичной».
ACh	Дамп диагностирования. Резервируется для диагностирования
ADh	Запретить клавиатуру. Устанавливает разряд 4 командного байта 8042, который запрещает интерфейс клавиатуры. До разрешения клавиатуры данные не передаются и не принимаются
AEh	Разрешить клавиатуру. Сбрасывает разряд 4 командного байта 8042, который разрешает интерфейс клавиатуры
AFh	Выдать номер версии
C0h	Прочитать входной порт. Настраивает 8042 на передачу содержимого входного порта и помещение его в выходной буфер (порт ввода 60h). *Пользуйтесь этой командой только при пустом выходном буфере*
D0h	Прочитать выходной порт. Настраивает 8042 на передачу текущего байта выходного порта в выходной буфер (порт ввода 60h). *Пользуйтесь этой командой только при пустом выходном буфере.*
D1h	Записать в выходной порт. Поместить следующий байт, записанный по адресу 60h, в выходной порт. Разряд 0 выходного порта подключен к системному сбросу. Этот разряд не должен быть нулевым.
D2h	Записать байт в выходной буфер клавиатуры.
D3h	Записать байт в выходной буфер мыши (PS/2 режим).
D4h	Передать манипулятору мышь байт данных (PS/2 режим). Команда D4h автоматически разрешает интерфейс мыши, сбрасывая D5 командного байта адаптера в ноль (т.е. дополнительно выполняется команда A8h).
E0h	Прочитать входы TEST0 и TEST1 (тестовый порт). Настраивает 8042 на передачу текущего состояния выводов TEST0 и TEST1 в выходной буфер (порт вывода 60h). TEST0 - это разряд 0, а TEST1 - 1
F0h – FFh	Пульсация выходного порта. Может быть вызвана пульсация разрядов 0-3 выходного порта (перевод в нулевое состояние примерно на 2-6 мкс). Разряды 0-3 в этой команде указывают, какой разряд выходного порта должен пульсировать, если 0, пульсация разрешена, если 1 - запрещена. Заметим, что разряд 0 выходного порта связан со сбросом системы. Пульсация разряда 0 может привести к сбросу системы. Код команды FFh выполняется адаптером ничего не изменяя в нем (нулевая или холостая команда).

Записать байт памяти (60h-7Fh). Данные команды позволяют изменить содержимое памяти адаптера, при этом, аналогично командам 20h-3Fh, биты 5-0 команды определяют шестнадцатеричный адрес памяти, по которому записывается байт данных, передаваемый вслед за командой по адресу 60h.

Проверить установку пароля (A4h). Данная команда предназначена для проверки наличия установленного пароля. Результат проверки в виде одного байта помещается в ответ на эту команду в выходной буфер (адрес 60h). При этом значение FAh означает, что пароль установлен, а значение F1h, что пароль не установлен.

Установить пароль (A5h). Данная команда инициирует процедуру установки пароля. После получения команды адаптер считывает из входного буфера байты, составляющие собственно пароль. Нулевой байт в передаваемой последовательности означает, что все байты, составляющие пароль, переданы. Байты, составляющие пароль, должны передаваться в формате скан-кодов клавиш. Пароль может быть установлен в

любое время, при этом старое значение пароля теряется.

Задание пароля, состоящего из единственного нулевого байта, равносильно отмене пароля.

Следует отметить, что адаптер не предоставляет никакой команды для считывания установленного пароля.

Включить парольную защиту (A6h). По этой команде адаптер клавиатуры включает функцию парольной защиты, что означает, что до получения адаптером последовательности скан-кодов, определенной в пароле, адаптер не передаст в систему никакой информации и не реагирует на команды со стороны системы. Обычно данная команда выполняется при начальном включение компьютера, а также при выполнении системной утилиты, входящей в состав базового программного обеспечения для PS/2. С точки зрения пользователя, после выполнения этой команды компьютер оказывается заблокированным до введения пароля.

1.1.6. Порты ввода и вывода микроконтроллера i8042.

Архитектура микроконтроллера i8042 включает в себя три порта для непосредственного подключения аппаратуры. При реализации функций адаптера клавиатуры и PS-Mouse микроконтроллером с помощью двух восьмиразрядных портов реализуются последовательные интерфейсы, а также осуществляется управление вентилем Gate A20, сигналом сброса системы и чтение сигналов от джамперов системной платы и ключа блокировки клавиатуры. Эти порты не имеют непосредственного отображения в пространстве адресов ввода-вывода PC; доступ к ним осуществляется через команды адаптера. Кроме этих восьмиразрядных портов микроконтроллер имеет два специальных входа T0 и T1 (тестовый порт), которые могут считываться его микропрограммой и использоваться в качестве источников его внутренних прерываний (это не прерывания CPU). Каждая из двунаправленных интерфейсных линий KB-Data, KB-Clock, MS-Data и MS-Clock реализуется битом порта вывода и порта ввода. В режиме PS/2 для чтения линий KB-Clock и MS-Clock используются входы T0 и T1 соответственно, в режиме AT вход T1 используется для чтения линии KB-Data.

Назначение выводов портов в режиме AT покезано на рис. 2, а в режиме PS/2 - приводится ниже.

Порт ввода **P1** доступен для чтения по команде C0h; в режиме PS/2 имеет следующее назначение разрядов:

- бит 7 - ключ блокировки клавиатуры *(KeyLock):* 0 - клавиатура заблокирована;
- бит 6 - перемычка установки видеорежима: 0 - цветной (Color), 1 - монохромный (Mono);
- бит 5 - системная перемычка: 0 - замкнута;
- бит 4 - перемычка задания объема ОЗУ: 0 - 256 Кбайт, 1 - 512 Кбайт и более;
- биты 3, 2 - не используются;
- бит 1 - линия MS-Data, в режиме AT не используется;
- бит 0 - линия KB-Data, в режиме AT не используется.

В современных ПК используются только биты 0, 1 и 7; в режиме AT биты 0 и 1 не используются.

Порт вывода **P2,** доступный для записи и чтения по командам D1h и D0h соответственно, имеет следующее назначение разрядов:

- бит 7 - выходная линия KB-Data;
- бит 6 - выходная линия KB-Clock;
- бит 5 - запрос прерывания от дополнительного интерфейса (IRQ12), в режиме AT не используется;
- бит 4 - запрос прерывания от клавиатуры (IRQ1);
- бит 3 - линия MS-Clock, в режиме AT не используется;
- бит 2 - линия MS-Data, в режиме AT не используется;
- бит 1 - вентиль линии адреса A20 *(Gate A20*): 0 - A20 обнулен, 1 - управляется выводом процессора;
- бит 0 - альтернативный сброс процессора (сигнала INIT, без формирования общего сигнала сброса);

при переключении значения из "1" в "0" возможен сброс или остановки по исключительным ситуациям.

Тестовый порт T, доступен для чтения командой E0h, имеет следующее назначение разрядов:

- бит 0 (T0) - входная линия KB-Clock;
- бит 1 (T1) - входная линия MS-Clock, в режиме AT - входная линия KB-Data.

1.1.7. Функционирование адаптера.

При обеспечение передачи данных с клавиатуры и манипулятора мышь в систему адаптер под управлением своего внутреннего программного обеспечения выполняет ряд предусмотренных интерфейсом действий.

Например, приняв посылку от клавиатуры, адаптер клавиатуры и мыши выполняет внутреннюю трансляцию скан-кода (если это не запрещено командным байтом), помещает его в выходной буфер и устанавливает в регистре состояния флажок OBF=1 (разряд 0), что приводит к генерации запроса прерывания IRQ1 (если это не запрещено командным байтом). В ответ на это хост должен прочитать данные

(транслированный скан-код, префиксы и т. п.) из порта данных (60h). После чтения адаптер сбрасывает флажок OBF в ноль. Трансляция скан-кодов обеспечивает программную совместимость клавиатур XT и AT (см. ниже) по чтению скан-кодов из порта 60h.

При получении посылки от мыши адаптер не выполняет никаких преобразований и помещает ее в выходной буфер мыши, устанавливает в регистре состояния флажок Mouse_OBF=1 (разряд 5), что приводит к генерации запроса прерывания IRQ12 (если это не запрещено командным байтом). Затем, примерно через 15 мкс, адаптер переписывает полученный байт из выходного буфера мыши в выходной буфер адаптера (клавиатуры) и устанавливает в регистре состояния флажок OBF=1 (разряд 0). По этому флагу данные от мыши должны быть считаны хостом (системой) из порта 60h. После чтения адаптер сбрасывает флажок OBF а флажок Mouse_OBF оставляет в единичном состоянии (подробности см. в разделах 1.1.8.3.5. и 1.1.8.3.6.).

При передаче команд и данных в клавиатуру адаптер выполняет сравнительно простые операции: он просто передает в клавиатуру байт данных, который был записан системой в его входной буфер по адресу 60h и ожидает ответного кода от клавиатуры. Получив ответный код, адаптер просто помещает его в выходной буфер без трансляции. Остальные действия соответствуют ранее рассмотренным в процедуре получения посылки от клавиатуры.

При передаче команд и данных в PS-Mouse необходимо сначала по адресу 64h передать в адаптер команду D4h "Передать мыши байт данных". После освобождения входного буфера по адресу 60h необходимо записать байт, который адаптер должен передать манипулятору мышь. Адаптер передает этот байт мыши и ожидает ответного кода. Получив от мыши ответный код, адаптер помещает его в выходной буфер мыши. Остальные действия соответствуют ранее рассмотренным в описании процедуры получения посылки от PS Mouse.

После получения по адресу 64h команды, по которой адаптер должен возвращать данные, он помещает байт данных в выходной буфер и устанавливает в регистре состояния OBF=1, что приводит к генерации запроса прерывания IRQ1 (если это не запрещено командным байтом). После этого данные должны быть прочитаны из порта данных (по адресу 60h). Если команда возвращает несколько байтов данных, прерывание генерируется для каждого байта.

После получения по адресу 64h кода команды, которая должна содержать дополнительный байт атрибутов, адаптер инициирует процедуру выполнения этой команды. Далее он устанавливает в регистре состояния признак "Входной буфер пуст" (разряд 1 регистра состояния устанавливается в 0) и ожидает получения от системы через регистр данных (входной буфер) с адресом 60h атрибутного байта команды. После его получения адаптер завершает выполнение этой команды и освобождает входной буфер.

Следует отметить, что на время выполнения адаптером команд, адаптер блокирует интерфейсы клавиатуры и мыши, устанавливая низкий уровней на их линиях синхронизации KB-Clock и MS-Clock. Аналогично при приеме данных от клавиатуры адаптер блокирует интерфейс мыши на время передачи одного байта и наоборот.

1.1.8. Программирование адаптера клавиатуры и PS Mouse.

Адаптер программируется в адресном пространстве портов 60h и 64h. Перед записью команды или данных в порты 60h или 64h необходимо убедиться, что в регистре состояния адаптера установлен указатель "Входной буфер пуст" (разряд 1 установлен в 0).

Перед чтением данных из порта 60h необходимо проверить регистр состояния, который должен указывать состояние "Выходной буфер полон" (разряд 0 установлен в 1) или "Выходной буфер мыши полон" (разряд 5 установлен в 1).

Для чтения данных из выходного буфера адаптера по адресу 60h используется команда IN микропроцессора. Данные в выходном буфере - это данные от клавиатуры или мыши, если только адаптер предварительно не получил определенную команду, например 20h - читать командный байт адаптера. В последнем случае из выходного буфера читается значение текущего командного байта.

Для передачи данных в клавиатуру через входной буфер по адресу 60h используется команда OUT. Все данные, записываемые во входной буфер по адресу 60h адаптером передаются в клавиатуру, если только адаптер не получил много байтовую команду, например 60h - записать командный байт. В последнем случае переданный через порт 60h байт будет записан в регистр командного байта, а не передан в клавиатуру или мышь. Для передачи данных в манипулятор мышь используется команда D4h - передать мыши байт данных. После получения этого кода по адресу 64h, адаптер ожидает поступление байта данных во входной буфер по адресу 60h для передачи его в манипулятор мышь.

Чтобы выдать на клавиатуру много байтовую команду, необходимо ввести первый командный байт в порт 60h и после положительного ответа клавиатуры (код FAh в порту 60h) ввести второй командный байт (атрибут) в порт 60h и дождаться ответа от клавиатуры (код FAh в порту 60h).

Для передачи много байтовой команды манипулятору мышь необходимо выполнять аналогичный набор операций, только перед каждым занесением байта данных по адресу 60h, необходимо по адресу 64h заносить команду D4h .

Для чтения регистра состояния адаптера используется команда IN (ввод из порта 64h), а чтобы передать команду в адаптер, используется команда OUT (вывод в порт 64h). Запись по этому адресу указывает, что следующий байт, записанный в порт 60h, может быть атрибутом команды. Большинство команд

выполняются за один шаг. Однако для выполнения некоторых команд требуется второй шаг, например чтение или запись портов и регистров 8042.

Для работы с адаптером клавиатуры и PS-Mouse используются следующие команды языка Ассемблер:

```
IN      AL, 64h   ;прочитать в аккумулятор AL процессора содержимое регистра состояния 8042
OUT     64h, AL   ;вывести команду из AL в 8042
IN      AL, 60h   ;прочитать в AL данные из порта 60h
OUT     60h, AL   ;вывести данные из AL в порт 60h
```

Перед отсылкой команд и данных в порты 64h и 60h программа должна предварительно убедиться в том, что входной буфер адаптера пуст и, следовательно, адаптер готов получить байт. Аналогично перед чтением данных из порта 60h программа должна предварительно убедиться в том, что в выходной буфер адаптер поместил байт данных. Информацию о состоянии входного и выходного буферов программа может получить чтением регистра состояния по адресу 64h.

1.1.8.1. Программирование портов микроконтроллера i8042.

Для работы с портами микроконтроллера i8042 используются команды адаптера D1h, D0h, C0h и E0h (см. табл. 1.2).

Запись в выходной порт.

Для записи в выходной порт необходимо:

- дождавшись освобождения входного буфера передать команду D1h (записать в выходной порт) по адресу 64h,
- дождавшись освобождения входного буфера передать значение данных для выходного порта по адресу 60h.

При программировании состояния разряда адресной шины A20 через выходной порт i8042 программа, использующая определенный режим адресации, должна «убедиться», что разряд Gate A20 установлен на нужное значение перед тем, как будут продолжены действия программы в выбранном режиме адресации.

Например, некоторые программы разрешают A20 при входе в защищенный режим для доступа в память выше адреса 100000h. Эти программы перед обращением к этой области памяти должны ждать, пока A20 будет действительно разрешен аппаратно.

Для того чтобы быть уверенным в установке A20 на нужное значение, рекомендуется выдать адаптеру нулевую команду (FFh) после завершения последовательности команды записи в выходной порт (D1h). Когда адаптер принимает эту нулевую команду, программа может быть «уверена», что предыдущая команда (записать в выходной порт) выполнена, и разряд порта Gate A20 установлен аппаратно на требуемое значение. Контролировать состояние вывода A20 чтением состояния выходного порта i8042 не рекомендуется, поскольку в компьютерах старше PC/AT этот разряд всегда показывает значение 1.

Не рекомендуется использовать команду D1 для изменения любого разряда выходного порта, кроме разряда 1 (Gate A20). Это связано с тем, что изменение состояния разряд 0 выходного порта с 1 на 0 в компьютерах PC/AT и вплоть до Pentium 133/166 вызывает сброс системы.

В чипсетах современных компьютеров операция записи в выходной порт выполняется как "холостая" операция, т.е. никакая информация в регистр выходного порта не записывается, однако фактически состояние управления линией A20 изменяется в зависимости от значения разряда 1 записываемого в выходной порт байта. Факт переключения состояния управления линией A20 можно проверить чтением в реальном режиме адресации содержимого области памяти HYMEM (первые 64 килобайта памяти выше адреса 100000h) и нижней страницы памяти. Если их содержимое со сдвигом на 16 адресов совпадает, то линя A20 установлена в 0, если нет, то ее состояние управляется адресным регистром процессора.

Чтение выходного порта.

Чтобы прочитать значение выходного порта i8042, нужно:

- дождавшись освобождения входного и выходного буферов адаптера передать команду D0h (прочитать выходной порт) по адресу 64h,
- дождавшись заполнения выходного буфера прочесть значение выходного порта по адресу 60h.

Чтение входного порта.

Чтобы прочитать значение входного порта i8042, нужно:

- дождавшись освобождения входного и выходного буферов адаптера передать по адресу 64h команду C0h (прочитать входной порт),
- дождавшись заполнения выходного буфера прочесть значение входного порта по адресу 60h.

Чтение тестового порта.

Чтобы прочитать значение тестового порта (TEST) микроконтроллера, необходимо:

- дождавшись освобождения входного и выходного буферов адаптера передать по адресу 64h команду E0h (чтение тестового порта),
- дождавшись заполнения выходного буфера прочитать значение тестового порта по адресу 60h.

1.1.8.2. Примеры программирования на Ассемблере.

Передача команды в адаптер может быть выполнена следующим фрагментом программы [8]:

```
STATUS_REG  EQU    64h         ;Регистр состояния
                               ;Флаги регистра состояния:
IBF         EQU    00000010b   ;- входной буфер полон
OBF         EQU    00000001b   ;- выходной буфер полон
;
INBUF_COM   EQU    64h         ;Входной буфер для команды
INBUF_DTA   EQU    60h         ;Входной буфер для данных
OUTBUF              EQU  60h          ;Выходной буфер
;
SOME_CMD    EQU    ??          ;Некоторая команда адаптера
SOME_PAR    EQU    ??          ;Возможно некоторый параметр
ENDS

CODESEG
            XOR    CX,CX       ;Загрузить счетчик для
                               ;ограничения времени цикла
            CLI                ;Запретить прерывания
;
;Ждать, пока входной буфер не будет свободен
;
Teet_IBF1:  IN     AL,STATUS_REQ
            TEST   AL,IBF
            LOOPNZ TEET_IBF1
            JNZ    BAD_EXIT    ;Ошибка, если буфер не осво-
                               ;бодился за FFFFh повторов
            MOV    AL,SOME_CMD      ;Послать команду в
                               ;адаптер
            OUT    INBUF_COM,AL
;
; Следующий участок кода необходим, если после команды должен пере-
; даваться некоторый параметр; иначе, необходимо просто разрешить
; прерывания (выдать команду STI).
;
            XOR    CX,CX
TEET_IBF2:  IN     AL,STATUS_REG ;Ждать, пока входной буфер не
                                 ;освободится (команда не будет
                                 ;принята)
            TEST   AT,IBF
            LOOPNZ TEET_IBF2
            JNZ    BAD_EXIT    ;Обработать ошибку
            MOV    AL,SOME_PAR ;Послать параметр
            OUT    INBUF_DTA,AL
            STI                ;Разрешить прерывания
```

В том случае, если команда адаптера предусматривает получение ответа, перед ее выполнением необходимо убедиться, что выходной буфер адаптера пуст. Ниже приведен фрагмент программного кода, демонстрирующий выполнение команды с получением ответа от адаптера:

```
            CLI                ; Запретить прерывания
;
; Ждать в бесконечном цикле, пока выходной буфер не освободится,
; т.е. пока все из него не считаем (лучше конечно цикл ограничить).
;
TEET_OBF1:  IN     AL,STATUS_REG
            TEST   AL,OBF
            JZ     READY_TO_READ
            IN     AL,OUTBUF
            JMP    SHORT TEET_OBF1
READY_TO_READ:
            XOR    CX,CX       ;Загрузить счетчик ограниче-
                               ;ния циклов
```

```
;
;Ждать, пока входной буфер не будет свободен
;
TEET_IBF1:  IN     AL,STATUS_REQ
            TEST   AL,IBF
            LOOPNZ TEST_IBF1
            JNZ    BAD_EXIT    ;Ошибка, если буфер не осво-
                               ;бодился за FFFFh повторов
            MOV    AL,SOME_CMD ;Послать команду
            OUT    INBUF_COM,AL
            XOR    CX,CX
;
;Поскольку прерывания запрещены, а выходной буфер пуст, то первое,
;что там появится - ответ адаптера.
;
TEST_OBF2:  IN    AL,STATUS_REG  ;Ждать, пока не появится ответ
                  TEST  AL,OBF
            JZ    TEET_OBF
            IN    AL,OUTBUF    ;Считать ответ адаптера
            STI                ;Разрешить прерывания
```

Следует обратить внимание на то, что считывание ответа адаптера из выходного буфера всегда приводит к сбросу признака "выходной буфер полон" (разряд 0 регистра состояния сбрасывается в 0). Сам же выходной буфер не очищается и если повторно прочитать его содержимое без проверки признака его заполнения, то будет выдано значение последнего значения выходного буфера.

1.1.8.3. Примеры программирования адаптера в среде AFD.

Приведенные в данном разделе программные модули сориентированы на их выполнение в AFD (отладчик DOS с полноэкранным интерфейсом пользователя), запускаемым из DOS 7. Использовался ПК типа Pentium III -500 мГц с материнской платой ASUS P3B-F. В этом ПК адаптер клавиатуры/мыши реализуется или в рамках специализированной микросхемы Super I/O типа Winbond (W83977EF) совместно с контроллерами других устройств и малых интерфейсов (LPT, COM, Floppy и др.) [1, 2], или эмулируется программно в области памяти SMRAM (System Management RAM), доступной процессору из режима работы SMM (System Management Mode - режим системного управления) [17].

1.1.8.3.1. Программа CKLD0.bin, позволяющая получить общие характеристики команд адаптера.

В листинге 1.1 представлен программный модуль, выполняемый в среде отладчика AFD, и реализующий процедуру чтения ***командой*** **D0h** байта состояния выходного порта микроконтроллера i8042. По команде D0h адаптер помещает значение байта состояния выходного порта в выходной буфер по адресу 60h. Во время выполнения команды интерфейсы клавиатуры и мыши блокируются примерно на 0,3 миллисекунды.

Листинг 1.1 Чтение состояния выходного порта микроконтроллера (CKLD0.BIN).

```
2C47:0100  FA      CLI           ;Запрет прерываний.
2C47:0101  E464    IN     AL,[64] ;Ожидание освобождения
2C47:0103  A802    TEST   AL,02   ;входного
2C47:0105  75FA    JNZ    0101    ;буфера.
2C47:0107  E464    IN     AL,[64] ;Проверка освобождения
2C47:0109  A801    TEST   AL,01   ;выходного буфера:
2C47:010B  7406    JZ     0113    ;если пуст, то переход на 0113;
2C47:010D  E460    IN     AL,[60] ;если буфер не пуст, то его содержимое
2C47:010F  88C3    MOV    BL,AL   ;считывается и переносится в BL и далее
2C47:0111  EBF4    JMP    0107    ;на новый цикл проверки.
2C47:0113  B0D0    MOV    AL,D0   ;Посылка кода команды D0h
2C47:0115  E664    OUT    [64],AL ;в адаптер клавиатуры.
2C47:0117  B9FFFF  MOV    CX,FFFF ;Задание максимального числа циклов.
2C47:011A  E464    IN     AL,[64] ;Проверка: помещены ли в выходной буфер
2C47:011C  A801    TEST   AL,01   ;результаты выполнения команды?
2C47:011E  E1FA    LOOPZ  011A    ;нет - новый цикл проверки.
2C47:0120  E460    IN     AL,[60] ;Чтение байта сост.вых.порта из вых. буф.
2C47:0122  88C7    MOV    BH,AL   ;и пересылка его в регистр BH.
2C47:0124  FB      STI           ;Разрешение аппаратных прерываний.
2C47:0125  CC      INT3          ;Возвращение управления отладчику.
```

Этот программный модуль может быть использован для задания адаптеру и других команд, по которым адаптер возвращает через выходной буфер 60h результат их выполнения. К таким командам относятся ***команды:*** **20h, 21h - 3Fh, A1h, A4h, A9h, ABh, и C0h.** Также он может быть использован для выполнения однобайтовых ***команд*** **(A7h, A8h, ADh, AEh, F0h - FFh)** и для определения некоторых характеристик любых других команд, поскольку в нем предусмотрен выход из цикла ожидания признака "выходной буфер заполнен" по тайм-ауту (CX=0).

Результаты выполнения программой CKLD0.BIN выше перечисленных команд представлены ниже.

Команда	Значение CX	Возвращаемое значение	Состояние интерфейсов мыши и клавиатуры	Примечание
20h	FF8Ch	47h	Нормальное	
21h - 3Fh	FF8Ch	00h	Нормальное	
A1h	FF84h	48h	Нормальное	
A4h	FF7Fh	F1h	Нормальное	
A5h	0000h	1Ch	Заблокированы	
A7h	B000h	9Ch	Мышь заблокирована	
A8h	B000h	9Ch	Нормальное	
A9h	FF67h	00h	Нормальное	
AAh	9F00h	55h	Заблокированы	
ABh	FF66h	00h	Нормальное	
ACh	0000h	1Ch	Заблокированы	
ADh	0000h	1Ch	Кл. заблокирована	
AEh	B000h	9Ch	Нормальное	
AFh	0000h	1Ch	Заблокированы	
C0h	FF8Fh	FFh	Нормальное	
D0h	FF8Ch	4Bh	Нормальное	
E0h	FF87h	00h	Нормальное	
FFh, F1h, F3h, F7h	B000h	9Ch	Нормальное	
F8h, FCh	B000h	9Ch	Нормальное	
FEh	--	--	Нормальное	Остановка в области BIOS по "286 exception Interrupt".

Относительное время выполнения команд, в которых предусмотрено возвращение результата их выполнения, можно оценить по значениям регистра CX (FFFFh - CX = число циклов). Значение CX почти для всех этих команд одинаково (CX = FF8Fh - FF66h). Исключение составляет команда **AAh** (CX = 9F00h). Это связано с большими временными затратами на выполнение адаптером процедур самотестирования. Возвращаемые этими командами параметры различны и, как правило, соответствуют ожидаемым значениям (команды **20h, A4h, A9h, AAh, ABh**). Команды **21h - 3Fh** возвращают значения 00h, если предварительно командами **61h - 7Fh** в соответствующие адреса не была занесена иная информация. Для однозначной интерпретации результатов выполнения остальных команд этого типа требуется более детальная информация о системе (команды **A1h, C0h, D0h, E0h).**

Почти для всех однобайтовых команд **(A7h, A8h, AEh, FFh, F0h - FDh)** программа через регистр AH возвращает код 9Ch (системный код отпускания клавиши "Enter"). Это связано с тем, что в среде отладчика AFD программный модуль CKLD0.BIN запускается на выполнение командой отладчика G100 по нажатию клавиши "Enter" (системный код нажатия 1Ch). Поскольку при выполнение команд адаптера интерфейсы клавиатуры и мыши блокируются, то код отпускания клавиши "Enter" поступает из клавиатуры в адаптер только после окончания выполнения однобайтовой команды во время цикла ожидания признака "выходной буфер заполнен". Время поступления этого кода в выходной буфер адаптера определяется содержимым регистра CX. Значение CX для этих команд приблизительно одинаково (CX=B000h). При выполнение команды **ADh** в AH возвращается значение 1Ch - последний переданный клавиатурой код (нажатие "Enter"). Помимо этого содержимое CX= 0000h, т. е. выход из цикла ожидания флага "выходной буфер заполнен" произошел по тайм-ауту. Это связано с тем, что эта команда блокирует интерфейс клавиатуры и передача кодов с клавиатуры не возможна. После выполнения программы CKLD0.BIN с этой командой приходится перезагружать систему кнопкой "Reset" или выключением и включением питания. При выполнение команды **FEh** происходит остановка программы в области BIOS по"286 exception Interrupt", т. е. эта команда влияет на систему в целом. В ПК более ранних версий, чем используемый при получении выше приведенных результатов, выполнение этой команды приводило к перезагрузке компьютеров.

Для команд **F5h, ACh** и **AFh,** у которых нет достаточно однозначного описания, были получены результаты, которые позволяют отнести их к много байтовым командам, требующим передачу адаптеру байтов их атрибутов. При выполнение этих команд программа CKLD0.BIN завершает свою работу по тайм-ауту (CX=0) и возвращает в AH значение 1Ch. При этом интерфейсы клавиатуры и мыши остаются заблокированными. Т.е. адаптер воспринял код команды и остался в состоянии ожидания передачи ему

атрибутов команды. Аналогичные результаты были получены и для двухбайтовых команд записи (**60h, D2h**, **D3h и D4h)**.

1.1.8.3.2. Программа CKLA1C.BIN циклического выполнения команд, возвращающих результаты.

Абсолютные значения временных параметров процесса выполнения адаптером команд, возвращающих результаты их выполнения, можно оценить с помощью программного модуля, представленного в листинге 1.2. Временные параметры фиксируются электронным осциллографом (см. рис. 1.7). Время блокировки интерфейсов определяется длительностью отрицательных импульсов, формируемых на линиях синхронизации интерфейсов во время выполнения программы. Время задержки установки признака "выходной буфер полон" определяется длительностью импульсов, формируемых на линиях данных параллельного порта (LPT 2). Для этого в программе предусмотрена выдача через регистр данных (RD) контроллера параллельного порта LPT 2 на линии интерфейса Centronics кода команды, в момент передачи ее в адаптер клавиатуры, и возвращаемого адаптером клавиатуры кода, в момент чтения его из порта 60h. Поскольку время обращения к портам адаптера клавиатуры и контроллера параллельного порта составляет примерно 0.5 мкс, то это значение и определяет ограничения точности получаемых результатов (не учитывая погрешности осциллографа).

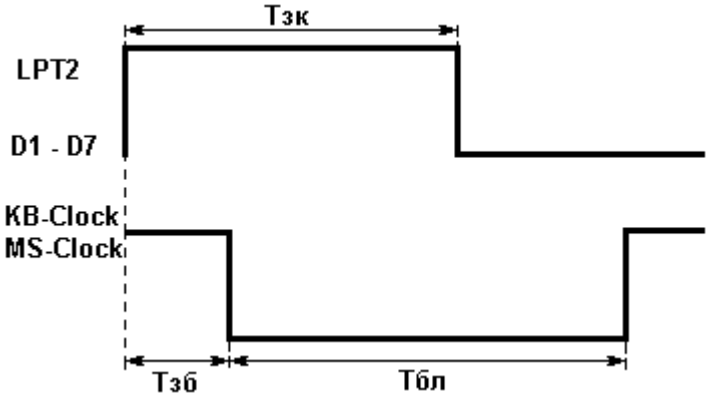

Рис. 1.7. Временные параметры выполнения команды A1h.

Для ***команды A1h*** были получены следующие временные параметры:

- время блокировки интерфейса (Тбл) - - - - - 0,3 мс.
- время задержки установки признака "выходной буфер полон" относительно момента записи кода команды в порт 64h и в RD LPT 2 (Тзк) - - - - - 0,17 мс.
- время задержки начала блокировки интерфейсов относительно момента записи кода команды в порт 64h и в RD LPT 2 (Тзб) - - - - - - - 50 мкс.

Приблизительно аналогичные временные параметры были получены и для ***команд: 20h, A4h, C0h, D0h и команд 21h - 3Fh.***

Временные параметры выполнения ***команд A9h и ABh*** имеют отличительные особенности: Тбл = 0,35мс, Тзк = 0,22мс, Тзб = 50мкс. Помимо этого на временной диаграмме блокирующих сигналов наблюдается положительный перепад длительностью 15мкс, возникающий через 0,1мс после формирования сигналов блокировки интерфейсов. Этот перепад отображает процедуру тестирования адаптером синхронизирующих линий интерфейсов мыши и клавиатуры соответственно.

Листинг 1.2. Циклическое выполнение команды A1h (CKLA1C.BIN).

```
2C47:0100  BA0001  MOV    DX,0100 ;Число циклов выполнения программы.
2C47:0103  FA      CLI            ;Запрет аппаратных прерываний
2C47:0104  52      PUSH   DX      ;Запоминание DX в стеке.
2C47:0105  E464    IN     AL,[64] ;Чтение регистра состояния и
2C47:0107  A802    TEST   AL,02   ;контроль: D1=0?
2C47:0109  75FA    JNZ    0105    ;если не 0, новый цикл.
2C47:010B  E464    IN     AL,[64] ;Чтение регистра состояния и
2C47:010D  A801    TEST   AL,01   ;контроль: D0=0?
2C47:010F  7406    JZ     0117    ;если не 0, новый цикл.
2C47:0111  E460    IN     AL,[60] ;Очистка выходного буфера чтением,
2C47:0113  88C3    MOV    BL,AL   ;прочитанный байт - в BL,
2C47:0115  EBF4    JMP    010B    ;новый цикл.
2C47:0117  B0A1    MOV    AL,A1   ;Код команды заносится в AL.
2C47:0119  BA7803  MOV    DX,0378 ;Адрес RD LPT2 в DX.
2C47:011C  EE      OUT    DX,AL   ;Содержимое AL - в RD LPT2.
2C47:011D  E664    OUT    [64],AL ;Код команды - в адаптер.
2C47:011F  B9FFFF  MOV    CX,FFFF ;Макс число циклов ожидания
```

```
2C47:0122  E464    IN     AL,[64] ;Чтение регистра состояния и
2C47:0124  A801    TEST   AL,01   ;проверка: выходной буфер полон?
2C47:0126  E1FA    LOOPZ  0122    ;если пуст, то новый цикл ожидания.
2C47:0128  E460    IN     AL,[60] ;Чтение результата выполнения команды
2C47:012A  EE      OUT    DX,AL   ;и пересылка его в RD LPT2.
2C47:012B  B90F00  MOV    CX,000F ;Двойной
2C47:012E  51      PUSH   CX      ;цикл
2C47:012F  B9FFFF  MOV    CX,FFFF ;программной
2C47:0132  E2FE    LOOP   0132    ;задержки
2C47:0134  59      POP    CX
2C47:0135  E2F7    LOOP   012E
2C47:0137  5A      POP    DX      ;Восстановление DX из стека,
2C47:0138  4A      DEC    DX      ;уменьшение его на 1 и если не 0, то
2C47:0139  75C9    JNZ    0104    ;на новый цикл программы.
2C47:013B  FB      STI            ;Разрешение аппаратных прерываний.
2C47:013C  CC      INT3           ;Возврат в AFD.
```

1.1.8.3.3. Программа CKLA8.BIN циклического выполнения однобайтовых команд.

В листинге 1.3 приводится программа, позволяющая определить временные параметры выполнения однобайтовых ***команд разрешения и запрещения интерфейсов мыши и клавиатуры (A7h, A8h, ADh и AEh).*** Эти команды управляют состоянием разрядов D5 и D4 командного байта адаптера. Поскольку клавиатура и мышь относятся к средствам взаимодействия интерфейса пользователя, то в программе предусмотрено восстановление разрешения интерфейсов клавиатуры и мыши при завершение работы программы.

Временные параметры выполнения команд фиксировались осциллографом на линиях синхронизации интерфейсов мыши и клавиатуры (Тбл) и линии D5 параллельного порта LPT2 (Твхб и Тзб) (см. рис. 1.8).

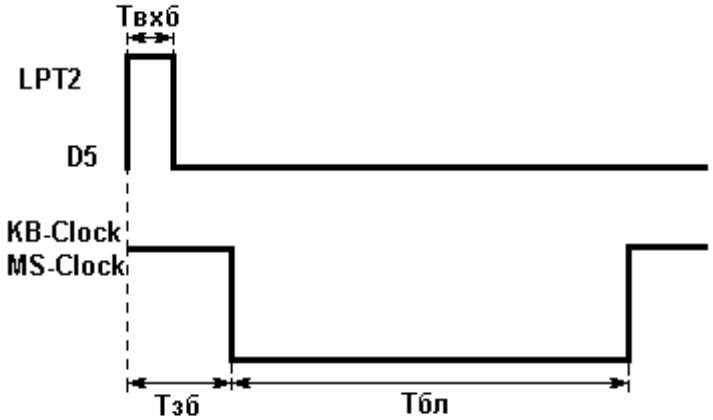

Рис. 1.8. Временные параметры выполнения команды A8h.

Для ***команд A8h и AEh:*** Тбл = 0,22мс, Тзб = 50мкс и Твхб = 23мкс (время освобождения адаптером входного буфера после записи в него кода команды). Для ***команд A7h и Adh***: Тбл = 0,2мс, Тзб = 50мкс, Твхб = 20мкс.

Листинг 1.3. Циклическое выполнение команды A8h (CKLA8.BIN)

```
2C47:0100  BB0011  MOV    BX,1100 ;Определение числа циклов.
2C47:0103  FA      CLI            ;Запрет прерываний
2C47:0104  E464    IN     AL,[64] ;Чтение регистра состояния и определение:
2C47:0106  A802    TEST   AL,02   ;свободен ли входной буфер?
2C47:0108  75FA    JNZ    0104    ;нет - новый цикл ожидания.
2C47:010A  B0A8    MOV    AL,A8   ;Код команды A8h.
2C47:010C  BA7803  MOV    DX,0378 ;Адрес RD LPT 2.
2C47:010F  EE      OUT    DX,AL   ;Выдача кода A8h через LPT-порт.
2C47:0110  E664    OUT    [64],AL ;Выдача кода A8h в адаптер клавиатуры.
2C47:0112  E464    IN     AL,[64] ;Циклы ожидания освобождения
2C47:0114  A802    TEST   AL,02   ;входного
2C47:0116  75FA    JNZ    0112    ;буфера.
2C47:0118  B000    MOV    AL,00   ;Выдача кода 00h
2C47:011A  EE      OUT    DX,AL   ;через LPT-порт.
2C47:011B  FB      STI            ;Разрешение прерываний.
2C47:011C  B90200  MOV    CX,0002 ;Организация программной
2C47:011F  51      PUSH   CX      ;задержки
2C47:0120  B9FFFF  MOV    CX,FFFF ;для разнесения
```

```
2C47:0123  E2FE    LOOP   0123    ;во времени
2C47:0125  59      POP    CX      ;циклов выполнения
2C47:0126  E2F7    LOOP   011F    ;команд адаптером.
2C47:0128  4B      DEC    BX      ;Уменьшение на 1 счетчика циклов.
2C47:0129  75D9    JNZ    0104    ;Если не 0, то переход на новый цикл.
2C47:012B  B0AE    MOV    AL,AE   ;Команда "разрешить работу с клавиатурой"
2C47:012D  E664    OUT    [64],AL ;предается в адаптер.
2C47:012F  E464    IN     AL,[64] ;Проверка состояния
2C47:0131  A802    TEST   AL,02   ;входного буфера адаптера, если буфер
2C47:0133  75FA    JNZ    012F    ;не освобожден, то новый цикл ожидания.
2C47:0135  B0A8    MOV    AL,A8   ;Команда "разрешить работу с мышью"
2C47:0137  E664    OUT    [64],AL ;передается в адаптер.
2C47:0139  CC      INT3           ;Передача управления отладчику.
```

1.1.8.3.4. Программа CKLAA2.BIN циклического выполнения команды AAh.

В листинге 1.4 представлена программа, позволяющая определить временные параметры выполнения ***команды AAh*** - "самотестирование адаптера". Особенность выполнения этой команды заключается в том, что после завершения ее выполнения адаптер оставляет интерфейсы клавиатуры и мыши в заблокированном состоянии. Для разблокирования интерфейсов в программе предусмотрено предварительное запоминание текущего командного байта адаптера в регистре AH, а после выполнения команды AAh - восстановление его значения в регистре командного байта.

Временные параметры выполнения команды AAh фиксировались осциллографом на линиях синхронизации интерфейсов мыши и клавиатуры (Тбл = 35мс) и линии D1 параллельного порта LPT2 (Тзк = 34.8мс и Тзб = 50мкс). После завершения работы программы в регистрах содержалась следующая информация: AX = 4747h (значение командного байта = 47h), BX = 5500h (возвращаемый адаптером код 55h - нормальное завершение теста), CX = 9F3Bh. Значение CX содержит информацию об относительном времени выполнения команды AAh.

Листинг 1.4. Циклическое выполнение команды AAh (CKLAA2.BIN)

```
2C47:0100  BA0001  MOV    DX,0100 ;Задание числа циклов программы.
2C47:0103  FA      CLI            ;Запрет прерываний.
2C47:0104  E464    IN     AL,[64] ;-!
2C47:0106  A802    TEST   AL,02   ; !
2C47:0108  75FA    JNZ    0104    ; !
2C47:010A  E464    IN     AL,[64] ; !
2C47:010C  A801    TEST   AL,01   ; !
2C47:010E  7406    JZ     0116    ; !
2C47:0110  E460    IN     AL,[60] ; ! Чтение и запоминание
2C47:0112  88C3    MOV    BL,AL   ; ! в регистре AH
2C47:0114  EBF4    JMP    010A    ; ! текущего значения
2C47:0116  B020    MOV    AL,20   ; ! командного байта
2C47:0118  E664    OUT    [64],AL ; ! адаптера.
2C47:011A  B9FFFF  MOV    CX,FFFF ; !
2C47:011D  E464    IN     AL,[64] ; !
2C47:011F  A801    TEST   AL,01   ; !
2C47:0121  E1FA    LOOPZ  011D    ; !
2C47:0123  E460    IN     AL,[60] ; !
2C47:0125  88C4    MOV    AH,AL   ;-!
2C47:0127  52      PUSH   DX      ;Текущее значение BX заносится в стек.
2C47:0128  E464    IN     AL,[64] ;-!
2C47:012A  A802    TEST   AL,02   ; !
2C47:012C  75FA    JNZ    0128    ; !Ожидание освобождения
2C47:012E  E464    IN     AL,[64] ; !входного и
2C47:0130  A801    TEST   AL,01   ; !выходного
2C47:0132  7406    JZ     013A    ; !буферов
2C47:0134  E460    IN     AL,[60] ; !адаптера.
2C47:0136  88C3    MOV    BL,AL   ; !
2C47:0138  EBF4    JMP    012E    ;-!
2C47:013A  BA7803  MOV    DX,0378 ;Базовый адрес LPT 2 в DX.
2C47:013D  B0AA    MOV    AL,AA   ;Код команды "Самотестир. адаптера"
2C47:013F  EE      OUT    DX,AL   ;AAh выдается в LPT 2 как байт
2C47:0140  E664    OUT    [64],AL ;и как команда, в адаптер.
2C47:0142  B9FFFF  MOV    CX,FFFF ;Цикл ожидания возвращения
2C47:0145  E464    IN     AL,[64] ;информации об окончании
2C47:0147  A801    TEST   AL,01   ;самотестирования
```

```
2C47:0149  E1FA    LOOPZ  0145    ;
2C47:014B  51      PUSH   CX      ;Запоминание числа циклов ожидания.
2C47:014C  E460    IN     AL,[60] ;Чтение результата самотестирования,
2C47:014E  EE      OUT    DX,AL   ;выдача его в LPT 2
2C47:014F  88C7    MOV    BH,AL   ;и запоминание в регистре BH.
2C47:0151  E464    IN     AL,[64] ;-!
2C47:0153  A802    TEST   AL,02   ; !
2C47:0155  75FA    JNZ    0151    ; !Восстановление из
2C47:0157  B060    MOV    AL,60   ; !регистра AH текущего
2C47:0159  E664    OUT    [64],AL ; !значения командного
2C47:015B  E464    IN     AL,[64] ; !байта адаптера
2C47:015D  A802    TEST   AL,02   ; !командой 60h.
2C47:015F  75FA    JNZ    015B    ; !
2C47:0161  88E0    MOV    AL,AH   ; !
2C47:0163  E660    OUT    [60],AL ;-!
2C47:0165  B9FFFF  MOV    CX,1FFF ;Цикл программной
2C47:0168  E2FE    LOOP   0168    ;задержки
2C47:016A  59      POP    CX      ;Восстановление из стека CX.
2C47:016B  5A      POP    DX      ;Восстановление из стека текущего значения
2C47:016C  4A      DEC    DX      ;числа циклов, уменьшение его на 1 и ,ели
2C47:016D  75B8    JNZ    0127    ;не нуль, то на новый цикл программы.
2C47:016F  FB      STI            ;Разрешение прерываний.
2C47:0170  CC      INT3           ;Остановка и возврат в AFD.
```

1.1.8.3.5. Программа CKL60M.BIN циклического выполнения двухбайтовых команд.

В листинге 1.5 приведена программа, позволяющая определить временные параметры выполнения ***команд 60h, D1h D2h и команд 61h - 7Fh.*** В программе предусмотрены процедуры передачи в адаптер кода команды, ее атрибута и процедура ожидания получения кода результата выполнения команды. Если в команде не предусмотрено возвращение кода результата, то последняя процедура завершаются по тайм-ауту (CX = 0).

Для ***команды 60h*** "запись командного байта" получены следующие параметры: Тбл = 0,24 мс, Твхб = 30мкс, Тзб = 54 мкс. Возвращение кода результата выполнения команды не предусмотрено. Выход из цикла ожидания заполнения выходного буфера адаптера осуществляется по таймауту (CX =0). Аналогичные результаты были получены и для команд **61h - 7Fh**.

Для ***команды D1h*** "запись байта в выходной порт" фиксируется только параметр Твхб = 5мкс. Блокировка интерфейса отсутствует и в выходной буфер ничего не возвращается. Для ПК Pentium III-500 МГц результатом выполнения команды D1h может быть только изменение состояния разрешения разряда A20 (разряд D6 выходного порта). Причем изменение состояния выходного порта невозможно определить чтением его состояния командой D0h. Изменение состояния разряда D6 выходного порта можно проконтролировать только косвенно: отображаются или не отображаются на нулевой сегмент памяти адреса FFFF:0010h - FFFF:FFFFh (высокая память - HMA). Для ПК с реальными микроконтроллерами i8042 командой D1h можно инициировать перезагрузку ПК.

Для ***команды D2h*** "запись байта в выходной буфер клавиатуры" получены следующие параметры: Тбл = 0,31 мс, Твхб = 20 мкс, Тзб = 45 мкс. Результат выполнения команды помещается в выходной буфер через 0,17 мс после начала выполнения команды (Твых.б=0,17 мс) (см. рис. 1.9).

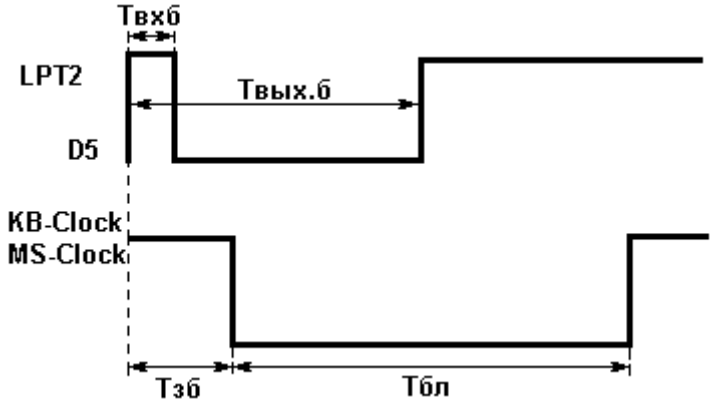

Рис. 1.9. Временные параметры выполнения команды D2h.

Для ***команды A5h*** "установить пароль" в двухбайтовом варианте при значение 00h второго байта были получены следующие параметры: Тбл = 0.22 мс, Твхб = 22 мес, Тзб = 50 мкс. Возвращение байта результата выполнения команды не предусмотрено (CX =0).

Для ***команды ACh*** "дамп диагностики" получены следующие параметры: Тбл = 0,31 мс, Твхб = 30 мкс, Тзб = 54 мкс. Результат выполнения команды помещается в выходной буфер через 0,17 мс после начала выполнения команды (Твых.б=0,17 мс). Значение возвращаемого адаптером байта зависит от значения второго байта команды. Например, по байту FEh возвращается значение 41h, по байту 88h - значение 1Dh а по 05h - значение 80h и т.д. Для интерпретации этих значений необходима более подробная информация о назначении команды ACh.

Эту же программу можно использовать и для определения временных параметров выполнения ***команд D3h и D4h***. Однако в этой программе не анализируется состояние разряда D5 регистра состояния адаптера, характеризующего состояние выходного буфера мыши. Это связано с тем, что разряд D5 не сбрасывается при чтение содержимого выходного буфера и во время выполнения всех циклов программы, после первого, остается в единичном состоянии. Разряд D5 может быть сброшен любой из команд, результат выполнения которой возвращается через выходной буфер (20h, A1h, A4h, A9h, ABh, C0h, D0h и др.). Разряд D5 сбрасывается также при поступление с клавиатуры скан-кода или кода подтверждения.

Листинг 1.5. Циклическое выполнение команды 60h (CKL60M.BIN).

```
2C47:0100  BA0001  MOV    BX,1000 ;Задание числа циклов программы.
2C47:0103  FA      CLI            ;Запрет прерываний.
2C47:0104  E464    IN     AL,[64] ;Ожидание освобождения
2C47:0106  A802    TEST   AL,02   ;входного
2C47:0108  75FA    JNZ    0104    ;буфера.
2C47:010A  E464    IN     AL,[64] ;Проверка состояния выходного
2C47:010C  A801    TEST   AL,01   ;буфера: если буфер пуст, то
2C47:010E  7406    JZ     0116    ;переход к следующим процедурам,
2C47:0110  E460    IN     AL,[60] ;если буфер не пуст, чтение его
2C47:0112  89C6    MOV    SI,AX   ;содержимого с сохранением в SI и
2C47:0114  EBF4    JMP    010A    ;переход на очередной цикл проверки.
2C47:0116  BA7803  MOV    DX,0378 ;Установка высокого уровня на
2C47:0119  B0FF    MOV    AL,FF   ;всех линиях шины данных
2C47:011B  EE      OUT    DX,AL   ;параллельного порта LPT2.
2C47:011C  B060    MOV    AL,60   ;Передача кода команды 60h
2C47:011E  E664    OUT    [64],AL ;в адаптер клавиатуры.
2C47:0120  E464    IN     AL,[64] ;Ожидание освобождения
2C47:0122  A802    TEST   AL,02   ;входного
2C47:0124  75FA    JNZ    0120    ;буфера.
2C47:0126  B000    MOV    AL,00   ;Установка низкого уровня на всех линиях
2C47:0128  EE      OUT    DX,AL   ;шины данных параллельного порта LPT2.
2C47:0129  B047    MOV    AL,47   ;Значение 47h передается в адаптер,
2C47:012B  E660    OUT    [60],AL ;как атрибут команды 60h.
2C47:012D  B9FF1F  MOV    CX,1FFF ;Число циклов ожидания.
2C47:0130  E464    IN     AL,[64] ;Ожидание установки признака "выходной
2C47:0132  A801    TEST   AL,01   ;буфер заполнился": если буфер пуст и CX
2C47:0134  E1FA    LOOPZ  0130    ;не 0, то новый цикл ожидания.
2C47:0136  B0FF    MOV    AL,FF   ;Установка высокого уровня на всех линиях
2C47:0138  EE      OUT    DX,AL   ;шины данных параллельного порта LPT2.
2C47:0139  E460    IN     AL,[60] ;Чтение содержимого выходного буфера и
2C47:013B  88C4    MOV    AH,AL   ;пересылка его в регистр AH.
2C47:013D  51      PUSH   CX      ;Содержимое CX помещается в стек.
2C47:013E  B9FF4F  MOV    CX,4FFF ;Формирование программной
2C47:0141  E2FE    LOOP   0141    ;задержки
2C47:0143  B000    MOV    AL,00   ;Установка низкого уровня на всех линиях
2C47:0145  EE      OUT    DX,AL   ;шины данных параллельного порта LPT2.
2C47:0146  59      POP    CX     ;В CX заносится верхнее слово стека.
2C47:0147  4B      DEC    BX     ;Уменьшение BX на 1.
2C47:0148  75BA    JNZ    0104    ;Если BX не 0, то новый цикл программы.
2C47:014A  FB      STI            ;Разрешение прерываний.
2C47:014B  CC      INT3           ;Остановка и возврат в отладчик.
```

1.1.8.3.6. Программа CKLD34.BIN циклического выполнения команд D3h и D4h.

В листинге 1.6 приведена программа, позволяющая определять временные параметры выполнения ***команд D3h и D4h*** со сбросом командой C0h разряда D5 регистра состояния адаптера в каждом цикле программы (см. замечания к командам D3 и D4 в разделе 1.1.8.3.5.). В начале каждого цикла программы выполняется проверка на обнуление входного буфера и выходных буферов клавиатуры и мыши. Определенные этапы выполнения программы фиксируются изменением содержимого регистра данных LPT-порта (LPT 2). После выполнения программы в регистрах возвращается следующая информация:

AH - результат выполнения команды D3h (AH=47h записанный в буфер код) или D4h (AH=FEh - мышью принята неправильная команда);

AL - значение регистра состояния после выполнения команд D3h или D4h (AL=34h для обеих команд - разряд D5 остается в единичном состоянии);

BL - значение регистра состояния после выполнения команды C0h (BL=1Ch - для обеих команд - разряд D5 обнулен);

CX - число циклов (исходное значение CX = FFFFh) ожидания результата выполнения команд: для D3h - CX=0F9Dh, для D4h - CX=0824h.

Временные параметры выполнения ***команды D3h,*** полученные с помощью программы CKL60.BIN (листинг 5.5) следующие: Твхб = 30мкс, Тзб = 50 мкс, Тбл = 0,31мс, Твых.б = 0,18мс. По программе CKLD34.BIN для первых трех параметров получены аналогичные результаты. Четвертый параметр имеет меньшую величину (Твых.б = 0,17мс), т.е. признак "выходной буфер мыши заполнен" устанавливается на 10 мкс раньше, чем признак "выходной буфер (клавиатуры) заполнен". Этого времени должно быть достаточно, чтобы обработчик прерывания мыши INT 74h перехватил обработку события "выходной буфер заполнен" у обработчика прерываний клавиатуры INT 09h.

Временные параметры выполнения ***команды D4h,*** полученные по обеим программам идентичны в пределах погрешностей. Это связано с относительно большим значением параметра Твых.б = 2,8 мс. Такое большое время задержки определяется затратой времени на передачу кода в мышь и получение от нее кода подтверждения. Остальные полученные временные параметры имеют следующие значения: Твхб = 25 мкс, Тзб = 50 мкс, Тбл = 0,4 мс, Тн.пер = 0,76 мс. - задержка начала передачи в мышь относительно начала выполнения команды; Тн.пр = 1,9 мс - задержка начала приема кода ответа от мыши; Т пр = 2,8 мс - время окончания приема кода ответа от мыши.

Листинг 1.6. Циклическое выполнение команды D3h (CKLD34.BIN).

```
2C47:0100  BB0010  MOV    BX,1000 ;Число циклов выполнения программы.
2C47:0103  FA      CLI            ;Запрет прерываний.
2C47:0104  E464    IN     AL,[64] ;Проверка: входной буфер пуст?
2C47:0106  A802    TEST   AL,02   ;если не пуст, то
2C47:0108  75FA    JNZ    0104    ;на новый цикл проверки.
2C47:010A  B9FF0F  MOV    CX,0FFF ;Организация ожидания освобождения
2C47:010D  E464    IN     AL,[64] ;выходного буфера клавиатуры и мыши:
2C47:010F  A821    TEST   AL,21   ;
2C47:0111  7404    JZ     0117    ;если буфера пусты, то на 0117h
2C47:0113  E460    IN     AL,[60] ;иначе чтение вых.буф. клавиатуры
2C47:0115  E2F6    LOOP   010D    ;проверка таймаута и на новый цикл ожидан.
2C47:0117  BA7803  MOV    DX,0378 ;Занесение в рег.данных LPT2
2C47:011A  B0FF    MOV    AL,FF   ;константы FFh - линии данных порта
2C47:011C  EE      OUT    DX,AL   ;переводятся в высокий уровень.
2C47:011D  B0D3    MOV    AL,D3   ;Код команды "запись в вых.буф. мыши"
2C47:011F  E664    OUT    [64],AL ;передается адаптеру.
2C47:0121  E464    IN     AL,[64] ;Проверка:
2C47:0123  A802    TEST   AL,02   ;входной буфер пуст?
2C47:0125  75FA    JNZ    0121    ;если не пуст, то новый цикл ожидания.
2C47:0127  B000    MOV    AL,00   ;В рег. данных LPT2 заносится
2C47:0129  EE      OUT    DX,AL   ;константа 00h - низкий уровень линий.
2C47:012A  B047    MOV    AL,47   ;47h - атрибут команды D3h
2C47:012C  E660    OUT    [60],AL ;передается в адаптер.
2C47:012E  B9FF0F  MOV    CX,0FFF ;Организация ожидания с использованием
2C47:0131  E464    IN     AL,[64] ;таймаута установки признака "выходной
2C47:0133  A820    TEST   AL,20   ;буфер мыши полон"
2C47:0135  E1FA    LOOPZ  0131
2C47:0137  B0FF    MOV    AL,FF   ;Занесение в рег.данных LPT2 константы
2C47:0139  EE      OUT    DX,AL   ;FFh - высокий уровень линий.
2C47:013A  E464    IN     AL,[64] ;Ожидание заполнения выходного буфера
2C47:013C  A801    TEST   AL,01   ;адаптера:
2C47:013E  74FA    JZ     013A    ;если пуст, то новый цикл ожидания.
2C47:0140  E460    IN     AL,[60] ;Чтение выходного буфера адаптера
2C47:0142  88C4    MOV    AH,AL   ;и запись его содержимого в рег. AH.
2C47:0144  E464    IN     AL,[64] ;Чтение рег. Состояния адаптера.
2C47:0146  50      PUSH   AX      ;Помещение в стек содержимого рег. AX
2C47:0147  51      PUSH   CX      ;и рег. CX.
2C47:0148  90      NOP
2C47:0149  B9FF3F  MOV    CX,3FFF ;Программная
2C47:014C  E2FE    LOOP   014C    ;задержка.
```

```
2C47:014E  B000    MOV    AL,00   ;Занесение в рег.данных LPT2 константы
2C47:0150  EE      OUT    DX,AL   ;00h - низкий уровень линий.
2C47:0151  B0C0    MOV    AL,C0   ;Командой C0h организуется сброс признака
2C47:0153  E664    OUT    [64],AL ;"выходной буфер мыши полон".
2C47:0155  E464    IN     AL,[64] ;
2C47:0157  A801    TEST   AL,01   ;
2C47:0159  74FA    JZ     0155    ;
2C47:015B  E460    IN     AL,[60] ;Завершение команды C0h.
2C47:015D  B9FFFF  MOV    CX,FFFF ;Программная
2C47:0160  E2FE    LOOP   0160    ;задержка.
2C47:0162  59      POP    CX      ;Извлечение из стека значений
2C47:0163  58      POP    AX      ;регистров CX и AX.
2C47:0164  4B      DEC    BX      ;Уменьшение числа циклов на 1.
2C47:0165  759D    JNZ    0104    ;Если BX не 0, то на новый цикл программы.
2C47:0167  50      PUSH   AX      ;Помещение в стек содержимого AX.
2C47:0168  E464    IN     AL,[64] ;Чтение регистра состояния адаптера и
2C47:016A  88C3    MOV    BL,AL   ;помещение его в регистр BL.
2C47:016C  58      POP    AX      ;Восстановление значения AX из стека.
2C47:016D  FB      STI            ;Разрешение прерываний.
2C47:016E  CC      INT3           ;Останов и передача управления отладчику.
```

1.1.8.3.7. Программа CKLAFT.BIN циклического выполнения трехбайтовой команды AFh.

В листинге 1.7 представлена программа циклического выполнения трехбайтовой команды AFh "выдать номер версии". В ней после передачи кода команды AFh адаптеру передается два байта атрибутов. Никаких ответных кодов адаптер не возвращает. Временные параметры выполнения команды при большинстве комбинаций значений атрибутов примерно соответствуют параметрам двухбайтовых команд: Твхб = 30 мкс, Тзб = 54 мкс и Тбл = 0.22 мс. Однако при комбинации передаваемых адаптеру атрибутов 02h, 02h, адаптер переходит в режим передачи клавиатуре блоков сообщений. Режим передачи можно сбросить программно только командой AFh с первым атрибутом 02h и вторым - не содержащим 0h в младшем разряде, но не равным 02h (например 01h, 55h, CFh и т.п.). Следует отметить, что при передаче блоков правила сигнального протокола интерфейса клавиатуры выполняются только при передаче первого байта блока. Всего передается 12 байт в каждом блоке. Краткое описание передачи блока приводится ниже.

Перед началом передачи блока адаптер переводит линии данных и синхронизации в низкий уровень на время примерно 200 мкс. За это время клавиатура также переводит свой передатчик линии синхронизации в низкий уровень. Через (1 - 1,5) мс после того, как адаптер "отпускает" линию синхронизации, клавиатура переводит ее в высокий уровень и через 0,5 мс выдает по ней 11 синхроимпульсов. После последнего синхроимпульса адаптер переводит линию синхронизации в низкий уровень, а линию данных - в высокий. Через 0,3 мс адаптер переводит линию синхронизации в высокий уровень. В ответ через 0,1 мс клавиатура выдает первый синхроимпульс и переводит линию данных в низкий уровень. Через два периода синхроимпульсов адаптер также переводит эту линию в низкий уровень на 6 периодов синхроимпульсов и затем отпускает ее. Через один тактовый импульс клавиатура переводит линию данных в низкий уровень. В ответ адаптер также переводит ее в низкий уровень и удерживает ее в этом состоянии пока клавиатура не отработает 10 пакетов синхроимпульсов по 11 импульсов в каждом пакете. Расстояние между пакетами 0,5 мс а время передачи пакета 0,88 мс.

Листинг 1.7. Циклическое выполнение команды AFh (CKLAFT.BIN).

```
2C47:0100  BB0001  MOV    BX,0100 ;Число циклов выполнения программы.
2C47:0103  FA      CLI            ;Запрещение прерываний.
2C47:0104  E464    IN     AL,[64] ;Проверка: входной буфер пуст?
2C47:0106  A802    TEST   AL,02   ;Если не пуст, то
2C47:0108  75FA    JNZ    0104    ;на новый цикл проверки.
2C47:010A  E464    IN     AL,[64] ;Проверка: выходной буфер полон?
2C47:010C  A801    TEST   AL,01   ;если пуст, то
2C47:010E  7406    JZ     0116    ;на 0116h,
2C47:0110  E460    IN     AL,[60] ;иначе чтение вых. буфера и запись
2C47:0112  89C6    MOV    SI,AX   ;результата в регистр SI и переход
2C47:0114  EBF4    JMP    010A    ;на новый цикл проверки.
2C47:0116  BA7803  MOV    DX,0378 ;В рег. данных LPT2 заносится константа
2C47:0119  B0FF    MOV    AL,FF   ;FFh - высокий уровень линий данных.
2C47:011B  EE      OUT    DX,AL   ;
2C47:011C  B0AF    MOV    AL,AF   ;Код команды "выдать номер версии"
2C47:011E  E664    OUT    [64],AL ;передается в адаптер.
2C47:0120  E464    IN     AL,[64] ;Проверка: входной буфер пуст?
2C47:0122  A802    TEST   AL,02   ;если не пуст, то
```

```
2C47:0124  75FA    JNZ    0120    ;переход на новый цикл проверки.
2C47:0126  B000    MOV    AL,00   ;В рег. данных LPT2 заносится константа
2C47:0128  EE      OUT    DX,AL   ;00h - низкий уровень линий данных.
2C47:0129  B002    MOV    AL,02   ;Первый атрибут 02h передается
2C47:012B  E660    OUT    [60],AL ;в адаптер.
2C47:012D  E464    IN     AL,[64] ;Проверка: входной буфер пуст?
2C47:012F  A802    TEST   AL,02   ;если не пуст, то
2C47:0131  75FA    JNZ    012D    ;переход на новый цикл проверки.
2C47:0133  B002    MOV    AL,02   ;Второй атрибут 02h передается
2C47:0135  E660    OUT    [60],AL ;в адаптер.
2C47:0137  B9FF1F  MOV    CX,1FFF ;Ожидание с контролем таймаута
2C47:013A  E464    IN     AL,[64] ;заполнения выходного
2C47:013C  A801    TEST   AL,01   ;буфера
2C47:013E  E1FA    LOOPZ  013A    ;адаптера.
2C47:0140  B0FF    MOV    AL,FF   ;В рег. данных LPT2 заносится константа
2C47:0142  EE      OUT    DX,AL   ;FFh - высокий уровень линий данных.
2C47:0143  E460    IN     AL,[60] ;Чтение выходного буфера и занесение
2C47:0145  88C4    MOV    AH,A    ;результата в регистр AH.
2C47:0147  51      PUSH   CX      ;Занесение содержимого CX в стек.
2C47:0148  B9FF1F  MOV    CX,1FFF ;Программная
2C47:014B  E2FE    LOOP   014B    ;задержка.
2C47:014D  B000    MOV    AL,00   ;В рег. данных LPT2 заносится константа
2C47:014F  EE      OUT    DX,AL   ;00h - низкий уровень линий данных.
2C47:0150  59      POP    CX      ;Восстановление из стека CX.
2C47:0151  4B      DEC    BX      ;Уменьшение счетчика циклов на 1.
2C47:0152  75B0    JNZ    0104    ;Если BX не 0, то на новый цикл программы.
2C47:0154  E464    IN     AL,[64] ;Проверка: выходной буфер пуст?
2C47:0156  A801    TEST   AL,01   ;если пуст, та
2C47:0158  7404    JZ     015E    ;на 015Eh,
2C47:015A  E460    IN     AL,[60] ;иначе чтение вых. буфера
2C47:015C  EBF6    JMP    0154    ;и переход на новый цикл проверки.
2C47:015E  E464    IN     AL,[64] ;Проверка: входной буфер пуст?
2C47:0160  A802    TEST   AL,02   ;если не пуст, то
2C47:0162  75FA    JNZ    015E    ;переход на новый цикл проверки.
2C47:0164  B020    MOV    AL,20   ;Передача адаптеру команды "чтение
2C47:0166  E664    OUT    [64],AL ;командного байта.
2C47:0168  E464    IN     AL,[64] ;Ожидание записи в вых. буфер результата
2C47:016A  A801    TEST   AL,01   ;выполнения команды, если буфер пуст, то
2C47:016C  74FA    JZ     0168    ;на новый цикл ожидания.
2C47:016E  E460    IN     AL,[60] ;Чтение командного байта из вых. буфера
2C47:0170  88C3    MOV    BL,AL   ;и помещение его в регистр BL.
2C47:0172  B060    MOV    AL,60   ;Передача адаптеру команды "запись
2C47:0174  E664    OUT    [64],AL ;командного байта".
2C47:0176  E464    IN     AL,[64] ;Ожидание признака "входной буфер пуст".
2C47:0178  A802    TEST   AL,02   ;Если не пуст, то
2C47:017A  75FA    JNZ    0176    ;на новый цикл ожидания.
2C47:017C  B047    MOV    AL,47   ;Стандартное значение командного байта
2C47:017E  E660    OUT    [60],AL ;передается в адаптер.
2C47:0180  B0AF    MOV    AL,AF   ;Отработка команды AFh
2C47:0182  E664    OUT    [64],AL ;
2C47:0184  E464    IN     AL,[64] ;
2C47:0186  A802    TEST   AL,02   ;
2C47:0188  75FA    JNZ    0184    ;
2C47:018A  B002    MOV    AL,02   ;с первым
2C47:018C  E660    OUT    [60],AL ;атрибутом 02h
2C47:018E  E464    IN     AL,[64] ;и вторым
2C47:0190  A802    TEST   AL,02   ;атрибутом 05h,
2C47:0192  75FA    JNZ    018E    ;что завершает передачу
2C47:0194  B005    MOV    AL,05   ;в клавиатуру
2C47:0196  E660    OUT    [60],AL ;блоков данных.
2C47:0198  FB      STI            ;Разрешение прерываний.
2C47:0199  CC      INT3           ;Остановка и передача управления AFD.
```

1.1.9. Проверка адаптера клавиатуры процедурой POST.

Для проверки корректности работы адаптера клавиатуры в процедуре POST предусмотрено - тестирование интерфейса 8042. Тестом проверяются входной и выходной буферные регистры, инициируется самотестирование адаптера, производится чтение входного порта и запись командного байта.

Адаптеры клавиатуры различаются версиями встроенного в их микроконтроллеры ПО, поэтому замена микроконтроллера другим, с отличающейся версией ПО (KBC BIOS), может оказаться проблематичной - микросхема BIOS системной платы должна знать особенности микроконтроллера адаптера клавиатуры.

В данном разделе использовались материалы из [4 - 6, 8, 11, 13, 17, 19].

Контрольные вопросы

1. Для чего используется подсистема клавиатуры и манипуляторов?
2. Через какое устройство подключаются к системе клавиатура и PS-Mouse?
3. Кратко охарактеризуйте системную поддержку клавиатуры на уровне BIOS.
4. Кратко охарактеризуйте системную поддержку PS-Mouse на уровне BIOS.
5. Какие номера векторов прерывания закреплены в системе за клавиатурой и PS-Mouse?
6. Как подключается к системе манипулятор Serial Mouse и какие системные ресурсы выделяются для этого подключения?
7. Как подключается к системе манипулятор джойстик?
8. Кратко охарактеризуйте интерфейс клавиатуры.
9. Охарактеризуйте упрощенную схему подключения клавиатуры и PS-Mouse к компьютеру (Рис. 1.1).
10. Как подключалась клавиатура к PC/XT?
11. Какие новые качества приобрел адаптер клавиатуры в архитектуре PC/AT?
12. Дайте краткое описание функционирования адаптера клавиатуры и PS-Mouse.
13. Охарактеризуйте блок-схему микроконтроллера адаптера клавиатуры PC/AT (Рис. 1.2).
14. Охарактеризуйте схему подключения микроконтроллера i8042, представленную на Рис. 1.3.
15. Охарактеризуйте интерфейс подключения адаптера клавиатуры к системе.
16. Дайте краткое описание интерфейса подключения клавиатуры и PS-Mouse к адаптеру.
17. Дайте краткое описание формата регистра состояния адаптера клавиатуры и PS-Mouse.
18. Охарактеризуйте регистр командного байта адаптера.
19. Дайте общую характеристику командам адаптера.
20. Охарактеризуйте дополнительные команды, поддерживаемые адаптером в режиме PS/2.
21. Дайте краткую характеристику портам ввода и вывода адаптера.
22. Кратко опишите действия адаптера по обслуживанию взаимодействия системы с клавиатурой и PS-Mouse.
23. Сформулируйте правила программного взаимодействия с адаптером.
24. Сформулируйте правила программного взаимодействия с клавиатурой через адаптер.
25. Сформулируйте правила программного взаимодействия с PS-Mouse через адаптер.
26. Сформулируйте правила программирования портов микроконтроллера адаптера.
27. Охарактеризуйте программу, представленную в листинге 1.1.
28. Дайте комментарии результатам выполнения команд адаптера, представленным в таблице раздела 1.1.8.3.1.
29. Поясните результаты, полученные при выполнении команд адаптера в рамках программы, представленной в листинге 1.2.
30. Поясните результаты, полученные при выполнении команд адаптера в рамках программы, представленной в листинге 1.3.
31. Поясните результаты, полученные при выполнении команд адаптера в рамках программы, представленной в листинге 1.4.
32. Поясните результаты, полученные при выполнении команд адаптера в рамках программы, представленной в листинге 1.5.
33. Поясните результаты, полученные при выполнении команд адаптера в рамках программы, представленной в листинге 1.6.
34. Поясните результаты, полученные при выполнении команд адаптера в рамках программы, представленной в листинге 1.7.
35. Как проверяется микроконтроллер адаптера клавиатуры процедурой POST?

1.2. Клавиатура

Клавиатура относится к периферийным устройствам системы ввода-вывода (СВВ) ПК и является основным средством взаимодействия пользователя с системой, представленной аппаратно-программными средствами ПК. В рамках интерфейса человек-машина такое устройство называется HID (Human Interface Device - устройство человеко-машинного интерфейса). Обмен данными между клавиатурой и оперативной памятью (ОП) системы осуществляется по программному каналу ввода-вывода, реализуемому средствами центрального процессора, аппаратными средствами СВВ и соответствующим программным обеспечением. На физическом уровне процессор взаимодействует с клавиатурой через систему внутренних и внешних интерфейсов СВВ и адаптер клавиатуры выполненный на базе микроконтроллера i8042 и расположенный на системной (материнской) плате ПК. На канальном уровне система и клавиатура обмениваются байтовыми сообщениями: со стороны клавиатуры передаются коды сканирования и коды ответов на команды; со стороны системы - коды команд и коды атрибутов команд. На уровне BIOS канал клавиатура - ОП поддерживается обработчиком INT 09h аппаратного прерывания IRQ 1 и драйвером клавиатуры (обработчик программного прерывания INT 16h). Для поддержки канала взаимодействия системы и клавиатуры последняя должна обладать соответствующими функциональными возможностями.

Для персональных компьютеров, совместимых с IBM PC, были разработаны три основных типа клавиатуры:

а) 83-клавишная клавиатура, появившаяся с IBM PC и использовавшаяся также в PC/XT (далее будем называть ее клавиатурой PC/XT). Клавиатура PC/XT имела следующие группы клавиш:

- основные клавиши (клавиатура пишущей машинки),
- функциональные клавиши (F1 - F10),
- клавиши управления курсором (клавиатура калькулятора),
- клавиши состояния и сдвигов (две клавиши Shift, клавиши Ctrl, Alt, Caps Lock, Num Lock и пр.).

Впоследствии к ним добавили индикаторы состояния Num Lock и Caps Lock, управляемые внутренним контроллером клавиатуры по нажатию соответствующих клавиш. Однако состояние этих индикаторов могло не быть синхронизированным с флагами в ОЗУ рабочей области BIOS, которыми пользуются драйверы;

б) 84-клавишная (клавиатура PC/AT), отличающаяся от клавиатуры PC/XT дизайном (изменились размер и расположение некоторых клавиш), клавишей Esc, клавишей SysRq и индикаторами Num Lock, Caps Lock и Scroll Lock. Двунаправленный интерфейс с системной платой позволяет программе корректно управлять индикаторами, а также программировать некоторые параметры клавиатуры и производить диагностику. Кроме того, адаптер клавиатуры стал обладать возможностями программирования, расширенной обработки ошибок, повторяемостью клавиш и некоторыми другими особенностями;

в) 101/102-клавишная клавиатура PS/2 (далее будем называть ее 101-клавишной клавиатурой). 101-клавишная клавиатура внешне существенно отличается от клавиатур PC/XT и PC/AT (рис. 1). На ней имеется множество новых клавиш: клавиши F11 и F12; правые клавиши Alt, Ctrl; вторая клавиша Enter; клавиша Pause; полностью продублированы клавиши управления курсором. Обычно на клавиатурах такого типа 101 клавиша, однако может появляться и 102-я в случае, если клавиатура должна обеспечивать переключение на национальный алфавит.

К новым возможностям 101-клавишных клавиатур можно отнести и программирование функций: только Нажатие, только Нажатие/Отпускание, только Повторение/Нажатие/Отпускание и только Повторение; расширенную обработку ошибок.

Кроме того, в 101-клавишной клавиатуре появилась возможность выбора одного из трех режимов работы (1, 2, 3), устанавливаемых командой клавиатуры F0h, посылаемой в клавиатуру центральным процессором через адаптер клавиатуры, расположенный на материнской плате компьютера. Первый режим имитирует работу клавиатуры PC/XT. Второй режим устанавливается по умолчанию (ниже будет рассмотрен в основном именно этот режим работы клавиатуры) и соответствует клавиатуре PC/AT. В этом режиме клавиатурой всегда выдается однобайтовый код нажатия и двухбайтовый код отпускания, состоящий из кода F0h и кода нажатия. Третий режим (режим PS/2) очень удобен при написании программ, интенсивно использующих клавиатуру и отличается тем, что у большинства клавиш не выдается код отпускания. Он генерируется только для клавиш сдвигов и состояния (Shift, Ctrl, Alt, Caps Lock, Num Lock) в виде двухбайтовой последовательности: кода F0h и кода нажатия.

Позже на базе 101/102-клавишной клавиатуры стали выпускаться клавиатуры с дополнительными клавишами: Microsoft Natural Keyboard (104 - 107 клавиш), Microsoft Internet Keyboard (добавлены клавиши управления браузером и почтовым клиентом, две программируемые "горячие" клавиши и кнопкой перевода в спящий режим) и Internet Keyboard Pro (добавлены клавиши управления функциями мультимедиа); общее число клавиш достигает 122.

На программном уровне клавиатура PC/AT, 101 и 122-клавишная клавиатуры совместимы между собой и обладают всеми возможностями клавиатуры PC/XT.

1.2.1. Конструктивы 101-клавишной клавиатуры

На рис.1.7 представлен внешний вид 101-клавишной клавиатуры Best Key. На общем поле клавиш в ней выделены клавиши основной, дополнительной, цифровой и функциональной клавиатур. Цифровая клавиатура (Numeric Keypad), при выключенном индикаторе Num Lock (или включенном Num Lock и нажатии Shift) используется для управления курсором и экраном. В дополнительной клавиатуре выделенные клавиши управления курсором и экраном дублируют эти функции цифровой клавиатуры. Клавиши SysRq и Break совмещены с клавишами Print Screen и Pause соответственно. В правом верхнем углу находятся светодиодные индикаторы режимов Num Lock, Caps Lock и Scroll Lock.

Рис.1.7 Общий вид 101-клавишной клавиатуры Best Key.

На рис.1.8 представлен вид клавиатуры без верхней защитной панели. В правом верхнем углу находится электронный модуль управления работой клавиатуры. На модуле располагаются следующие основные компоненты:

1. Разъем подключения интерфейсного кабеля клавиатуры.
2. Разъем подключения нижней группы пленочных контактов матрицы датчиков клавиш.
3. Разъем подключения верхней группы пленочных контактов матрицы датчиков клавиш.
4. Однокристальная микро-ЭВМ (микроконтроллер типа i8042) управляющая работой клавиатуры - контроллер клавиатуры.
5. Светодиодные индикаторы клавиатуры.

Рис. 1.8. Вид клавиатуры без верхней защитной панели.

На рис. 1.9 представлен вид 101-клавишной клавиатуры без блока клавиш. На этом рисунке цифрами выделены следующие компоненты:

1. Матрица упругих латексных элементов, обеспечивающих замыкание пленочных контактов датчиков при нажатие клавиш и их размыкание при отпускание.
2. Прозрачные гибкие пластиковые пластины, формирующие матрицу пленочных контактных датчиков и систему проводников, позволяющих контроллеру клавиатуры контролировать их состояние.

На рис. 1.10 представлен вид 101-клавишной клавиатуры без матриц упругих латексных элементов. Цифрой 1 на рисунке обозначен "сэндвич" из трех прозрачных гибких пластиковых пластин, формирующих матрицу датчиков клавиш. На верхней и нижней пластинах нанесены токопроводящие аппликации пленочных контактов датчиков клавиш и аппликации соединительных проводников. В средней пластине в местах расположения пленочных контактов нижней и верхней пластин отперфорированы круглые отверстия, позволяющие

замыкаться контактам при прогибающем воздействии на верхнюю пластину упругим латексным элементом нажимаемой клавиши.

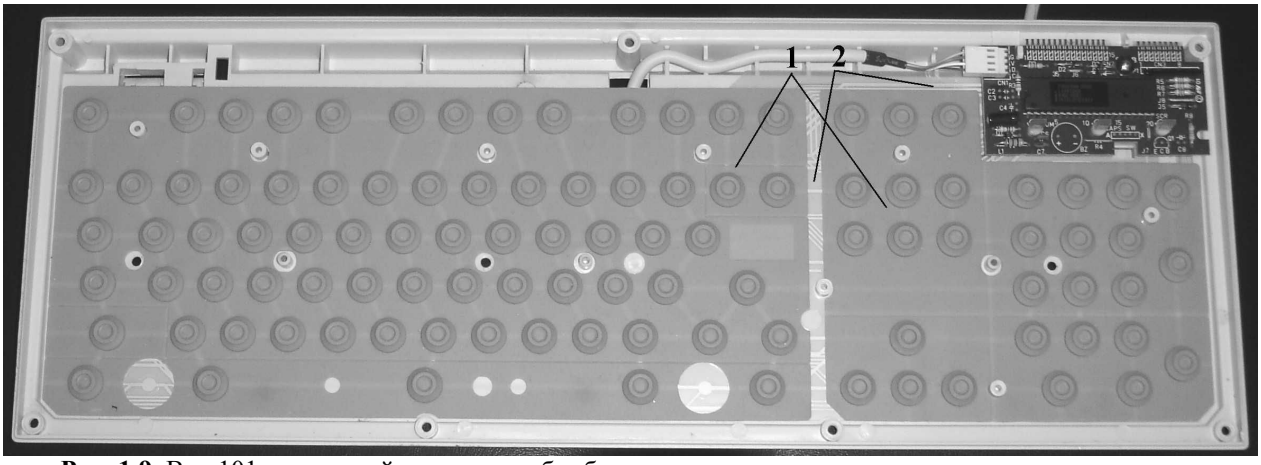

Рис. 1.9. Вид 101-клавишной клавиатуры без блока клавиш.

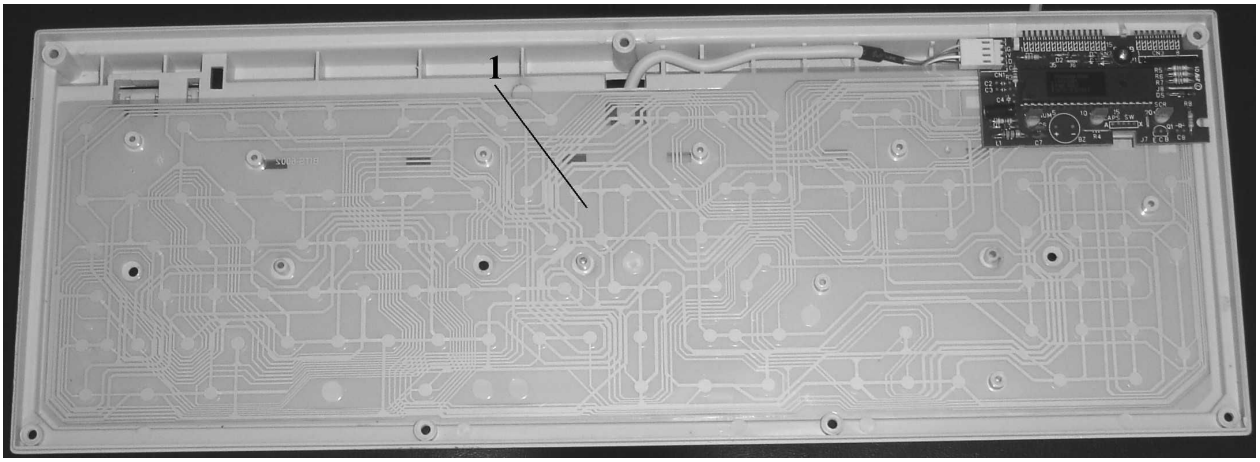

Рис. 1.10. Вид 101-клавишной клавиатуры без матриц упругих латексных элементов.

Отдельно вид "сэндвича" пластин представлен на рис. 1.11. Темный фон позволяет достаточно детально рассмотреть структуру матрицы датчиков клавиш. Цифрой 1 обозначены пленочные разъемные контакты, с помощью которых нижняя пластина матрицы датчиков подключается к разъему 2 электронного модуля управления клавиатурой, а цифрой 2 - пленочные разъемные контакты, обеспечивающие подключение к модулю управления верхней пластины матрицы датчиков клавиш. Цифрой 3 обозначены пленочные контакты матрицы датчиков клавиш, а цифрой 4 - пленочные проводники, соединяющие между собой группы контактов датчиков и соответствующие пленочные разъемные контакты.

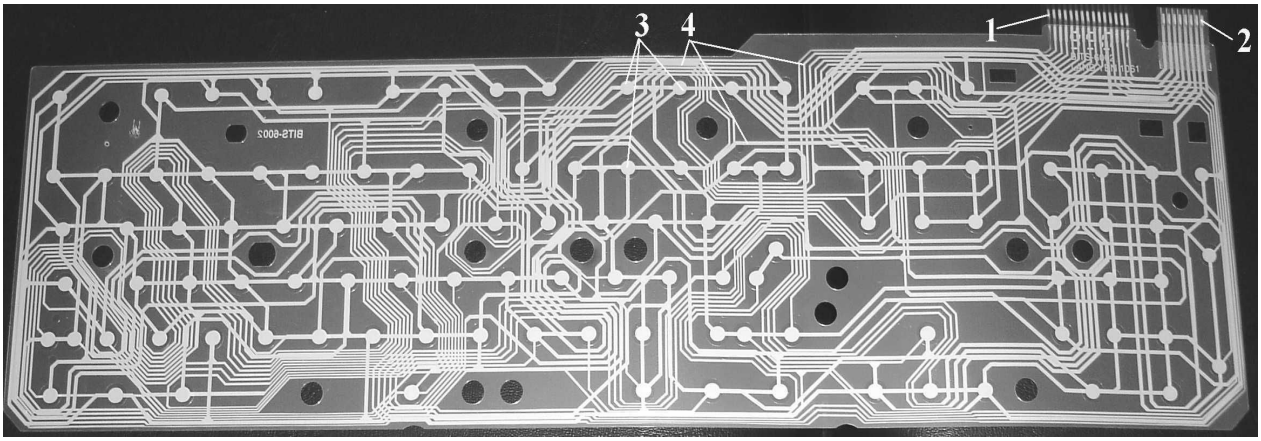

Рис. 1.11. "Сэндвич" из трех прозрачных гибких пластиковых пластин, формирующих матрицу датчиков.

На рисунках 1.12 и 1.13 представлены, соответственно, нижняя и верхняя пластины матрицы датчиков. Видно, что контакты датчиков нижней пластины объединены в 15 групп, каждая из которых подключается к соответствующему контакту разъема 2 блока управления. Аналогично контакты датчиков верхней пластины матрицы объединены в 8 групп, каждая из которых также подключается к блоку управления, но через

контакты разъема 3. Такая группировка контактов нижней и верхней пластин позволяет построить коммутационную матрицу размерностью 8 × 15. Это в свою очередь позволяет, при использование обычной схемы опроса матрицы, адресовать максимум 120 клавиш. В 101-клавишой клавиатуре используются не все узлы матрицы.

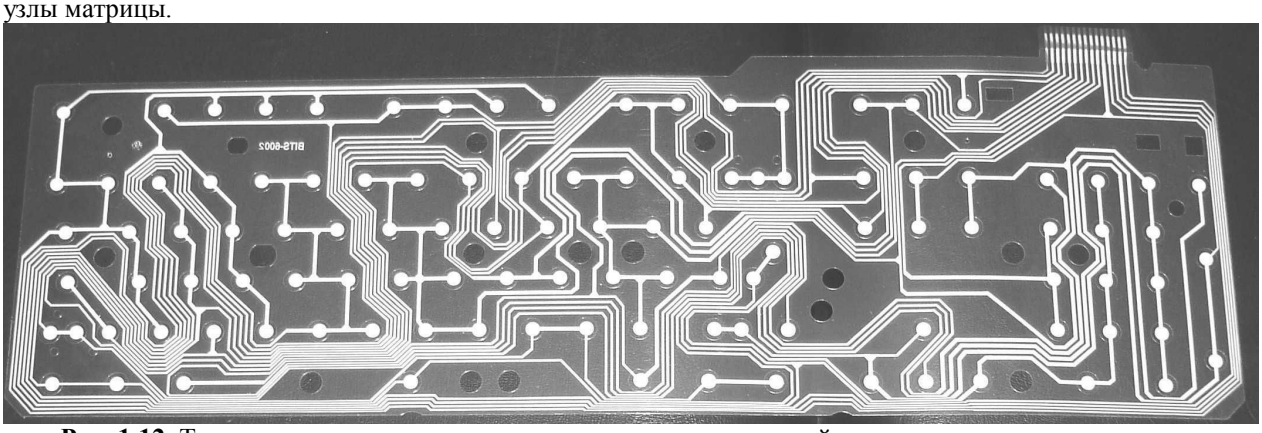

Рис. 1.12. Топология размещения контактов и проводников нижней пластины матрицы датчиков.

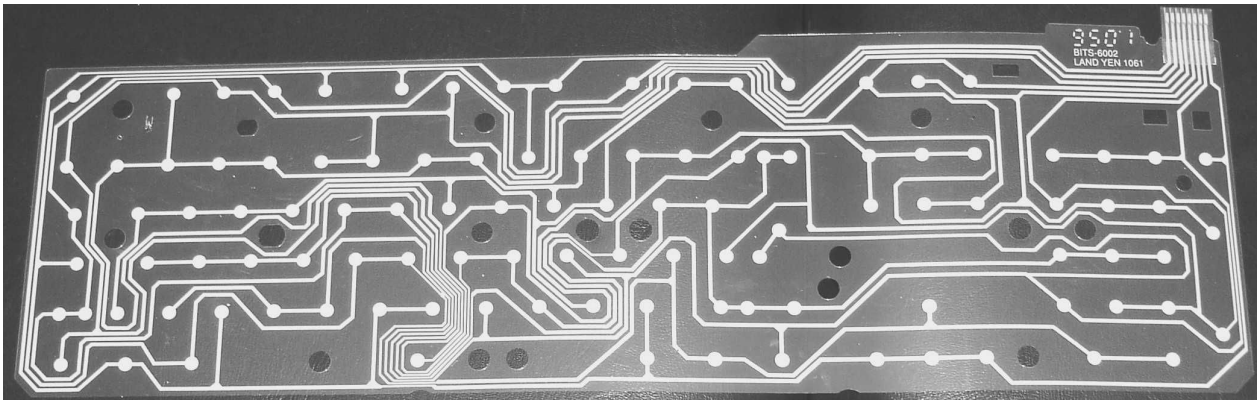

Рис. 1.13. Топология размещения контактов и проводников верхней пластины матрицы датчиков.

1.2.2. Блок-схема клавиатуры

На рис. 1.14. приведена блок-схема клавиатуры

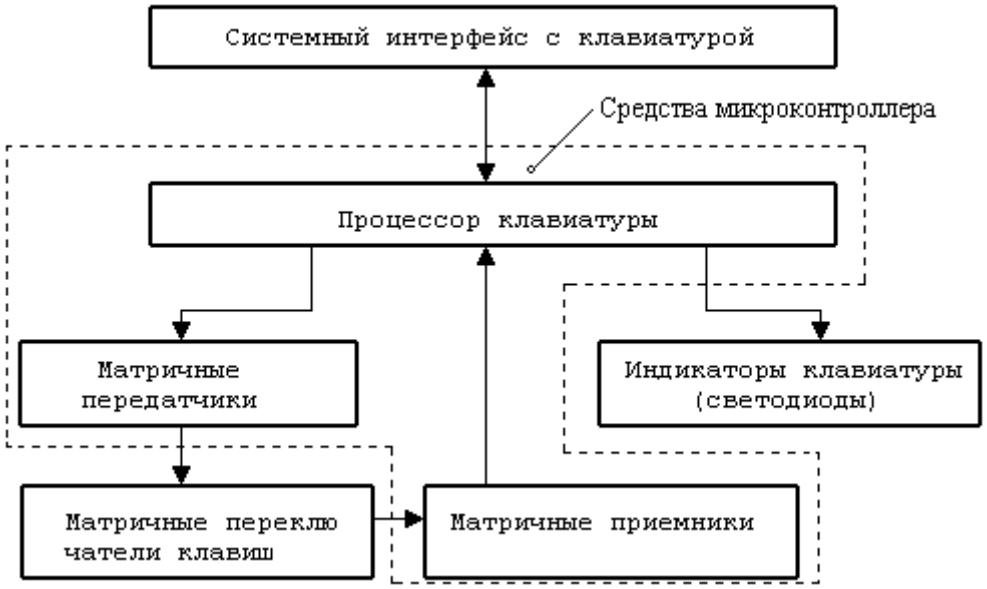

Рис. 1.14. Блок-схема клавиатуры

Клавиатура PC представляет собой унифицированное устройство со стандартным разъемом и последовательным двунаправленным (полудуплексным) интерфейсом связи с системной платой. В качестве датчиков нажатия клавиш применяют механические контакты (открытые, герконовые или пленочные), кнопки на основе токопроводящей резины, индуктивные или емкостные датчики и датчики на эффекте Холла. Типы клавишных датчиков влияют на надежность, долговечность и цену клавиатуры. Последние три типа датчиков являются самыми долговечными, поскольку в них исключены механические контактные

системы. Независимо от типов применяемых датчиков нажатия клавиш, все они объединяются в матрицу. В клавиатуре имеется микроконтроллер (однокристальная микро-ЭВМ) из семейства MCS-48 фирмы Intel (например, i8042) на базе которого выполнено устройство управления клавиатурой - контроллер клавиатуры. Этот контроллер осуществляет сканирование матрицы клавиш, управление индикаторами, внутреннюю диагностику и связь с системной платой (с адаптером клавиатуры). Связь осуществляется через последовательный интерфейс, включающий в себя двунаправленные линии Data и Clock. По линии Data передаются данные в последовательном формате, а по линии Clock производится синхронизация передачи этих данных. От системы в клавиатуру поступают команды управления клавиатурой, а системе из клавиатуры - коды сканирования и подтверждения. Данные сканирования вырабатываются клавиатурой при нажатие и отпускание (отжатие) клавиши. Эти коды вместе называются "кодами сканирования" (скан-коды), или "кодами клавиш".

Все действия по управлению клавиатурой и ее взаимодействию с системой процессор микроконтроллера клавиатуры выполняет под управлением программ, "прошитых" в ПЗУ микроконтроллера.

Буфер клавиатуры. 101-клавишная и более поздние клавиатуры имеют буфер, организованный в виде очереди (FIFO - "первый пришел, первый ушел"). Этот клавиатурный буфер может хранить 32 кода (нажатия и отпускания) в режиме 1, 16 кодов сканирования (код нажатия и префикс отпускания F0h) в режиме 2 и до 32 кодов нажатия в режиме 3. Скан-коды сохраняются в буфере до тех пор, пока система не будет готова их принять. Если буфер клавиатуры переполнен, то новый скан-код не генерируется, а в буфер помещается код переполнения (FFh в режиме 1, 00h в режимах 2 и 3), для которого в буфере постоянно резервируется место.

Клавиши. За исключением клавиши Pause все клавиши 101-122-клавишной клавиатуры работают на нажатие/отпускание и большинство из них являются повторяемыми (если клавиша удерживается нажатой длительное время, то после посылки кода нажатия контроллер клавиатуры, находящийся в клавиатуре, начинает посылать через одинаковые промежутки времени код нажатия этой клавиши). Скорость и задержка повторений могут быть изменены с помощью команды клавиатуры F3h.

Если удерживаются две и более клавиш, повторяется код только той клавиши, которая была нажата последней. После отпускания этой клавиши автоматическое повторение прекращается, даже если остальные клавиши остаются нажатыми. Если клавиша нажата и удерживается во время запрета клавиатуры, то в буфере сохраняется только первый код нажатия. Тем самым предотвращается переполнение буфера вследствие автоматического повторения.

1.2.3. Схемотехника клавиатуры

Ниже на рис. 1.15 приведена общая электрическая схема 101-клавишной клавиатуры фирмы X-RING. Основным элементом клавиатуры является микроконтроллер U1 (9306S). Аналогом данного микроконтроллера является однокристальная микросхема КМ1816ВЕ48. Так же на плате находится микросхема U2 (74 LS05). Аналогом данной микросхемы являебтся К555ЛН2 (шесть инверторов с открытым коллектором).

Компоненты общей электрической схемы.

В состав эклектической схемы клавиатуры входят следующие компоненты:

- Интерфейсный кабель, содержащий четыре линии:
- G -сигнальная земля;
- V -питание +5В;
- D -двунаправленная линия передачи данных в последовательном формате;
- C -двунаправленная линия передачи синхросигналов (линия синхронизации);
- R9, R10 -нагрузочные резисторы передатчиков двунаправленных линий синхронизации и данных, выполненных на инверторах с открытым коллектором;
- C9, C10 -конденсаторы фильтрации высокочастотных наводок на линиях данных и синхронизации;
- C1, C2, C7, C8 -конденсаторы фильтрации высокочастотных наводок в шинах питания (развязка по питанию);
- U2 -микросхема 74LS05 (К555ЛН2) - шесть инверторов с открытым коллектором: два инвертора используются для передачи сигналов по двунаправленным линиям интерфейса (линии данных (D) и линии синхронизации (C)); три - для управления светодиодами клавиатуры (SCR, CAP и NUM); один - для приема синхросигналов с двунаправленной линией синхронизации (C).
- D1-D5, D7 -диоды, реализующие логическую операцию "ИЛИ";
- X1 -кварцевый резонатор (5 мГц);
- R1-R8 -резисторы, обеспечивающие подачу напряжения смещения +5В на линии входного порта (D0-D7) микроконтроллера U1;
- CN2 -входной разъем подключения матрицы контактных датчиков клавиш;
- CN1 -выходной разъем подключения матрицы контактных датчиков клавиш;
- U1 -микроконтроллер (однокристальная микро-ЭВМ) типа 9306S (аналог i8042, КМ1816ВЕ48).

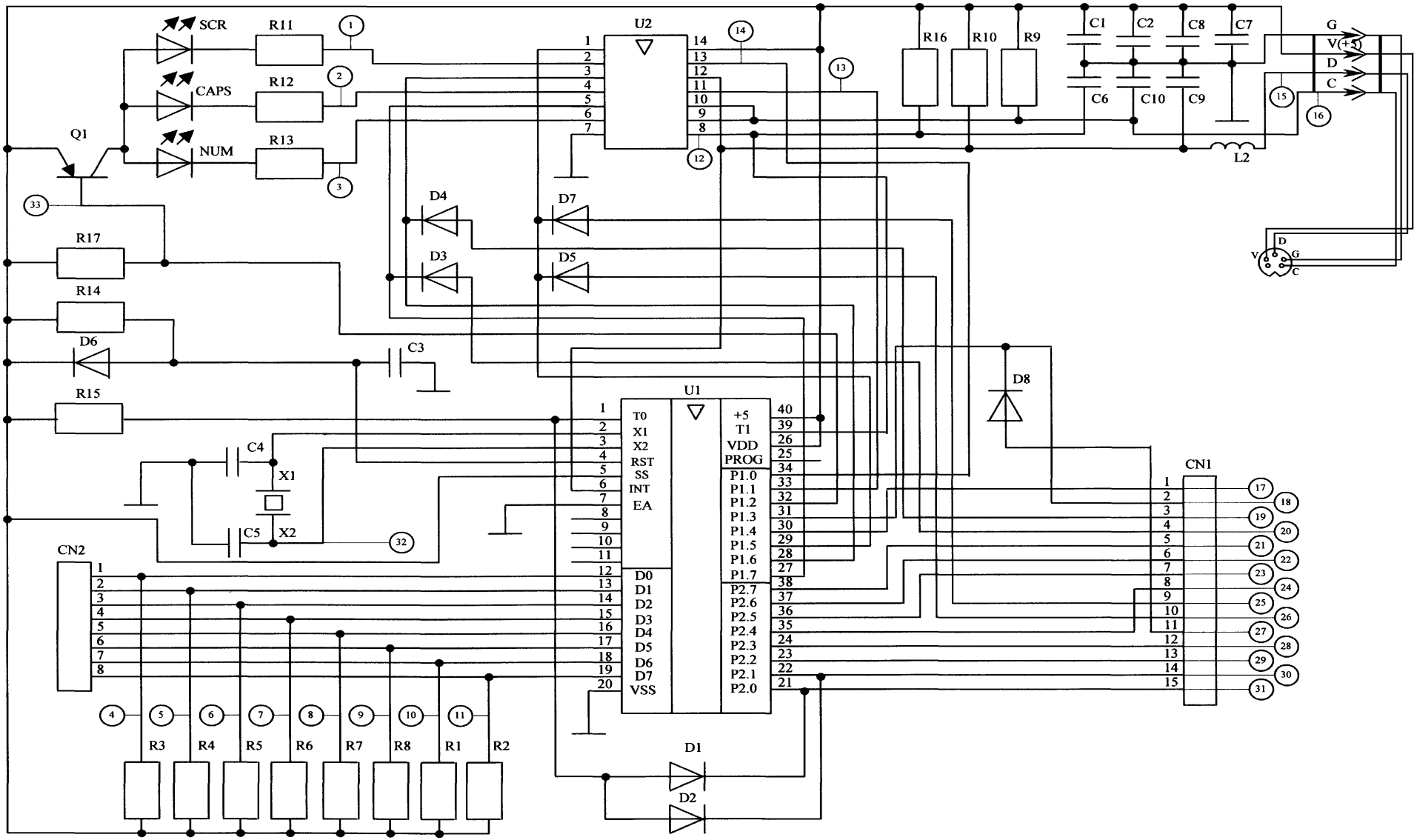

Рис. 1.15 Общая электрическая схема клавиатуры

Контрольные точки электрической схемы

1 - Сигналы включения светодиода Scroll Lock и импульсы опроса столбцов матрицы датчиков клавиш

2 - Сигналы включения светодиода Caps Lock и импульсы опроса столбцов матрицы датчиков клавиш..

3 - Сигналы включения светодиода Num Lock и импульсы опроса столбцов матрицы датчиков клавиш.

4, 5, 6, 7, 8, 9, 10, 11 - контакты на разъеме CN2, 8 - битный двунаправленный порт ввода-вывода информации (порт D0-D7 микроконтроллера U1) используются для приема сигналов со строк матрицы датчиков клавиш.

12 - Инвертированный входной сигнал линии синхронизации интерфейса (вывод 8 микросхемы U2)

13 - Инвертированный выходной сигнал синхронизации интерфейса (вывод 33 микросхемы U1)

14 - Инвертированный выходной сигнал данных интерфейса (вывод 34 микросхемы U1)

15 - Сигнал на линии данных интерфейса (KB-Data)

16 - Сигнал на линии синхронизации интерфейса (KB-Clock).

17, 18, 21-24, 28-31 - Сигналы на контактах разъема CN1, непосредственно подключенные к квази двунаправленным портам ввода-вывода микроконтроллера U1, и используемые для опроса столбцов матрицы датчиков клавиш (10 столбцов).

19, 20, 26 - Сигналы на контактах разъема CN1, подключенных к выводам (27 - 29) квази двунаправленного порта ввода-вывода P1 микроконтроллера U1 через разделительные диоды D3 - D5, и используемые для опроса столбцов матрицы датчиков клавиш (3 столбца). Эти выводы порта P1 управляют также через U2 светодиодами SCR, CAP и NUM.

25 - Сигнал опроса клавиши правый Enter, подключаемый диодом D7 с выхода 29 U1 к контакту 9 разъема CN1 при нажатие этой клавиши.

27 - Сигнал опроса клавиши Page Down, подключаемый диодом D8 с выхода 31 U1 к контакту 11 разъема CN1 при нажатие этой клавиши.

32 - Сигнал на кварцевом резонаторе (вывод 3 микроконтроллера U1).

33 - Сигнал управления транзистором Q1 цепи питания светодиодов (вывод 32 микроконтроллера U1).

1.2.4. Интерфейс клавиатуры

Как уже упоминалось для персональных компьютеров, совместимых с IBM PC, были разработаны три основных типа клавиатуры: 83-клавишная клавиатура PC/XT, 84-клавишная клавиатура PC/AT и расширенные (enhanced) клавиатуры AT или PS/2, имеющие более 100 клавиш. В настоящее время используется в основном третий тип клавиатуры или совместимые с ней. По электрическому интерфейсу клавиатуры XT и AT совпадают, за исключением того, что двунаправленный интерфейс позволяет клавиатуре AT принимать команды от системной платы. Однако по логическому интерфейсу они несовместимы (клавиатура AT иногда имеет переключатель режима XT/AT). Клавиатура PS/2 отличается от AT только исполнением разъема, при необходимости можно использовать переходник. С распространением шины USB появились клавиатуры и с этим интерфейсом; они имеют и встроенный хаб, например, для подключения мыши USB. Клавиатура USB питается от шины. Для клавиатуры USB требуется специальная поддержка со стороны BIOS; она имеется в современных системных платах. Большинство клавиатур с интерфейсом USB могут поддерживать интерфейс клавиатуры PS/2. Для подключения таких клавиатур к разъему PS/2 используются переходники.

Как упоминалось в описании подсистемы адаптера клавиатуры, для подключения клавиатуры к системе используется последовательный двунаправленный интерфейс, включающий в себя две обязательные линии сигналов KB-Data и KB-Clock. Обе линии на системной плате подтягиваются резисторами к шине +5 В. На обеих сторонах интерфейса выходные сигналы низкого уровня формируются выходами элементов с открытым коллектором (стоком), а состояние линий может быть прочитано через входные линии микроконтроллеров. Вид разъемов (со стороны задней панели) и назначение контактов приведены на рис. 1.16. Конструктивно возможны два варианта разъема — обычная 5-контактная розетка DIN (клавиатура AT) или малогабаритная розетка mini-DIN (PS/2). На этот же разъем через плавкий предохранитель поступает напряжение питания клавиатуры +5В.

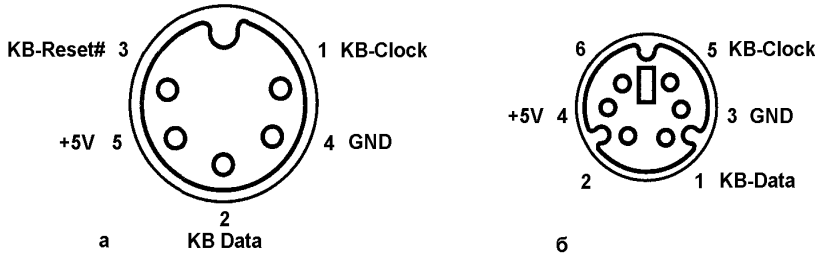

Рис.1.16. Разъемы (вид со стороны контактов) подключения клавиатур: *а* - AT и *б* - PS/2

Процессор общается с клавиатурой через *адаптер клавиатуры,* выполненный на базе микроконтроллера i8042 или совместимого с ним, и установленный на системной плате. Для обмена информацией в основном используется порт 60h, из которого принимаются скан-коды и через который

передаются команды клавиатуре. О необходимости чтения скан-кода адаптер клавиатуры сигнализирует процессору через аппаратное прерывание IRQ1, сигнал которого вырабатывается по

1.2.5. Функционирование клавиатуры

Каждому событию в клавиатуре, сопровождаемому посылкой байта в адаптер клавиатуры (нажатию и отпусканию клавиши, приему команд или ее атрибута и т.п.). Задание параметров автоповтора, выбор таблиц скан-кодов, управление светодиодными индикаторами, а также управление режимом сканирования матрицы клавиш и запуск диагностического теста осуществляется командами, посылаемыми клавиатуре через этот же порт. Адаптер клавиатуры транслирует команды в посылки, направляемые к клавиатуре через последовательный двунаправленный интерфейс.

Описание протоколов взаимодействия системы и клавиатуры приводится ниже в разделе "Функционирование клавиатуры".

На системной плате PC/XT адаптера клавиатуры на базе микроконтроллера i8042 не было, а интерфейс клавиатуры (однонаправленный) был реализован аппаратной логикой — регистром сдвига, параллельный выход которого подключается к входам порта A системного интерфейса i8255. По приему байта от клавиатуры вырабатывалось аппаратное прерывание IRQ1, обработчик которого читал принятый байт из порта 60h. С помощью разрядов 7 и 6 порта 61h была возможна программная блокировка и сброс клавиатуры PC/XT соответственно. Сброс клавиатуры XT осуществлялся обнулением линии KB-Clock.

При включение питания контроллер клавиатуры, находящийся в блоке управления клавиатуры, генерирует сигнал "сброс по включению питания" (POR) на время от 150 мс до 2,0 с, после чего переходит к выполнению базового теста.

1.2.5.1. Базовый тест

Базовый тест (BAT) состоит из теста процессора контроллера, контрольной суммы ПЗУ (ROM) и теста ЗУПВ (RAM). Во время процедуры BAT сигналы на линиях Clock и Data игнорируются. Светодиоды включаются в начале BAT и выключаются в конце BAT. BAT занимает от 300 до 500 мс (не считая времени, требуемого POR).

После успешного завершения BAT в систему передается код завершения (AAh) и начинается сканирование клавиатуры. При возникновение ошибки BAT клавиатура передает системе код ошибки (FDh). После этого клавиатура прекращает сканирование и ожидает ввода команды. Коды завершения посылаются через 0,45-2,0 с после POR и через 300-500 мс после подтверждения команды RESET.

Немедленно после POR клавиатура регистрирует сигналы на линиях Clock и Data и устанавливает протокол линии. Схема связей клавиатуры представлена на рис. 1.17.

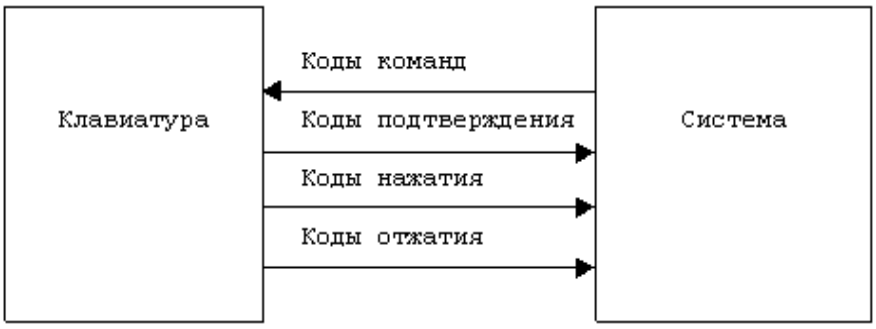

Рис. 1.17. Схема связей клавиатуры и системы

1.2.5.2. Синхросигналы и сигналы данных

Как уже упоминалось, клавиатура и система общаются между собой посредством линий синхросигналов Clock и сигналов данных Data. Источником сигналов на каждой из этих линий служат устройства с открытым коллектором, расположенные как в адаптере клавиатуры, так и на клавиатуре, которые позволяют клавиатуре или системе установить данные линии на неактивный (низкий) уровень. Когда никакого обмена данными не происходит, линия синхросигнала находится на активном (высоком) уровне. Состояние линии данных поддерживается на активном (высоком) уровне клавиатурой.

Когда система посылает данные на клавиатуру, она устанавливает линию данных на неактивный (низкий) уровень и позволяет линии синхросигнала быть на активном уровне.

Линия синхросигнала клавиатуры обеспечивает синхросигналы, которые используются для синхронизации последовательности передачи данных с клавиатуры и на клавиатуру.

Когда клавиатура передает данные системе или получает данные от системы, она генерирует синхросигнал для синхронизации данных. Система может запретить клавиатуре передавать данные, установив линию синхросигнала на неактивный (низкий) уровень. При этом линия данных может быть активной или не активной.

Во время BAT клавиатура разрешает линиям синхросигнала и данных быть активными.

1.2.5.3. Поток данных

Передача данных на клавиатуру и с клавиатуры осуществляется потоком 11-разрядных данных (режим 2 и 3), которые передаются последовательно по линии данных. При активном (высоком) уровне пересылается логическая 1. Назначение разрядов показано в табл. 1.3.

Разряд паритета может содержать 1 или 0, так что бы 8 разрядов данных вместе с разрядом паритета всегда содержали нечетное число единиц.

В режиме 1 передача данных осуществляется потоком 9-разрядных данных, в которых отсутствуют разряд паритета и стоповый разряд.

Таблица. 1.3. Назначение разрядов

Разряд	Назначение
1	Стартовый (всегда 0)
2	Разряд данных 0 (младший)
3	Разряд данных 1
4	Разряд данных 2
5	Разряд данных 3
6	Разряд данных 4
7	Разряд данных 5
8	Разряд данных 6
9	Разряд данных 7 (старший)
10	Разряд паритета
11	Стоповый (всегда 1)

1.2.5.4. Обмен данными

Обмен данными осуществляется между системой и клавиатурой, при этом на физическом уровне подразумевается обмен данными между двумя микроконтроллерами типа i8042: одним в системе (адаптер клавиатуры), другим в клавиатуре (контроллер клавиатуры).

1.2.5.4.1. Ввод данных с клавиатуры

Когда клавиатура готова к передаче данных, она сначала проверяет состояние линий синхросигнала и данных, чтобы определить, нет ли запрета клавиатуры, или, запрашивает ли система передачу. Если линия синхросигнала не активна (низкий уровень), данные сохраняются в буфере клавиатуры. Если линия синхросигнала активна (высокий уровень), а линия данных не активна (системой сформирован запрос на передачу и стартовый разряд START BITc), данные сохраняются в буфере клавиатуры, а клавиатура переходит в режим приема данных от системы, осуществляя синхронизацию этого процесса. Если линии синхронизации и данных активны (высокий уровень), клавиатура начинает генерировать синхросигналы и передает нулевой стартовый разряд (START BITk), 8 разрядов данных, разряд паритета PARITY и стоповый разряд STOP (см. диаграмму на рис. 1.18б). Данные будут действительны от заднего фронта и до переднего фронта синхроимпульса. Период Tcy синхроимпульса не менее 60 мкс, отрицательный полу период Tcl не менее 5 мкс. Клавиатура изменяет состояние линии данных за время Thk (минимальное значение 0 мкс) до формирования заднего фронта синхросигнала. Во время передачи клавиатура проверяет линию синхросигнала по меньшей мере через каждые 60 мкс. Если после начала передачи данных с клавиатуры система переводит линию синхросигнала на неактивный уровень, имеет место состояние, которое называется конфликтной ситуацией на линии, и клавиатура прекращает передачу данных. Если конфликтная ситуация возникает до переднего фронта 10-го синхроимпульса (разряд паритета) контроллер клавиатуры возвращает передававшийся символ в буфер клавиатуры и возвращает линии синхросигнала и данных на активный уровень. Если до 10-го синхроимпульса конфликтной ситуации не возникает, клавиатура завершает передачу. После конфликтной ситуации система может запросить у клавиатуры повторную посылку данных.

После приема данных от клавиатуры система может заблокировать клавиатуру на время обработки поступивших данных или пока система не запросит ответ от клавиатуры (участок [6] временной диаграммы). При работе в MS DOS после ввода очередного кода с клавиатуры она блокируется системой путем удержания линии синхронизации в пассивном состоянии примерно на 400 мкс. Этот сигнал может рассматриваться интерфейсом клавиатуры как сигнал подтверждения приема системой информации от клавиатуры (ACKc).

В PC/AT блокировка линии синхронизации может осуществляться программно командой адаптера клавиатуры ADh "Запретить клавиатуру", а разблокирование - командой AEh "Разрешить клавиатуру". В PC/XT эти операции осуществляются установкой и сбросом разряда 7 порта 61h.

Временная диаграмма передачи данных с клавиатуры в систему показана на рис. 1.18б.

1.2.5.4.2. Вывод данных в клавиатуру

Когда система готова к передаче данных в клавиатуру, она сначала проверяет, не передает ли клавиатура данные. Если клавиатура передает данные, но еще не передала 10-й синхросигнал, система может заблокировать вывод данных с клавиатуры путем установки линии синхросигнала клавиатуры на

неактивный (низкий) уровень. Если клавиатура передала 10-й синхроимпульс, система должна принять передаваемые данные.

Если клавиатура не передает данные или если клавиатура передает данные, но еще не передала 10-й синхросигнал, система устанавливает линию синхросигнала клавиатуры на неактивный уровень на время более 60 микросекунд (мкс) и готовится в это время к передаче данных. Когда система готова к передаче (время T1 на рис. 1.18а), она устанавливает стартовый разряд START BITc (линия данных не активна) а затем через время T2 разрешает линии синхросигнала подняться на активный уровень.

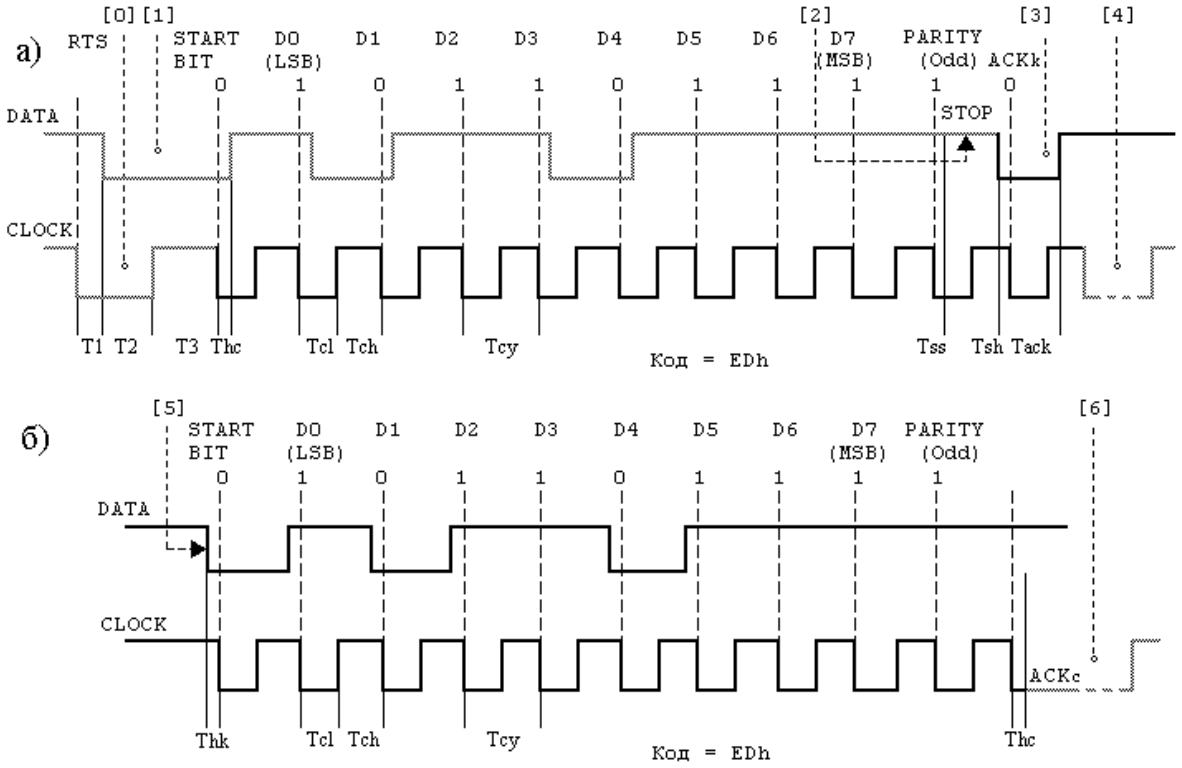

Рис. 1.18. Временные диаграммы связи клавиатуры и системы: а) - передача данных системой в клавиатуру; б) - передача данных клавиатурой в систему (серым цветом обозначены сигналы, формируемые системным адаптером клавиатуры, черным - клавиатурой).

Фазы приема-передачи:

[0] - CLOCK - совместно с низким уровнем линии DATA рассматривается клавиатурой как сигнал RTS - запрос на передачу данных от системы в клавиатуру;

[1] - DATA - сигнал низкого уровня, формируемый системой как стартовый разряд передаваемых в клавиатуру данных;

[2] - DATA - сигнал высокого уровня, формируемый системой как стоповый разряд STOP;

[3] - DATA - сигнал низкого уровня, устанавливаемый клавиатурой как сигнал ACKk подтверждения приема данных от системы;

[4] - CLOCK - сигнал низкого уровня, формируемый системой как сигнал запроса на передачу клавиатурой байта подтверждения приема байта команды или байта ее атрибута;

[5] - DATA - сигнал низкого уровня, формируемый клавиатурой как стартовый разряд START BITk при передаче данных с клавиатуры в систему;

[6] - CLOCK - сигнал низкого уровня, формируемый системой после приема очередного байта данных с клавиатуры; может использоваться клавиатурой, как сигнал подтверждения приема системой очередного байта данных с клавиатуры (ACKc).

В этом режиме клавиатура проверяет состояние линии синхросигнала с интервалами не более 10 мкс. Если обнаруживается состояние запроса передачи (RTS), клавиатура переходит в состояние считывания разрядов, передаваемых системой по линии данных. Считывание начинается после завершения состояния RTS формированием отрицательного фронта сигнала CLOCK (время T3). Клавиатура считывает 10 разрядов (START BITc, 8 битов данных и бит паритета PARITY). После этого клавиатура проверяет уровень линии данных и, если эта линия активна (системой сформирован стоповый разряд STOP), переводит ее на неактивный уровень (формирует сигал ACKk) и считывает еще один разряд. Это указывает системе, что клавиатура приняла переданные данные. После этого *адаптер клавиатуры* (система) формирует импульс KB-Clock длительностью не менее 60 мкс, который является запросом на прием ответа от клавиатуры (состояние [4] на временной диаграмме рис 1.18а). Адаптер ожидает окончания ответа на этот запрос не

долее 20 мс и, если ответ не придет за это время, сформирует ошибку тайм-аута. Ошибка будет также в случае, если клавиатура не введет первый синхроимпульс за 15 мс от начала запроса или адаптер не примет данные, включая стоп-бит, за 2 мс с момента появления синхроимпульса бита 0.

Если после чтения 10-го разряда обнаруживается неактивное состояние линии данных, это значит, что возникла ошибка кодирования, и клавиатура продолжает попытки чтения до тех пор, пока линия данных не станет активной. После этого клавиатура понижает уровень линии данных и передает системе команду RESEND (FEh).

При каждой передаче команды или данных от системы в клавиатуру система требует подтверждения от клавиатуры (код ответа FAh - логическое ACKk). Если только система не запретит вывод данных с клавиатуры, последняя дает этот ответ в течение 20 мс.

Если ответ клавиатуры недействителен или содержит ошибку паритета, система повторно посылает ей команду. Однако двухбайтовые команды требуют особой обработки. Если были переданы и подтверждены команды F3h (установить скорость/задержку повторения), F0h (выбрать альтернативные коды сканирования) или EDh (установить/сбросить индикаторы режима), а также передан байт данных (атрибут команды), но ответ оказался недействительным или с ошибкой паритета, то система передаст повторно и команду и ее атрибут.

Для определения конкретных временных параметров сигнального протокола интерфейса клавиатуры можно воспользоваться простой программой (см. листинг KLEE1.TXT), выполняемой в среде отладчика DOS (Debug или AFD). В программе реализовано циклическое выполнение команды клавиатуры "Эхо" (код EEh, см. раздел 1.2.6.2.2). Код команды EEh передается через интерфейс в последовательном формате в клавиатуру, которая в ответ передает адаптеру клавиатуры код ответа EEh в том же формате. Подключившись осциллографом к линиям KB-Clock и KB-Data интерфейса клавиатуры можно фиксировать электрические сигналы, реализующие передачу этих кодов на физическом уровне в соответствие с сигнальным протоколом аппаратного интерфейса клавиатуры. На рис. 1.19 представлены осциллограммы этих сигналов на линиях интерфейса клавиатуры.

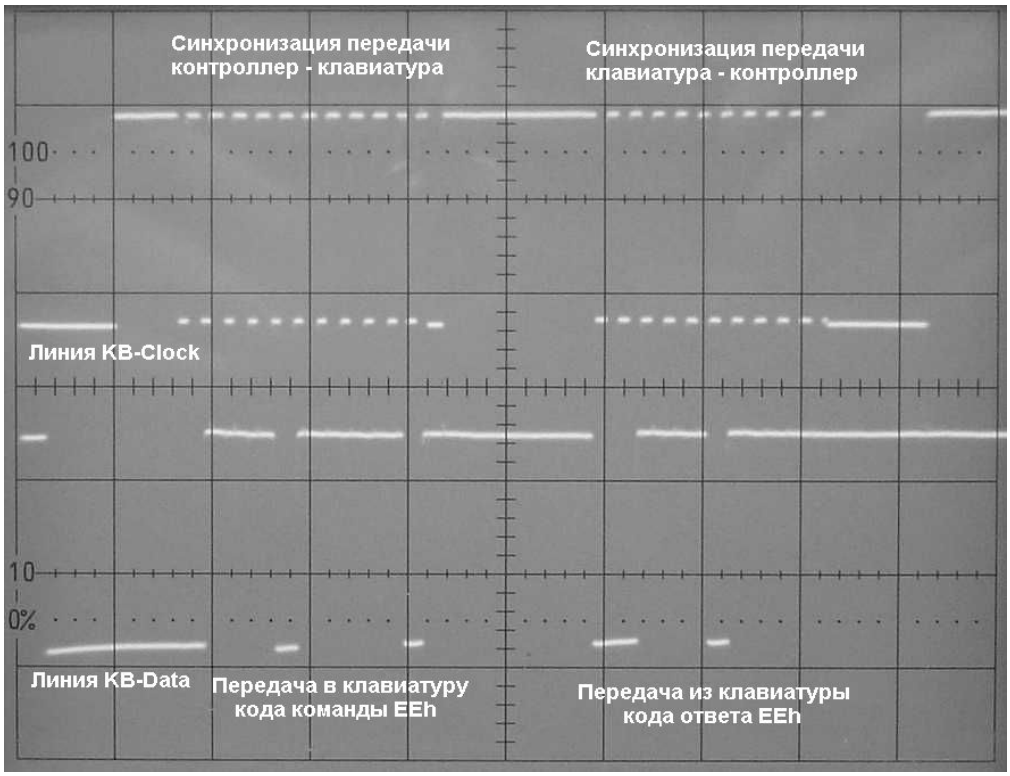

Рис. 1.19. Осциллограммы сигналов интерфейса клавиатуры зарегистрированные при выполнении программы KLEE1.BIN: развертка - 0,36 мс/дел., чувствительность по напряжению - 2 В/дел.; развертка запускается отрицательным фронтом сигнала линии KB-Clock.

Листинг KLEE1.TXT. Циклическое выполнение команды клавиатуры "Эхо" (код EEh).

```
2321:0100  FA        CLI           ;Запрет аппаратных прерываний.
2321:0101  BB0001    MOV    BX,0100;Число циклов выполнения программы.
2321:0104  E464      IN     AL,[64];Чтение рег. состояния адаптера и
2321:0106  A802      TEST   AL,02  ;проверка: Вх. буфер пуст? Если нет,
2321:0108  75FA      JNZ    0104   ;то новый цикл проверки.
2321:010A  B0EE      MOV    AL,EE  ;Передача кода EEh в клавиатуру
2321:010C  E660      OUT    [60],AL;через порт 60h.
```

```
2321:010E  E464      IN     AL,[64] ;Чтение рег. состояния адаптера и
2321:0110  A801      TEST   AL,01   ;проверка признака "вых. буфер полон":
2321:0112  74FA      JZ     010E    ;если пуст, то на новый цикл.
2321:0114  E460      IN     AL,[60] ;Чтение кода ответа клавиатуры из вых.
2321:0116  88C4      MOV    AH,AL   ;буфера и запись его в регистр AH.
2321:0118  B0AE      MOV    AL,AE   ;Команда "разрешить интерфейс клавиат."
2321:011A  E664      OUT    [64],AL ;передается в адаптер (код AEh).
2321:011C  B90F00    MOV    CX,000F ;Организация
2321:011F  51        PUSH   CX      ;вложенного
2321:0120  B9FFFF    MOV    CX,FFFF ;цикла
2321:0123  E2FE      LOOP   0123    ;программной
2321:0125  59        POP    CX      ;задержки
2321:0126  E2F7      LOOP   011F
2321:0128  4B        DEC    BX      ;Уменьшение счетчика циклов BX.
2321:0129  75D9      JNZ    0104    ;Если BX не 0, то на новый цикл прогр.
2321:012B  FB        STI            ;Разрешение маскируемых прерываний.
2321:012C  CC        INT3           ;Остановка и передача упр. отладчику.
```

1.2.5.5. Режимы работы клавиатуры

У 101-клавишной клавиатуры есть три режима работы - 1, 2 и 3. Каждый из этих режимов может быть выбран с помощью команд от системы. Эти режимы предоставляют широкий диапазон функциональных возможностей и совместимости для различных прикладных систем.

Контроллер клавиатуры генерирует фиксированный набор кодов нажатия и отпускания для каждой из клавиш на клавиатуре. При нажатие клавиши контроллер клавиатуры посылает в систему код нажатия этой клавиши.

В режиме 1 клавиатура генерирует однобайтовый код отпускания, состоящий из кода нажатия плюс 80h. Когда клавиатура работает в режимах 2 и 3, код отпускания представляет собой двухбайтовый код - код F0h и код нажатия клавиши.

Режим 1

Коды сканирования, вырабатываемые расширенной клавиатурой в режиме 1, совместимы с кодами сканирования клавиатуры PC/XT. Поскольку никакого преобразования кодов сканирования в этом режиме не происходит, коды сканирования, генерируемые клавиатурой в режиме 1, идентичны системным кодам, требуемым для ввода в BIOS. Для установки режима 1 необходимо переслать в клавиатуру команду F0h с байтом конфигурации 01h. Чтобы получать системные коды и информацию о состоянии клавиатуры, прикладные системы, использующие клавиатуру в режиме 1, должны пользоваться функциями 00h, 01h и 02h прерывания INT 16h BIOS (драйвера клавиатуры).

В наборе кодов сканирования режима 1 каждой клавише присваивается базовый код сканирования, а в некоторых случаях и дополнительные коды для генерирования в системе искусственных состояний переключения регистров. Коды сканирования автоматического повторения каждой клавиши идентичны одиночным кодам сканирования.

Режим 2

Режим 2 - это устанавливаемый по умолчанию режим расширенной клавиатуры, который система выбирает при инициализации после включения питания. В этом режиме клавиатура генерирует скан коды нажатия, отличающиеся от системных кодов, а скан код отпускания состоит из двух байт, первый из которых является префиксом отпускания (F0h), а второй идентичен коду нажатия этой клавиши. В этом режиме системный адаптер клавиатуры должен программироваться на преобразование генерируемых клавиатурой скан кодов в системные коды, требуемые BIOS. За исключением системных кодов новых клавиш расширенной клавиатуры системные коды соответствуют кодам сканирования, которые генерируются клавиатурой в режиме 1. Для доступа к системным кодам, генерируемым новыми клавишами расширенной клавиатуры, прикладные системы должны пользоваться функциями 10h, 11h и 12h прерывания INT 16h BIOS вместо обычных функций INT 16h с AH=00h, 01h и 02h. Для доступа к системным кодам, генерируемым новыми клавишами 122-клавишной клавиатуры, прикладные системы должны пользоваться функциями BIOS INT 16h с AH=20h, 21h и 22h.

Режим 3

Набор кодов сканирования, генерируемых в режиме 3, отличается от кодов режимов 1 и 2. В этом режиме адаптеру клавиатуры должно быть запрещено преобразование кодов сканирования в системные коды поскольку адаптер не может преобразовать этот набор кодов сканирования. Прикладные системы, использующие расширенную клавиатуру в режиме 3, должны выбирать этот режим с помощью команды клавиатуры F0h (значение байта конфигурации - 3). Эти прикладные системы должны взять на себя обработку кодов сканирования, генерируемых клавиатурой, поскольку адаптер клавиатуры и BIOS не могут обрабатывать набор кодов сканирования, генерируемых в режиме 3.

В наборе кодов сканирования 3 каждой клавише присваивается уникальный 8-разрядный код нажатия, который передается при нажатии клавиши. Генерацию кода отпускания можно программно разрешать или запрещать для всех или отдельных обычных клавиш. Код отпускания формируется всегда для клавиш сдвига и состояния (Shift, Ctrl, Alt, Caps Lock, Num Lock). Код отпускания, как и в режиме 2, состоит из двух байтов, первый из которых является префиксом отпускания (F0h), а второй идентичен коду нажатия данной клавиши. В режиме 3 каждая клавиша посылает только один код сканирования и состояние любой клавиши не зависит от состояния других.

1.2.5.6. Скан-коды

Скан-коды передаются от клавиатуры в компьютер по фактам нажатия и отпускания клавиш. При нажатие клавиши передается ее скан-код — номер, идентифицирующий ее расположение на клавиатуре. При отпускание клавиши клавиатура XT передает ее скан-код, увеличенный на 80h, а клавиатура AT передает 2 байта: в первом содержится префикс F0h, во втором — скан-код. Прием байта от клавиатуры вызывает аппаратное прерывание IRQ1 (номер вектора 09h). Расширенная клавиатура позволяет выбирать один из 3 наборов скан-кодов, по умолчанию устанавливается Set#2 — совместимый с AT-84. Скан-коды нажатия и отпускания трех возможных наборов, поступающие с клавиатуры на вход адаптера клавиатуры, расположенного на системной (материнской) плате ПК приведены в табл. 1.4 - 1.9. Их можно прочитать из порта 60h адаптера клавиатуры, если в командном байте адаптера запретить преобразование микроконтроллером адаптера скан-кодов в системные коды. В режиме 3 (Set#3) не была разрешена генерация кодов отпускания. Интерпретацией скан-кода, который считывается из порта 60h адаптера клавиатуры, занимается программа основного процессора (например, обработчик прерывания INT 09h).

Таблица 1.4. Скан-коды основной клавиатуры

Клавиша	Set#l H / O	Set#2 Наж. / Отп.	Set#3 Наж. / Отп.	Клавиша	Set#l H / O	Set#2 Наж. / Отп.	Set#3 Наж. / Отп.
1 !	02/82	16/0F,16	16	A	1E/9E	1C/F0,1C	1C
2 @	03/83	1E/0F,1E	1E	S	1F/9F	1B/F0,1B	1B
3 #	04/84	26/0F,26	26	D	20/A0	23/F0,23	23
4 $	05/85	25/0F,25	25	F	21/A1	2B/F0,2B	2B
5 %	06/86	2E/0F,2E	2E	G	22/A2	34/F0,34	34
6 ^	07/87	36/0F,36	36	H	23/A3	33/F0,33	33
7 &	08/88	3D/0F,3D	3D	J	24/A4	3B/F0,3B	3B
8 *	09/89	3E/0F,3E	3E	K	25/A5	42/F0,42	42
9 (	0A/8A	46/0F,46	46	L	26/A6	4B/F0,4B	4B
0)	0B/8B	45/0F,45	45	; :	27/A7	4C/F0,4C	4C
- _	0C/8C	4E/0F,4E	4E	' "	28/A8	52/F0,52	52
= +	0D/8D	55/F0,55	55	` ~	29/A9	0E/F0,0E	0E
Backspace	0E/8E	66/F0,66	66	Shift (L)	2A/AA	12/F0,12	12/F0,12
Tab	0F/8F	0D/F0,0D	0D	\ |	2B/AB	5D/F0,5D	5C
Q	10/90	15/F0,15	15	Z	2C/AC	1A/F0,1A	1A
W	11/91	1D/F0,1D	1D	X	2D/AD	22/F0,22	22
E	12/92	24/F0,24	24	C	2E/AE	21/F0,21	21
R	13/93	2D/0F,2D	2D	V	2F/AF	2A/F0,2A	2A
T	14/94	2C/0F,2C	2C	B	30/B0	32/F0,32	32
Y	15/95	35/0F,35	35	N	31/B1	31/F0,31	31
U	16/96	3C/0F,3C	3C	M	32/B2	3A/F0,3A	3A
I	17/97	43/0F,43	43	, <	33/B3	41/F0,41	41
O	18/98	44/0F,44	44	. >	34/B4	49/F0,49	49
P	19/99	4D/0F,4D	4D	/ ?	35/B5	4A/F0,4A	4A
[{	1A/9A	54/0F,54	54	Shift (R)	36/B6	59/F0,59	59/F0,59
] }	1B/9B	5B/0F,5B	5B	Alt (L)	38/B8	11/F0,11	19/F0,19
Enter	1C/9C	5A/0F,5A	5A	(Space bar)	39/B9	29/F0,29	29
Ctrl (L)	1D/9D	14/0F,14	11/F0,11	Caps Lock	3A/BA	58/F0,58	14/F0,14

Таблица 1.5. Скан-коды функциональной и цифровой клавиатур

Клавиша	Set#l Наж./Отп.	Set#2 Наж./Отп.	Set#3 Наж./Отп.	Примечания
Esc	01/81	76/F0,76	08	-
F1	3B/BB	05/F0,05	07	-
F2	3C/BC	06/F0,06	0F	-
F3	3D/BD	04/F0,04	17	-
F4	3E/BE	0C/F0,0C	1F	-
F5	3F/BF	03/F0,03	27	-
F6	40/C0	0B/F0,0B	2F	-
F7	41/C1	83/F0,83	37	-
F8	42/C2	0A/F0,0A	3F	-
F9	43/C3	01/F0,01	47	-
F10	44/C4	09/F0,09	4F	-
F11	57/D7	78/F0,78	56	101/102
F12	58/D8	07/F0,07	5E	101/102
Scroll Lock	46/C6	7E/F0,7E	5F	-
[1.]SysRq	54/D4	-	-	AT-84[1.]
PrtSc	37/B7	-	-	XT, AT-84
* (Keypad)	37/B7	7C/F0,7C	7E	101/102
Num Lock	45/C5	77/F0,77	76	-
Home [7]	47/C7	6C/F0,6C	6C	-
↑ [8]	48/C8	75/F0,75	75	-
PgUp [9]	49/C9	7D/F0,7D	7D	-
- (Keypad)	4A/CA	7B/F0,7B	84	-
← [4]	4B/CB	6B/F0,6B	6B	-
[5]	4C/CC	73/F0,73	73	-
→ [6]	4D/CD	74/F0,74	74	-
+ (Keypad)	4E/CE	79/F0,79	7C	-
End [1]	4F/CF	69/F0,69	69	-
↓ [2]	50/D0	72/F0,72	72	-
PgDn [3]	51/D1	7A/F0,7A	7A	-
Ins [0]	52/D2	70/F0,70	70	-
Del [.]	53/D3	71/F0,71	71	-

[1.]Клавиша SysRq, есть только в AT-84, в 101/102 клавиатурах она соответствует комбинации Alt+Print Screen.

Остальные клавиши 101/102 клавиатур (управляющие) вызывают посылку цепочек скан-кодов, зависящую от состояния клавиш Shift, Alt, Ctrl.

Таблица 1.6. Цепочки скан-кодов управляющих клавиш (активный режим Num Lock)

Клавиша	Set#l Наж./Отп.	Set#2 Наж./Отп.	Set#3 Наж./Отп.
Alt (R)	E0,38/0E,88	E0,11/E0,F0,11	39
Ctrl (R)	E0,1D/0E,9D	E0,14/E0,F0,14	58
Print Screen	E0,2A,E0,37/E0,67,E0,AA	E0,12,E0,7C/E0,F0,7C,E0,F0,12	57
Shift+Print Screen	2A+E0,37/E0,B7+AA	-	-
Ctrl+Print Screen	1D+E0,37/E0,B7+9D	-	-
Alt+Print Screen (SysReq)	38+54/D4+B8	-	-
Pause	E1,1D,45,E1,9D,C5	E1,14,77,E1,F0,14,F0,77	62
Ctrl+Pause (Break)	1D+E0,46,E0,C6/+9D	-	-

Insert	E0,2A,E0,52/E0,D2,E0,AA	E0,70/E0,F0,70	67
Shift+Insert	2A+E0,52/E0,D2+AA	-	52
Delete	E0,2A,E0,53/E0,D3,E0,AA	E0,71/E0,F0,71	64
Shift+Delete	2A+E0,53/E0,D3+AA	-	-
←	E0,2A,E0,4B/E0,CB,E0,AA	E0,6B/E0,F0,6B	61
Shift+←	2A+E0,4B/E0,CB+AA	-	-
Home	E0,2A,E0,47/E0,C7,E0,AA	E0,6C/E0,F0,6C	6E
Shift+Home	2A+E0,47/E0,C7+AA	-	-
End	E0,2A,E0,4F/E0,CF,E0,AA	E0,69/E0,F0,69	65
Shift+End	2A+E0,4F/E0,CF+AA	-	-
↑	E0,2A,E0,48/E0,C8,E0,AA	E0,75/E0,F0,75	63
Shift+↑	2A+E0,48/E0,C8+AA	-	-
↓	E0,2A,E0,50/E0,D0,E0,AA	E0,72/E0,F0,72	60
Shift+↓	2A+E0,50/E0,D0+AA	-	-
Page Up	E0,2A,E0,49/E0,C9,E0,AA	E0,7D/E0,F0,7D	6F
Shift+Page Up	2A+E0,49/E0,C9+AA	-	-
Page Down	E0,2A,E0,51/E0,D1,E0,AA	E0,7A/E0,F0,7A	6D
Shift+Page Down	2A+E0,51/E0,D1+AA	-	-
→	E0,2A,E0,4D/E0,CD,E0,AA	E0,74/E0,F0,74	6A
Shift+→	2A+E0,4D/E0,CD+AA	-	-
Keypad Enter	E0,1C/E0,9C	E0,5A/E0,F0,5A	79
Keypad /	E0,35/E0,B5	E0,4A/E0,F0,4A	77
Shift+Keypad /	2A+E0,AA,E0,35/E0,B5,E0,2A+AA	-	-

Таблица 1.7. Цепочки скан-кодов управляющих клавиш (пассивный режим Num Lock)

Клавиша	Set#l Наж./Отп.	Set#2 Наж./Отп.	Set#3 Наж./Отп.
Alt (R)	E0,38/0E,88	E0,11/E0,F0,11	39
Ctrl (R)	E0,1D/0E,9D	E0,14/E0,F0,14	58
Print Screen	E0,2A,E0,37/E0,67,E0,AA	E0,12,E0,7C/E0,F0,7C,E0,F0,12	57
Shift+Print Screen	2A+E0,37/E0,B7+AA	-	-
Ctrl+Print Screen	1D+E0,37/E0,B7+9D	-	-
Alt+Print Screen (SysReq)	38+54/D4+B8	-	-
Pause	E1,1D,45,E1,9D,C5	E1,14,77,E1,F0,14,F0,77	62
Ctrl+Pause (Break)	1D+E0,46,E0,C6/+9D	-	-
Insert	E0,52/E0,D2	E0,70/E0,F0,70	67
Shift-Insert	2A+E0,AA,E0,52/E0,D2,E0,2A+AA	-	52
Delete	E0,53/E0,D3	E0,71/E0,F0,71	64
Shift-Delete	2A+E0,AA,E0,53/E0,D3,E0,2A+AA	-	-
←	E0,4B/E0,CB	E0,6B/E0,F0,6B	61
Shift+←	2A+E0,AA,E0,4B/E0,CB,E0,2A+AA	-	-
Home	E0,47/E0,C7	E0,6C/E0,F0,6C	6E
Shift+Home	2A+E0,AA,E0,47/E0,C7,E0,2A+AA	-	-
End	E0,4F/E0,CF	E0,69/E0,F0,69	65
Shift+End	2A+E0,AA,E0,4F/E0,CF,E0,2A+AA	-	-
↑	E0,48/E0,C8	E0,75/E0,F0,75	63
Shift+↑	2A+E0,AA,E0,48/E0,C8,E0,2A+AA	-	-
↓	E0,50/E0,D0	E0,72/E0,F0,72	60
Shift+↓	2A+E0,AA,E0,50/E0,D0,E0,2A+AA	-	-
Page Up	E0,49/E0,C9	E0,7D/E0,F0,7D	6F
Shift+Page Up	2A+E0,AA,E0,49/E0,C9,E0,2A+AA	-	-

Page Down	E0,51/E0,D1	E0,7A/E0,F0,7A	6D
Shift+Page Down	2A+E0,AA,E0,51/E0,D1,E0,2A+AA	-	-
→	E0,4D/E0,CD	E0,74/E0,F0,74	6A
Shift+→	2A+E0,AA,E0,4D/E0,CD,E0,2A+AA	-	-
Keypad Enter	E0,1C/E0,9C	E0,5A/E0,F0,5A	79
Keypad /	E0,35/E0,B5	E0,4A/E0,F0,4A	77
Shift+Keypad /	2A+E0,AA,E0,35/E0,B5,E0,2A+AA	-	-

Таблица 1.8. Цепочки скан-кодов дополнительных управляющих клавиш 107-клавишной клавиатуры

Клавиша	Set#l Наж./Отп.	Set#2 Наж./Отп.	Set#3 Наж./Отп.	Примечание
Power	E0,5E/E0,DE	E0,37/E0,F0,37	-	Поддержка ACPI
Sleep	E0,5F/E0,DF	E0,3F/E0,F0,3F	-	
Wake	E0,63/E0,E3	E0,5E/E0,F0,5E	-	
Окно (левое)	E0,5B/E0,DB	E0,1F/E0,F0,1F	8B/F0,8B	Microsoft Natural Kb.
Окно (правое)	E0,5C/E0,DC	E0,27/E0,F0,27	8C/F0,8C	
Меню	E0,5D/E0,DD	E0,2F/E0,F0,2F	8D/F0,8D	

Таблица 1.9. Цепочки скан-кодов дополнительных управляющих клавиш 122-клавишной клавиатуры

Клавиша	Set#l Наж./Отп.	Set#2 Наж./Отп.
Mute	E0,20/E0,A0	E0,23/E0,F0,23
Volume Down (-)	E0,2E/E0,AE	E0,21/E0,F0,21
Volume Up (+)	E0,30/E0,B0	E0,32/E0,F0,32
Media Select	E0,6D/E0,ED	E0,50/E0,F0,50
Previous Track (|◄◄)	E0,10/E0,90	E0,15/E0,F0,15
Next Track (►►|)	E0,19/E0,99	E0,4D/E0,F0,4D
Play/Pause (►/‖)	E0,22/E0,A2	E0,34/E0,F0,34
Stop (■)	E0,24/E0,A4	E0,3B/E0,F0,3B
E-mail	E0,6C/E0,EC	E0,48/E0,F0,48
WWW Stop	E0,68/E0,E8	E0,28/E0,F0,28
WWW Refresh	E0,67/E0,E7	E0,20/E0,F0,20
WWW Home	E0,32/E0,B2	E0,3A/E0,F0,3A
WWW Back	E0,6A/E0,EA	E0,38/E0,F0,38
WWW Forward	E0,69/E0,E9	E0,30/E0,F0,30
WWW Search	E0,65/E0,E5	E0,10/E0,F0,10
WWW Favorites	E0,66/E0,E6	E0,18/E0,F0,18
Sleep	E0,5F/E0,DF	E0,3F/E0,F0,3F
Calculator	E0,21/E0,A1	E0,2B/E0,F0,2B
My Computer	E0,6B/E0,EB	E0,40/E0,F0,40
Excel	E0,14/E0,94	E0,2C/E0,F0,2C
Word	E0,13/E0,93	E0,2D/E0,F0,2D
Ролик прокрутки - (наж./отп.) - прокрутка вниз - прокрутка вверх	 E0,23/E0,A3 E0,50/E0,D0 E0,48/E0,C8	 E0,33/E0,F0,33 E0,72/E0,F0,72 E0,75/E0,F0,75

1.2.6. Программирование клавиатуры

Программирование клавиатуры PC/XT заключается в обработке нажатия и отпускания клавиш клавиатуры, квитирование приема скан-кода по последовательному интерфейсу, а также в запрещение и разрешение интерфейса клавиатуры.

Клавиатура PC/AT и расширенные клавиатуры подключаются к системе через модифицированный адаптер клавиатуры и двунаправленный последовательный интерфейс, позволяющие передавать управляющую информацию в клавиатуру. Поэтому программирование этих клавиатур дополнительно

включает в себя процедуры управления адаптером клавиатуры и передачу в клавиатуру кодов команд и их атрибутов, а также принятие от клавиатуры различных кодов подтверждения, формируемых клавиатурой в ответ на команды.

1.2.6.1. Обработка нажатия и отпускания клавиш.

Как после нажатия, так и после отпускания клавиши адаптер клавиатуры генерирует аппаратное прерывание IRQ1. Обработчик прерывания INT 09h читает из порта 60h байт данных и начинает обработку нажатия (отпускания) клавиши. Результатом работы INT 09h могут являться

- установка флажков состояния клавиатуры (после отпускания клавиш Caps Lock, Num Lock и пр.);
- включение специальных режимов обработки (после нажатия Alt, Ctrl, Shft и пр.);
- вызов специальных программ - печать экрана, пауза и пр.;
- перевод скан кодов в двухбайтовые коды (ASCII-код и последний байт скан-кода) и занесение их в буфер клавиатуры, расположенный в оперативной памяти, отведенной для переменных BIOS (таблицы этих кодов приведены в разделе 1.2.6.4. 9).

1.2.6.1.1. Обработка специальных клавиш

Нажатие одной из *клавиш сдвига* (Ctrl, Shift, Alt) вызывает изменение статуса сдвигов, при этом изменяется трансляция других клавиш. Например, нажатие клавиши C при нажатой клавише Shift заставляет INT 09h интерпретировать ее как заглавную "C", а нажатие Ctrl + C - как прерывание текущей задачи. Отпускание клавиши сдвига возвращает клавиатуру в нормальное состояние.

Клавиши переключения - Caps Lock, Num Lock. Нажатие этих клавиш вызывает постоянное (до повторного нажатия) изменение статуса основной (Caps Lock) или цифровой (Num Lock) клавиатуры.

Комбинации клавиш:

- Ctrl-Alt-Del - перезагрузка ПК;
- Shift-PrtSc (или клавиша Print Screen на 101-клавишной клавиатуре) - вызов прерывания INT 05h - печать копии экрана;
- Ctrl-Num Lock (клавиша Pause на 101-клавишной клавиатуре) - пауза до первого нажатия клавиши;
- Ctrl-Break - генерирование прерывания INT 1Bh и установка разряда 7 в байте 0040:0071h;
- SysRq (или Alt-SysRq на 101-клавишной клавиатуре) - вызов подфункции AH=85h прерывания INT 15h.

1.2.6.2. Программное взаимодействие с клавиатурой

Программное взаимодействие с клавиатурой возможно через порты 60h, 61h и 64h, через обработчик аппаратного прерывания INT 09h; посредством функций программных прерываний INT 15h и INT 16h, обрабатываемых BIOS, функций программного прерывания INT 21h, обрабатываемого DOS и средствами стандартных прикладных программных интерфейсов, например Win32 API в среде современных ОС.

1.2.6.2.1. Работа с клавиатурой через порты ввода-вывода

Доступ к клавиатуре через порты ввода-вывода является самым низкоуровневым методом доступа. Такой метод используется обычно только прерываниями INT 09h, INT 15h и драйверами клавиатуры.

С помощью порта 61h можно определить, доступна ли клавиатура, и разрешить (запретить) доступ к ней (только для PC/XT, в PC/AT используются порты 60h и 64h и соответствующие команды адаптера клавиатуры). Причем, если значение, возвращаемое из порта 61h, содержит в седьмом разряде нуль, то клавиатура не доступна. Пример работы с портом 61h из обработчика прерывания INT 09h в PC/XT приведен ниже:

```
    in    al,  61h ; прочитать состояние PPI
    mov   ah,  al  ; сохранить состояние
    or    al,  80h ; запретить работу с клавиатурой
    out   61h, al
    xchg  ah,  al  ; восстановить статус
    out   61h, al ; установка и сброс D7 порта 61h формирует сигнал
                  ; подтверждения приема системой кода с клавиатуры
```

Аналогичный пример для PC/AT представлен ниже

```
    MOV   AL,AD   ; пересылка в адаптер кода команды "Блокировка
    OUT   [64],AL ; клавиатуры" (ADh)
A1: IN    AL,[64] ; проверка состояния входного буфера адаптера
    TEST  AL,02   ; клавиатуры
    JNZ   A1      ;
    MOV   AL,AE   ; пересылка в адаптер кода команды "Разблокировать
    OUT   [64],AL ; клавиатуру" (AEh).
```

В PC/AT как правило, блокировка интерфейса клавиатуры после приема очередного байта с клавиатуры удерживанием линии синхронизации на низком уровне осуществляется адаптером клавиатуры автоматически на время, пока система не прочитает поступивший байт из выходного буфера адаптера через порт 60h.

В PC/XT порт 60h предназначен для чтения данных с клавиатуры. Пример чтения скан-кода с помощью порта 60h и 61h в обработчике прерывания клавиатуры PC/XT приведен ниже.

```
in    al,60h     ; прочитать скан-код
mov   bl,al      ; в bl - скан-код

in    al,61h     ; см. предыдущий пример
mov   ah,al
or    al,80h
out   61h,al
xchg  ah,al
out   61h,al

mov   al,20H     ; послать сигнал конца прерывания
out   20H,al     ; в контроллер прерываний
```

В PC/AT при обмене информацией с клавиатурой порт 60h используется для чтения информации, поступающей с клавиатуры (скан-коды и коды ответа клавиатуры) и передачи команд и их атрибутов клавиатуре, а порт 64h - для определения статуса клавиатуры и запрещения или разрешение ее работы. Такое разделение функций обеспечивает максимальную гибкость программам. Ниже приводится пример чтения скан-кода с помощью портов 60h и 64h в PC/AT:

```
2C47:0100  FA       CLI           ;запрет обработки маскируемых аппаратных
                                  ;прерываний
2C47:0101  E464     IN    AL,[64] ;проверка наличия данных в выходном буфере
2C47:0103  A801     TEST  AL,01   ;адаптера клавиатуры
2C47:0105  74FA     JZ    0101    ;
2C47:0107  E460     IN    AL,[60] ;чтение скан-кода из выходного буфера и
2C47:0109  88C3     MOV   BL,AL   ;пересылка его в регистр BLh
2C47:010B  E464     IN    AL,[64] ;отработка сигнала ACKk подтверждения приема
2C47:010D  A802     TEST  AL,02   ;скан-кода системой (см. предыдущий пример)
2C47:010F  75FA     JNZ   010B    ;Как правило эту операцию
2C47:0111  B0AD     MOV   AL,AD   ;адаптер клавиатуры
2C47:0113  E664     OUT   [64],AL ;выполняет автоматически
2C47:0115  E464     IN    AL,[64] ;своими программными
2C47:0117  A802     TEST  AL,02   ;средствами, удерживая линию синхронизации
2C47:0119  75FA     JNZ   0115    ;на низком уровне пока процессор не прочтет
2C47:011B  B0AE     MOV   AL,AE   ;содержимое порта 60
2C47:011D  E664     OUT   [64],AL ;
2C47:011F  FB       STI           ;разрешение обработки маскируемых аппаратных
                                  ;прерываний
2C47:0120  CC       INT3          ;возвращение управления отладчику
```

В данном фрагменте программы команда CLI блокирует работу обработчика прерываний клавиатуры INT 09h на время выполнения программы. Команда STI разрешает работу обработчика INT 09h. При запуске этого фрагмента программы в среде отладчика AFD в результате ее работы в регистр BL занесется код 9Ch - скан код отпускания клавиши "Enter", нажатием которой инициируется команда G запуска программы в среде AFD.

Как правило, блокировка интерфейса клавиатуры после приема очередного байта с клавиатуры удерживанием линии синхронизации на низком уровне осуществляется адаптером автоматически на время, пока система не прочитает поступивший байт из выходного буфера адаптера через порт 60h. Поэтому в выше приведенной программе можно исключить команды, реализующие формирование сигнала подтверждения приема скан-кода системой (команды с адреса 2C47:010B по 2C47:011D). Управление активностью интерфейса клавиатуры командами адаптера клавиатуры ADh и AEh может применяться в программах, которые должны блокировать клавиатуру на время выполнения определенного набора операций и т. п.

В PC/AT запись в порт 60h позволяет программировать клавиатуру, т.е. все, что помещается в порт 60h передается в клавиатуру (исключение составляют атрибуты команд адаптера клавиатуры, которые также записываются по адресу 60h). С помощью порта 64h можно читать состояние линий интерфейса клавиатуры, состояние клавиатуры и адаптера клавиатуры, состояние портов его микроконтроллера, осуществлять программирование адаптера клавиатуры (используются команды взаимодействия с адаптером клавиатуры, см. раздел 1.1).

При программирование клавиатуры необходимо первоначально в порт 60h записать код команды, сделать задержку и затем записать байт данных в тот же порт 60h (список команд приведен ниже):

```
mov   al, код команды ; см. ниже
          out   60h, al          ; запись кода команды в порт
                                 ; если нужно посылать данные,
          mov   cx, 2000h        ; то необходима задержка
 pause:   loop  pause            ; задержка
          mov   al, байт данных ; запись параметра команды
          out   60h, al
```

Приведенный фрагмент программы может работать только на ПК с тактовыми частотами процессора меньше 30 мГц. На современных ПК вместо программной задержки нужно контролировать поступление от клавиатуры кода подтверждения приема (код FAh). Этот код доступен по чтению через порт 60h. Соответствующий фрагмент программы представлен ниже.

```
2C47:0100  FA      CLI             ;запрет маскируемых прерываний
2C47:0101  E464    IN     AL,[64]  ;определение состояния входного
2C47:0103  A802    TEST   AL,02    ;буфера: 0 - свободен, 1 - занят
2C47:0105  75FA    JNZ    0101     ;
2C47:0107  B0ED    MOV    AL,ED    ;пересылка в клавиатуру кода команды
2C47:0109  E660    OUT    [60],AL  ;"Установить индикаторы состояния" (код EDh)
2C47:010B  E464    IN     AL,[64]  ;определение состояния выходного буфера
2C47:010D  A801    TEST   AL,01    ;адаптера: 0 - пуст, 1 - принят код с
2C47:010F  74FA    JZ     010B     ;клавиатуры
2C47:0111  E460    IN     AL,[60]  ;чтение кода из выходного буфера и сравнение
2C47:0113  3CFA    CMP    AL,FA    ;его с кодом FAh - подтверждение АСК
2C47:0115  7512    JNZ    0129     ;если не FAh - выход по ошибке
2C47:0117  B007    MOV    AL,07    ;код конфигурации - включить все индикаторы
2C47:0119  E660    OUT    [60],AL  ;пересылается в клавиатуру (код 07h)
2C47:011B  E464    IN     AL,[64]  ;определение состояния выходного буфера
2C47:011D  A801    TEST   AL,01    ;адаптера: 0 - пуст, 1 - принят код с
2C47:011F  74FA    JZ     011B     ;клавиатуры
2C47:0121  E460    IN     AL,[60]  ;чтение кода из выходного буфера и сравнение
2C47:0123  3CFA    CMP    AL,FA    ;его с кодом FAh - подтверждение АСК
2C47:0125  7502    JNZ    0129     ;если не FAh - выход по ошибке
2C47:0127  FB      STI             ;разрешение маскируемых прерываний
2C47:0128  CC      INT3            ;передача управления отладчику
2C47:0129  88C4    MOV    AH,AL    ;пересылка ошибочного кода в регистр AH
2C47:012B  EBFA    JMP    0127     ;переход на окончание программы
```

При выполнение этого фрагмента программы в среде отладчика на клавиатуре включаются все три индикатора состояния: Num Lock, Caps Lock и Scroll Lock.

1.2.6.2.2. Команды клавиатуры и их описание.

Команды передаются клавиатуре через входной буфер адаптера клавиатуры, записью их кода по адресу 60h. Ответные коды клавиатуры на команды помещаются микроконтроллером адаптера в выходной буфер и доступны для чтения по адресу 60h. При программирование клавиатуры на уровне ее команд необходимо соблюдать ряд правил:

- передавать код команды через входной буфер адаптера можно только при наличие признака "Входной буфер пуст" (разряд D1 регистра состояния адаптера клавиатуры установлен в 0);
- ответные коды клавиатуры на команды можно читать из выходного буфера только после установки адаптером признака "Выходной буфер полон" (разряд D0 регистра состояния адаптера клавиатуры установлен в 1);
- перед программированием клавиатуры необходимо запретить, при завершение - разрешить обработку прерывания клавиатуры.

Список команд клавиатуры приведен в табл. 1.10, а ниже - их описание.

Табл. 1.10. Команды клавиатуры

Код	Функция
EDh	Установка и сброс индикаторов состояния
EEh	Эхо (средство диагностирования)
EFh, F1h	Резерв холостые команды (NOP)
F0h	Выбрать альтернативный режим
F2h	Идентификатор расширенной клавиатуры
F3h	Установить частоту повторения клавиши
F4h	Разрешить клавиатуре передачу данных
F5h	Запрет по умолчанию

F6h	Установить условия по умолчанию
F7h - FAh	Установить все клавиши
FBh - FDh	Установить отдельные клавиши
FEh	Послать повторно
FFh	Сброс

Установить индикаторы состояния (EDh)

"Установить индикаторы состояния" (EDh) - это двухбайтовая команда, которая изменяет состояние светодиодных индикаторов клавиатуры. После получения этой команды клавиатура прекращает сканирование, возвращает системе код ACK (FAh - код подтверждения) и ожидает от системы байта конфигурации. Байт конфигурации определяет установку светодиодных индикаторов и имеет следующий формат:

7	6	5	4	3	2	1	0
Резерв					CL	NL	SL

Бит CL задаст значение индикатора CAPS LOCK (0 - выключен, 1 - включен).

Бит NL управляет значением индикатора NUM LOCK (0 -выключен, 1 - включен).

Бит SL задает значение индикатора SCROLL LOCK (0 - выключен, 1 - включен).

Зарезервированные биты 7-3 должны быть установлены в 0.

После получения байта конфигурации клавиатура устанавливает индикатор состояния, возвращает код ACK и возобновляет сканирование. Если в то время, когда клавиатура ожидает байт конфигурации, поступает другая команда, выполнение команды "Установить индикаторы состояния" прекращается и выполняется новая команда.

Эхо (EEh)

"Эхо" (EEh) используется для диагностирования. После получения этой команды клавиатура возвращает ответ - "Эхо" (EEh). Обычно команда EEh используется при возникновение ошибок в работе клавиатуры. Посылка команды EEh после выполнения сброса клавиатуры позволяет определить, восстановился ли нормальный режим функционирования. Неправильный ответ на эхо-команду означает, что контроллер клавиатуры "зациклился" и привести систему в нормальное состояние можно только отключением питания.

Выбрать альтернативный режим (F0h)

"Выбрать альтернативный режим" (F0h): клавиатура подтверждает эту двухбайтовую команду, сбрасывает выходной буфер и повторение клавиши. После этого она готовится к приему байта конфигурации, который задает рабочий режим. Клавиатура подтверждает байт конфигурации, устанавливает новый рабочий режим и возобновляет сканирование.

Байт конфигурации должен содержать номер устанавливаемого режима клавиатуры (01h - режим 1; 02h - режим 2 и 03h - режим 3).

Когда система посылает байт конфигурации 00h, клавиатура отвечает кодом подтверждения и посылает байт конфигурации, указывающий системе текущий режим.

Холостые операции (EFh, F1h)

"Холостые операции" (EFh, F1h): в ответ на такую команду клавиатура посылает код повторной посылки (FEh), не предпринимая никаких других действий.

Идентификатор расширенной клавиатуры (F2h)

"Идентификатор расширенной клавиатуры" (F2h) - это команда, передаваемая клавиатуре при включении питания. Клавиатура отвечает кодом ACK, прекращает сканирование, передает два байта идентификатора расширенной клавиатуры: сначала код ABh, а затем код 83h (или другой, например 41h) и возобновляет сканирование клавиш.

Установить частоту повторения клавиши (F3h)

"Установить частоту повторения клавиши" (F3h) - это двухбайтовая команда, которая изменяет значения задержки и частоты повторения клавиши. Схема работы этой команды аналогична схеме работы команды "Установить индикаторы состояния" (EDh).

Как только клавиатура получает команду "Установить частоту повторения клавиши", она прекращает сканирование и возвращает код ACK. После этого система посылает клавиатуре байт, в котором содержатся значения частоты повторения и времени задержки. Частота повторения показывает, сколько раз в секунду повторяется данная клавиша. Время задержки - это время, в течение которого клавиша должна удерживаться нажатой, прежде чем начнется функция повторения. Клавиатура возвращает код ACK и остается в предшествующем состоянии сканирования - разрешенном или запрещенном.

Для возобновления сканирования должна быть передана команда "Разрешить клавиатуру" (F4h).

Байт частоты повторения и времени задержки имеет следующий формат:

- частота повторения (число повторений в секунду) - разряды (4 - 0):

00000=30,0,	01000=15,0,	10000=7,5,	11000=3,7,
00001=26,6,	01001=13,3,	10001=6,7,	11001=3,3,
00010=24,0,	01010=12,0,	10010=6,0,	11010=3,0,
00011=21,8,	01011=10,9,	10011=5,5,	11011=2,7,
00100=20,0,	01100=10,0,	10100=5,0,	11100=2,5,
00101=18,4,	01101=9,2,	10101=4,6,	11101=2,3,
00110=17,1,	01110=8,6,	10110=4,3,	11110=2,1,
00111=16,0,	01111=8,0,	10111=4,0,	11111=2,0;

- время задержки (в миллисекундах) - разряды (6 - 5):

00=250,	01=500,	10=750,	11=1000.

Седьмой разряд всегда должен содержать нуль.

Разрешить клавиатуру (F4h)

Команда "Разрешить клавиатуру" (F4h) разрешает клавиатуре начать сканирование и передачу данных. После получения этой команды клавиатура возвращает код ACK, сбрасывает выходной буфер и начинает сканирование.

Запрет по умолчанию (F5h)

"Запрет по умолчанию" (F5h) останавливает сканирование клавиатуры. После получения этой команды клавиатура возвращает системе код ACK (подтверждение) и:

- устанавливает частоту повторения клавиши и время задержки, принятые по умолчанию;
- прекращает сканирование;
- очищает буфер клавиатуры;
- устанавливает типы клавиш по умолчанию (только для режима 3);
- сбрасывает последнюю повторяемую клавишу;
- ожидает дальнейших команд.

Установить условия по умолчанию (F6h)

По команде "Установить условия по умолчанию" (F6h) клавиатура посылает код ACK и:

- устанавливает частоту повторения клавиши и время задержки, принятые по умолчанию;
- очищает буфер клавиатуры;
- устанавливает типы клавиш по умолчанию (только для режима 3);
- сбрасывает последнюю повторяемую клавишу;
- продолжает сканирование.

Установить все клавиши (F7h, F8h, F9h, FAh)

Таблица 1.11. Назначение команд установки клавиш

Команда	Действие всех клавиш
F7h	Только Повторение
F8h	Только Нажатие/Отпускание
F9h	Только Нажатие
FAh	Повторение/Нажатие/Отпускание

По команде "Установить все клавиши" (F7h, F8h, F9h, FAh) клавиатура посылает подтверждение получения команды, очищает выходной буфер, устанавливает все клавиши так, как это указано командой (табл. 1.11), и продолжает сканирование, если оно до этого было разрешено. Эти команды используются только в режиме 3.

Установить отдельные клавиши (FBh, FCh, FDh)

По команде "Установить отдельные клавиши" (FBh, FCh, FDh) клавиатура подтверждает получение этой двухбайтовой команды, устанавливает команду кода отдельной клавиши, очищает выходной буфер, прекращает сканирование и ожидает байт идентификации клавиши. Идентификация клавиши выполняется с помощью кода сканирования позиции клавиши, приведенного в табл. 1.12. Клавиатура подтверждает байт идентификации клавиши, устанавливает новый тип клавиши и ожидает следующей команды. Клавиатура не возобновляет сканирование, пока система не позволит сделать это посредством команды "Разрешить клавиатуру" (F4h), что дает возможность одновременной установки нескольких клавиш. Эти команды используются только в режиме 3.

Таблица 1.12. Назначение команд установки отдельных клавиш

Команда	Действие всех клавиш
FBh	Только Повторение
FCh	Только Нажатие/Отпускание
FDh	Только Нажатие

Послать повторно (FEh)

Команда "Послать повторно" (FEh) выдается только после передачи данных с клавиатуры. Клавиатура отвечает повторной передачей последнего байта, переданного системе. Если последний переданный байт был RESEND (FEh), клавиатура передает предшествующий ему байт.

Сброс (FFh)

По команде "Сброс" (FFh) клавиатура возвращает системе код ACK. Для того чтобы принять этот код, система возбуждает линии CLOCK и DATA по меньшей мере на 500 мкс. После передачи кода ACK клавиатура выполняет внутренний тест (BAT) и посылает в систему код завершения теста - значение AAh, если тест завершился удачно, и FDh, если произошла ошибка.

1.2.6.2.3. Коды ответов клавиатуры

Клавиатура отвечает системе кодами, описание которых приведено ниже.

Переполнение/ошибка клавиши (00h или FFh). Для указания переполнения буфера клавиатура помещает 00h в буфер в качестве последнего символа для кодов сканирования в режимах 2 и 3. Для кодов сканирования в режиме 1 используется FFh.

Клавиатура передает код ошибки клавиши, если не удается идентифицировать замыкание контактов на клавиатуре. Для кодов сканирования в режиме 1 передается FFh, а в режимах 2 и 3 - 00h.

Подтверждение (FAh). Клавиатура выдает подтверждение (ACK) в ответ на любую правильную команду, кроме команд "Эхо" и RESEND. Если клавиатура прерывается во время передачи ACK, то ACK отменяется и клавиатура начинает обработку новой команды.

Код завершения включения питания (AAh). После успешного завершения включения питания клавиатура передает системе код AAh. Передача любого другого кода после включения указывает на неисправность клавиатуры.

Эхо (EEh). Клавиатура передает этот код в ответ на команду "Эхо".

Идентификатор клавиатуры (83ABh). Идентификатор клавиатуры состоит из двух байтов - 83ABh. Клавиатура отвечает на команду "Идентификатор расширенной клавиатуры" подтверждением, прерывает сканирование и передает два байта идентификатора. Первым передается младший байт. После вывода идентификатора клавиатура начинает сканирование.

Повторная посылка (FEh). Клавиатура передает этот код после получения неправильных входных данных или данных с ошибкой паритета.

1.2.6.3. Работа с клавиатурой через прерывание INT 09h

Прерывание INT 09h является аппаратным прерыванием (IRQ 1). Обработчик этого прерывания обрабатывает нажатия и отпускания клавиш клавиатуры на уровне BIOS. Принятое с клавиатуры значение скан-кода и соответствующее ему значение ASCII кода обработчик прерывания INT 09 сохраняет в буфере клавиатуры. Буфер расположен в памяти рабочей области BIOS (в области переменных BIOS) по адресам с 0040:001A по 0040:003C. Обработчик также устанавливает флажки состояния статуса клавиш-переключателей (Shift, Ctrl, Caps Lock и т. д.) в ячейках 0040:0017 и 0040:0018. Ячейка 0:0419 используется как аккумулятор кода Alt-набора.

Буфер клавиатуры может накапливать до 15 нажатий на клавишу независимо от того, являются они однобайтовыми кодами ASCII или двухбайтовыми расширенными кодами. Таким образом в буфере отводится два байта памяти для каждого нажатия на клавишу. Для однобайтовых кодов первый байт содержит код ASCII, а второй - скан-код клавиши. Для расширенных кодов первый байт содержит код ASCII 0, а второй - номер расширенного кода. Этот код обычно совпадает со скан-кодом клавиши, но не всегда, поскольку некоторые клавиши можно комбинировать с клавишами сдвига для генерации различных кодов.

Буфер устроен, как циклическая очередь, которую называют также буфером FIFO (первый вошел - первый ушел). Он занимает непрерывную область адресов памяти. Однако определенной ячейки памяти, которая хранит "начало строки" в буфере, нет. Вместо этого два указателя хранят позиции головы (0040:001A) и хвоста (0040:001C) строки символов, находящейся в буфере в текущий момент. Новые нажатия клавиш сохраняются в позициях, следующих за хвостом (в более старших адресах памяти), и соответственно обновляется указатель хвоста буфера. После того как израсходовано все буферное пространство, новые символы продолжают вставляться, начиная с самого начала буферной области (0040:001E); поэтому возможны ситуации, когда голова строки в буфере имеет больший адрес, чем хвост. Когда буфер заполнен, новые вводимые символы игнорируются, при этом обработчик прерывания клавиатуры выдаст гудок через динамик.

В то время как указатель на голову строки установлен на первый введенный символ, указатель на хвост установлен на позицию за последним введенным символом. Когда оба указателя равны, буфер пуст. Чтобы разрешить ввод 15 символов, требуется 16-я пустая позиция, 2 байта которой всегда содержат код возврата каретки (ASCII 13 (0Dh)) и скан-код клавиши «Enter», равный 28 (1Ch). Эта пустая позиция непосредственно предшествует голове строки символов. 32 байта буфера начинаются с адреса 0040:001E. Указатели на голову

и хвост расположены по адресам 0040:001A и 0040:001C соответственно. Хотя под указатели отведено 2 байта, используется только младший байт. Значения указателей меняются от 30 (1Eh) до 60 (3Ch), что соответствует позициям буфера клавиатуры в области данных BIOS. Для очистки буфера надо просто установить значение ячейки 0040:001A равным значению ячейки 0040:001C.

Отметим, что программа может вставлять символы в буфер, завершая строку символом возврата каретки и меняя значения указателей. Если это сделано перед завершением программы, то при возврате управления в MS-DOS эти символы будут считаны и может быть автоматически загружена другая программа.

В связи с такой организацией буфера не рекомендуется вызывать прерывание INT 09h непосредственно из пользовательских программ, поскольку это может привести к непредсказуемым последствиям.

Назначение флажковых разрядов, определяющих статус клавиш-переключателей и устанавливаемых обработчиком INT 09h в ячейках памяти 0040:0017 и 0040:0018, приведено в таблице 1.13.

Перехват прерывания INT 09h можно использовать в резидентных программах для работы с кодами национальных шрифтов или в программах, построенных на интенсивной обработке данных с клавиатуры (компьютерные игры). Как правило, обработка клавиш в INT 09h выполняется с помощью портов 60h/61h/64h, (порт 61h используется в PC/XT) как рассмотрено выше.

Стандартный INT 09h вызывает следующие прерывания:

- INT 05h после нажатия клавиши Prt Sc для печати копии экрана;
- INT 15h (подфункция 4Fh) для альтернативной обработки нажатия (отпускания) клавиши. Эта подфункция вызывается (для 83- и 101/102-клавишных клавиатур) перед обработкой прочитанного кода клавиши. При вызове INT 15h/4Fh в AL передается скан-код клавиши и устанавливается CF; если клавиша обработана альтернативным обработчиком, то после завершения подфункции должен быть сброшен CF (иначе значения регистров не должны измениться);
- INT 15h (подфункция 85h) для обработки клавиши Sys Req с входными параметрами: AH = 85h (номер подфункции), AL = 00h (если Sys Req нажата) или 01h (если Sys Req отжата). Обычно эта подфункция никак не задействована в BIOS и используется прикладными программами;
- INT 15h (подфункция 91h - "прерывание завершено") со значением AL=02h (клавиатура).
- и ряд других.

Таблица 1.13. Назначение флажковых разрядов

Адрес	Бит	Клавиша	Значение, когда бит = 1
0040:0017	0	Правый Shift	Клавиша нажата
	1	Левый Shift	Клавиша нажата
	2	Любой Ctrl	Клавиша нажата
	3	Любой Alt	Клавиша нажата
	4	Scroll Lock	Режим Scroll Lock включен
	5	Num Lock	Режим Num Lock включен
	6	Caps Lock	Режим Caps Lock включен
	7	Insert	Режим вставки включен
0040:0018	0	Левый Ctrl	Клавиша нажата
	1	Левый Alt	Клавиша нажата
	2	Правый Ctrl	Клавиша нажата
	3	Правый Alt	Клавиша нажата
	4	Scroll Lock	Клавиша нажата
	5	Num Lock	Клавиша нажата
	6	Caps Lock	Клавиша нажата
	7	SysReq	Клавиша нажата

1.2.6.4. Работа с клавиатурой через прерывание INT 16h (драйвер клавиатуры)

Программно вызываемое прерывание BIOS INT 16h представляет интерфейс прикладного уровня для клавиатуры. Его основное назначение — извлечение слов из клавиатурного буфера. Функция задается в регистре AH при вызове:

- AH = 00h — чтение (с ожиданием готовности) и выборка слова из буфера;
- AH = 01h — проверка готовности, чтение без выборки;
- AH = 02h — чтение состояния флагов;
- AH = 03h — установка задержки и скорости автоповтора;
- AH = 05h — запись слова в буфер клавиатуры.

Функции *с* AH = 10h, 11h, 12h аналогичны 00h, 0lh, 02h, но предназначены специально для 101/102-клавишных клавиатур. Для ряда клавиш, отсутствующих в клавиатуре AT-84, эти функции дадут результаты, отличающиеся от результатов вызовов 00h, 0lh и 02h.

Функции с AH = 20h, 21h и 22h используются для дополнительных клавиш 122-клавишной клавиатуры.

Ниже приводится описание функций 00 - 05 и 10 - 12.

1.2.6.4.1. Функции прерывания INT 16h

Чтение данных с клавиатуры с ожиданием нажатия

Реально эта функция берет данные из буфера клавиатуры, поэтому с ее помощью нельзя определить нажатие таких клавиш, как Alt, Shift и т.п.

Вход: AH = 00h.
Выход: AL - ASCII-код клавиши,
AH - скан-код клавиши.

Проверка клавиатуры

Эта подфункция проверяет, есть ли в буфере символ (код клавиши). При наличии символа возвращает его значение, но не очищает буфер клавиатуры.

Вход: AH = 01h.
Выход: ZF=ZR=1, если в буфере нет символа;
ZF=ZR=0, если в буфере есть символ, при этом
AL - ASCII-код клавиши,
AH - скан-код клавиши.

Прочитать статус сдвигов

Возвращает значение байта 0040:0017h.

Вход: AH = 02h.
Выход: AL - статус сдвигов:
разряд 0 - 1, если правая клавиша Shift нажата,
разряд 1 - 1, если левая клавиша Shift нажата,
разряд 2 - 1, если клавиша Ctrl (с любой стороны) нажата,
разряд 3 - 1, если клавиша Alt (с любой стороны) нажата,
разряд 4 - 1, если активен режим Scroll Lock,
разряд 5 - 1, если активен режим Num Lock,
разряд 6 - 1, если активен режим Caps Lock,
разряд 7 - 1, если активен режим Insert.

Установка скорости генерации символов повтора и паузы

Не используется с клавиатурой PC XT.

Вход: AH = 03h,
AL = 05h,
BL - скорость генерации символов (0 - быстрее, чем 1Fh, точнее см. в разделе 1.2.6.2.2., команда F3h),
BH - задержка (0 - 0,25 с, 1 - 0,5 с, 2 - 0,75 с, 3 - 1 с).
Выход: нет.

Записать символ в буфер клавиатуры

Не используется с клавиатурой PC/XT.

Вход: AH = 05h,
CL - ASCII-код символа,
CH - скан код или нуль, если скан код безразличен.
Выход: AL = 0, если операция записи завершилась успешно.

Чтение данных с клавиатуры с ожиданием нажатия (для 101-клавишной клавиатуры)

Эта подфункция является как бы расширением подфункции 00h.

Вход: AH = 10h.
Выход: AL - ASCII-код клавиши,
AH - скан-код клавиши.

Проверка клавиатуры (для 101-клавишной клавиатуры)

Эта подфункция является расширением подфункции 01h.

Вход: AH = 11h.
Выход: ZF=ZR=1, если в буфере нет символа;
ZF=ZR=0, если в буфере есть символ, при этом
AL - ASCII-код клавиши,

AH - скан-код клавиши.

Прочитать статус сдвигов 101-клавишной клавиатуры

Вход: AH = 12h.

Выход: в AL - статусы, совпадающие с результатами работы функции 02h

AH - статус сдвигов дополнительных клавиш 101-клавишной клавиатуры:

разряд 0 - 1, если левая клавиша "Ctrl" нажата,
разряд 1 - 1, если левая клавиша "Alt" нажата,
разряд 2 - 1, если правая клавиша "Ctrl" нажата,
разряд 3 - 1, если правая клавиша "Alt" нажата,
разряд 4 - 1, если клавиша "Scroll Lock" нажата,
разряд 5 - 1, если клавиша "Num Lock" нажата,
разряд 6 - 1, если клавиша "Caps Lock" нажата,
разряд 7 - 1, если клавиша "Sys Req" нажата.

Младший байт считанного функциями 00h, 10h и 20h слова содержит *ASCII-символ,* старший — *скан-код.* Символы, полученные нестандартным способом (в русском регистре или Alt-набором), сопровождаются нулевым скан-кодом. При нулевом младшем байте старший содержит *расширенный ASCII-код* (Extended ASCII Keystroke). Дополнительные клавиши 101/102 клавиатур при использование функций 10h-11h генерируют код E0h в младшем байте и скан-код, соответствующий аналогичным управляющим клавишам 83/84-клавишных клавиатур.

Функция записи (05h), несколько неожиданная для клавиатуры, позволяет легко имитировать работу оператора для различных демонстрационных программ. Если прикладная программа не перехватывает обслуживание клавиатуры на уровне аппаратного прерывания (INT 9h), то резидентная программа может ей «подбрасывать» слова в буфер, которые будут восприниматься как нажатие клавиш.

1.2.6.4.2. ASCII-коды

ASCII-коды буфера клавиатуры, соответствующие нажатию клавиш, приведены в табл. 1.14 и 1.15. В скобках приводятся значения скан-кода и ASCII-кода клавиши, возвращаемые в регистрах AH и AL функциями 00h и 10h прерывания INT 16h соответственно. При русификации (или другой локализации) клавиатуры отслеживание переключения регистров (языков) ложится на соответствующий обработчик аппаратного прерывания клавиатуры.

Таблица 1.14. ASCII-коды основной клавиатуры

Клавиша		Регистр				
		Латинский		Русский		Л/Р[2]
Лат.	Рус.	Norm.[1]	Shift[1]	Norm. [1]	Shift[1]	Ctrl
Esc		1B (01,1B/01,1B)	1B (01,1B/01,1B)	1B (01,1B/01,1B)	1B (01,1B/01,1B)	1B (01,1B/01,1B)
` ~	ё Ё (( ))	60 (29,60/29,60)	7E (29,7E/29,7E)	F1 (29,F1/29,F1)	F0 (00,F0/00,F0)	-
1 !	1 !	31 (02,31/02,31	21 (02,21/02,21)	31 (02,31/02,31)	21(02,21/02,21)	-
2 @	2 "	32 (03,32/03,32)	40 (03,40/03,40)	32 (03,32/03,32)	22 (03,22/03,22)	03,00/03,00
3 #	3 № (/)	33 (04,33/04,33)	23 (04,23/04,23)	33 (04,33/04,33)	FC (04,FC/04,FC)	-
4 $	4 ; (☼)	34 (05,34/05,34)	24 (05,24/05,24)	34 (05,34/05/34)	3B (05,3B/05,3B)	-
5 %	5 % (:)	35 (06,35/06,35)	25 (06,25/06,25)	35 (06,35/06,35)	25 (06,25/06,25)	-
6 ^	6 : (,)	36 (07,36/07,36)	5E (07,5E/07,5E)	36 (07,36/07,36)	3A (07,3A/07,3A)	1E (07,1E/07,1E)
7 &	7 ? (.)	37 (08,37/08,37)	26 (08,26/08,26)	37 (08,37/08,37)	3F (08,3F/08,3F)	-
8 *	8 * (;)	38 (09,38/09,38)	2A (09,2A/09,2A)	38 (09,38/09,38)	3B (09,2A/09,2A)	-
9 (	9 ((?)	39 (0A,39/0A,39)	28 (0A,28/0A,28)	39 (0A,39/0A,39)	28 (0A,38/0A,28)	-
0)	0) (%)	30 (0B,30/0B,30)	29 (0B,29/0B,29)	30 (0B,30/0B,30)	29 (0B,29/0B,29)	-
- _		2D	5F (0C,5F/0C,5F)	2D	5F (0C,5F/0C,5F)	1F (0C,1F/0C,1F)
= +		3D	2B	3D	2B	-
Backspace		08 (0E,08/0E,08)	08 (0E,08/0E,08)	08 (0E,08/0E,08)	08 (0E,08/0E,08)	7F (0E,7F/0E,7F)
Tab		09 (0F,09/0F,09)	EXT(0F,00/0F,00	09 (0F,09/0F,09)	EXT(0F,00/0F,00	EXT(-/94,00)
Q q	Й й	71 (10,71/10,71)	51 (10,51/10,51)	A9 (10,A9/10,A9)	89 (10,89/10,89)	11 (10,11/10,11)
W w	Ц ц	77 (11,77/11,77)	57 (11,57/11,57)	E6 (11,E6/11,E6)	96(11,96/11,96)	17 (11,17/11,17)
E e	У у	65 (12,65/12,65)	45 (12,45/12,45)	E3 (12,E3/12,E3)	93(12,93/12,93)	05 (12,05/12,05)
R r	К к	72 (13,72/13,72)	52 (13,52/13,52)	AA	8A(13,8A/13,8A)	12 (13,12/13,12)
T t	Е е	74 (14,74/14,74)	54 (14,54/14,54)	A5 (14,A5/14,A5)	85 (14,85/14,85)	14 (14,14/14,14)

Y y	Н н	79 (15,79/15,79)	59 (15,59/15,59)	AD	8D(15,8D/15,8D)	19 (15,19/15,19)
U u	Г г	75 (16,75/16,75)	55 (16,55/16,55)	A3 (16,A3/16,A3)	83 (16,83/16,83)	15 (15,15/16,15)
I i	Ш ш	69 (17,69/17,69)	49 (17,49/17,49)	E8 (17,E8/17,E8)	98 (17,98/17,98)	09 (17,09/17,09)
O o	Щ щ	6F (18,6F/18,6F)	4F (18,4F/18,4F)	E9 (18,E9/18,E9)	99 (18,99/18,99)	0F (18,0F/18,0F)
P p	З з	70 (19,70/19,70)	50 (19,50/19,50)	A7 (19,A7/19,A7)	87 (19,87/19,87)	10 (19,10/19,10)
[{	Х х	5B (1A,5B/1A,5B)	7B (1A,7B/1A,7B)	E5 (1A,E5/1A.E5)	95 (1A,95/1A,95)	1B (1A,1B/1A,1B)
] }	Ъ ъ	5D (1B,5D/1B,5D)	7D (1B,7D/1B,7D)	EA (1B,EA/1B,EA)	9A (1B,9A/1B,9A)	1D (1B,1D/1B,1D)
Enter		0D (1C,0D/1C,0D)	0D (1C,0D/1C,0D)	0D (1C,0D/1C,0D)	0D (1C,0D/1C,0D)	0A (1C,0A/1C,0A)
A a	Ф ф	61 (1E,61/1E,61)	41 (1E,41/1E,41)	E4 (1E,E4/1E,E4)	94 (1E,94/1E,94)	01 (1E,01/1E,01)
S s	Ы ы	73 (1F,73/1F,73)	53 (1F,53/1F,53)	EB (1F,EB/1F,EB)	9B (1F,9B/1F9B)	13 (1F,13/1F,13)
D d	В в	64 (20,64/20,64)	44 (20,44/20,44)	A2 (20,A2/20,A2)	82 (20,82/20,82)	04 (20,04/20,04)
Z z	Я я	7A (2C,7A/2C,7A)	5A (2C,5A/2C,5A)	EF (2C,EF/2C,EF)	9F (2C,9F/2C,9F)	1A (2C,1A/2C,1A)
F f	А а	66 (21,66/21,66)	46 (21,46/21,46)	A0(21,A0/21,A0)	80 (21,80/21,80)	06 (21,06/21,06)
G g	П п	67 (22,67/22,67)	47 (22,47/22,47)	AF(22,AF/22,AF)	8F (22,8F/22,8F)	07 (22,07/22,07)
H h	Р р	68 (23,68/23,68)	48 (23,48/23,48)	E0 (00,E0/00,E0)	90 (23,90/23,90)	08 (23,08/23,08)
J j	О о	6A (24,6A/24,6A)	4A (24,4A/24,4A)	AE (24,AE/24,AE)	8E (24,8E/24,8E)	0A (24,0A/24,0A)

Таблица 1.14 *(продолжение)*

Клавиша		Регистр				
		Латинский		Русский		Л/Р[2]
K k	Л л	6B (25,6B/25,6B)	4B (25,4B/25,4B)	AB (25,AB/25,AB)	8B (25,8B/25,8B)	0B (25,0B/25,0B)
L l	Д д	6C (26,6C/26,6C)	4C (26,4C/26,4C)	A4 (26,A4/26,A4)	84 (26,84/26,84)	0C (26,0C/26,0C)
; :	Ж ж	3B (27,3B/27,3B)	3A (27,3A/27,3A)	A6 (27,A6/27,A6)	86 (27,86/27,86)	-
' "	Э э	27 (28,27/28,27)	22 (28,22/28,22)	ED (28,ED/28,ED)	9D (28,9D/28,9D)	-
X x	Ч ч	78 (2D,78/2D,78)	58 (2D,58/2D,58)	E7 (2D,E7/2D,E7)	97 (2D,97/2D,97)	18 (2D,18/2D,18)
C c	С с	63(2E,63/2E,63)	43 (2E,43/2E,43)	E1(2E,E1/2E,E1)	91 (2E,91/2E,91)	03 (2E,03/2E,03)
V v	М м	76 (2F,76/2F,76)	56 (2F,56/2F,56)	AC (2F,AC/2F,AC)	8C (2F,8C/2F,8C)	16 (2F,16/2F,16)
B b	И и	62 (30,62/30,62)	42 (30,42/30,42)	A8[8] (30,A8/30,A8)	88 (30,88/30,88)	02 (30,02/30,02)
N n	Т т	6E (31,6E/31,6E)	4E (31,4E/31,4E)	E2 (31,E2/31,E2)	92 (31,92/31,92)	0E (31,0E/310E)
M m	Ь ь	6D (32,6D/32,6D)	4D (32,4D/32,4D)	EC (32,EC/32,EC)	9C (32,9C/32,9C)	0D (32,0D/32,0D)
, <	Б б	2C (33,2C/33,2C)	3C (33,3C/33,3C)	A1 (33,A1/33,A1)	81 (33,81/33,81)	-
. >	Ю ю	2E (34,2E/34,2E)	3E (34,3E/34,3E)	EE (34,EE/34,EE)	9E (34,9E/34,9E)	-
/ ?	. , (Ё ё)	2F (35,2F/35,2F)	3F (35,3F/35,3F)	2F (35,2E/35,2E)	3F (35,2C/35,2C)	-
\ |		5C (2B,5C/2B,5C)	7C (2B,7C/2B,7C)	5C(2B,5C/2B,5C)	2F (2B,2F/2B,2F)	1C (2B,1C/2B,1C)
Keypad * (PrtSc)		2A (37,2A/37,2A)	(37,2A/37,2A)	2A (37,2A/37,2A)	(37,2A/37,2A)	EXT(-/96,00)
Space		20 (39,20/39,20)	20 (39,20/39,20)	20 (39,20/39,20)	20 (39,20/39,20)	20 (39,20/39,20)

[1] Для клавиш букв алфавитов при включенном *CapsLock* колонки Norm и Shift меняются местами.
[2] Коды не зависят от русского/латинского регистров.
EXT - расширенный ASCII-код.

Таблица 1.15. ASCII-коды цифровой клавиатуры

Клавиша	Альтерация		
	Norm.[1]	Shift[1]	Ctrl
/	2F (35,2F/E0,2F)	2F (35,2F/E0,2F)	EXT[2,3](-/95,00)
*	2A (37,2A/37,2A)	2A (37,2A/37,2A)	EXT[2,3](-/96,00)
-	2D (4A,2D/4A,2D)	2D (4A,2D/4A,2D)	EXT[2,3](-/8E,00)
+	2B (4E,2B/4E,2B)	2B (4E,2B/4E,2B)	EXT[2,3](-/90,00)
Enter	0D (1C,0D/E0,0D)	0D (1C,0D/1C,0D)	EXT(1C,0A/E0,0A)
Home [7]	37 (47,37/47,37)	EXT[2](47,00/47,00)	EXT[4] (77,00/77,00)
↑ [8]	38 (48,38/48,38)	EXT[2](48,00/48,00)	EXT[3,4](-/8D,00)
PgUp [9]	39 (49,39/49,39)	EXT[2](49,00/49,00)	EXT[4](84,00/84,00)
← [4]	34 (4B,34/4B,34)	EXT[2](4B,00/4B,00)	EXT[4](73,00/73,00)
5 [5]	35 (4C,35/4C,35)	EXT[2,3](-/4C,00)	EXT[3,4](-/8F,00)

→ [6]	36 (4D,36/4D,36)	EXT[2](4D,00/4D,00)	EXT[4](74,00/74,00)
End [1]	31 (4F,31/4F,31)	EXT[2](4F,00/4F,00)	EXT[4](75,00/75,00)
↓ [2]	32 (50,32/50,32)	EXT[2](50,00/50,00)	EXT[3,4](-/91,00)
PgDn [3]	33 (51,33/51,33)	EXT[2](51,00/51,00)	EXT[4](76,00/76,00)
Ins [0]	30 (52,30/52,30)	EXT[2](52,00/52,00)	EXT[3,4](-/92,00)
Del [.]	2E (53,2E/53,2E)	EXT[2](53,00/53,00)	EXT[3,4](-/93,00)

[1] При включенном *NumLock* колонки Norm и Shift меняются местами.

[2] Расширенный ASCII-код.

[3] Доступны только через функции 10h - 12h прерывания INT 16h.

[4] Расширенный ASCII-код, отличный от EXT.

Расширенный ASCII-код (Extended ASCII keystroke) вырабатывается при нажатие функциональных клавиш и комбинаций управляющих и символьных клавиш.

В нижеприведенных таблицах (1.16 – 1.19) приняты обозначения: «*» — доступны только через функции 10h-12h INT 16h.

При сочетание клавиш альтерации действует только одна с высшим приоритетом. Приоритеты в порядке убывания: ALT, CTRL, SHIFT.

Таблица 1.16. Расширенные ASCII-коды функциональных клавиш

Клави-ша	Normal		Shift		Ctrl		Alt	
	Hex (00h/10h)	Dec	Hex (00h/10h)	Dec	Hex (00h/10h)	Dec	Hex (00h/10h)	Dec
F1	3B (3B,00/3B00)	59	54 (54,00/54,00)	84	5E (5E,00/5E,00)	94	68 (68,00/68,00)	104
F2	3C (3C,00/3C,00)	60	55 (55,00/55,00)	85	5F (5F,00/5F,00)	95	69 (69,00/69,00)	105
F3	3D (3D,00/3D,00)	61	56 (56,00/56,00)	86	60 (60,00/60,00)	96	6A (6A,00/6A,00)	106
F4	3E (3E,00/3E,00)	62	57 (57,00/57,00)	87	61 (61,00/61,00)	97	6B (6B,00/6B,00)	107
F5	3F (3F,00/3F,00)	63	58 (58,00/58,00)	88	62 (62,00/62,00)	98	6C (6C,00/6C,00)	108
F6	40 (40,00/40,00)	64	59 (59,00/59,00)	89	63 (63,00/63,00)	99	6D (6D,00/6D,00)	109
F7	41 (41,00/41,00)	65	5A (5A,00/5A,00)	90	64 (64,00/64,00)	100	6E (6E,00/6E,00)	110
F8	42 (42,00/42,00)	66	5B (5B,00/5B,00)	91	65 (65,00/65,00)	101	6F (6F,00/6F,00)	111
F9	43 (43,00/43,00)	67	5C (5C,00/5C,00)	92	66 (66,00/66,00)	102	70 (70.00/70,00)	112
F10	44 (44,00/43,00)	68	5D (5D,00/5D,00)	93	67 (67,00/67,00)	103	71 (71,00/71,00)	113
F11*	85 (-/85,00)	133	87 (-/87,00)	135	89 (-/89,00)	137	8B (-/8B,00)	139
F12*	86 (-/86,00)	134	88 (-/88,00)	136	8A (-/8A,00)	138	8C (-/8C,00)	140

Таблица 1.17. Расширенные ASCII-коды Alt-комбинаиий символьных клавиш

Клави-ша	**Hex**	**Dec**	**Клави-ша**	**Hex**	**Dec**	**Клави-ша**	**Hex**	**Dec**
A	1E (1E,00/1E,00)	30	Q	10 (10,00/10,00)	16	5	7C (7C,00/7C,00)	124
B	30 (30,00/30,00)	48	R	13 (13,00/13,00)	19	6	7D (7D,00/7D,00)	125
C	2E (2E,00/2E,00)	46	S	1F (1F,00/1F,00)	31	7	7E (7E,00/7E,00)	126
D	20 (20,00/20,00)	32	T	14 (14,00/14,00)	20	8	7F (7F,00/7F,00)	127
E	12 (12,00/12,00)	18	U	16 (16,00/16,00)	22	9	80 (80,00/80,00)	128
F	21 (21,00/21,00)	33	V	2F (2F,00/2F,00)	47	-	82 (82,00/82,00)	130
G	22 (22,00/22,00)	34	W	11 (11,00/11,00)	17	-	83 (83,00/83,00)	131
H	23 (23,00/23,00)	35	X	2D (2D,00/2D,00)	45	[	1A* (-/1A,00)	26
I	17 (17,00/17,00)	23	Y	15 (15,00/15,00)	21	]	1B* (-/1B,00)	27
J	24 (24,00/24,00)	36	Z	2C (2C,00/2C,00)	44	;	27* (-/27,00)	39
K	25 (25,00/25,00)	37	-	-	-	'	28* (-/28,00)	40
L	26 (26,00/26,00)	38	0	81 (81,00/81,00)	129	`	29* (-/29,00)	41
M	32 (32,00/32,00)	50	1	78 (78,00/78,00)	120	\	2B* (-/2B,00)	43
N	31 (31,00/31,00)	49	2	79 (79,00/79,00)	121	Запятая	33* (-/33,00)	51
0	18 (18,00/18,00)	24	3	7A (7A,00/7A,00)	122	.	34* (-/34,00)	52
P	19 (19,00/19,00)	25	4	7B (7B,00/7B,00)	123	/	35* (-/35)	53

Таблица 1.18. Расширенные ASCII-коды дополнительной клавиатуры

Клавиша	Normal		Ctrl		Alt	
	Hex	Dec	Hex	Dec	Hex	Dec
Insert	52 (52,00/52,E0)	82	92* (-/92,E0)	147	A2* (-/A2,00)	162
Delete	53 (53,00/53,E0)	83	93* (-/93,E0)	148	A3* (-/A3,00)	163
End	4F (4F,00/4F,E0)	79	75 (75,00/75,E0)	117	9F* (-/9F,00)	159
Home	47 (47,00/47,E0)	71	77 (77,00/77,E0)	119	97* (-/97,00)	151
Page Down	51 (51,00/51,E0)	81	76 (76,00/76,E0)	118	A1* (-/A1,00)	161
Page Up	49 (49,00/49,E0)	73	84 (84,00/84,E0)	132	99* (-/99,00)	153
←	4B (4B,00/4B,E0)	75	73 (73,00/73,E0)	115	9B* (-/9B,00)	155
→	4D (4D,00/4D,E0)	77	74 (74,00/74,E0)	116	9D* (-/9D,00)	157
↑	48 (48,00/48,E0)	72	8D* (-/8D,E0)	143	98* (-/98,00)	152
↓	50 (50,00/50,E0)	80	91* (-/91,E0)	146	A0* (-/A0,00)	160

Таблица 1.19. Расширенные ASCII-коды цифровой клавиатуры

Клавиша	Normal[1]		Ctrl		Alt	
	Hex	Dec	Hex	Dec	Hex	Dec
/	NORM.[2]	-	95* (-/95,00)	149	A4* (-/A4,00)	164
*	NORM.[2]	-	96* (-/96,00)	150	37* (-/37,00)	55
-	NORM.[2]	-	8E* (-/8E,00)	142	4A* (-/4A,00)	74
+	NORM.[2]	-	90* (-/90,00)	144	4E* (-/4E,00)	78
Enter	NORM.[2]	-	NORM.[2]	-	A6* (-/A6,00)	166
Home [7]	47 (47,00/47,00)	71	77 (77,00/77,00)	119	-	-
↑ [8]	48 (48,00/48,00)	72	8D* (-/8D,00)	141	-	-
PgUp [9]	49 (49,00/49,00)	73	84 (84,00/84,00)	132	-	-
← [4]	4B (4B,00/4B,00)	75	73 (73,00/73,00)	115	-	-
5 [5]	4C* (-/4C,00)	76	8F* (-/8F,00)	143	-	-
→ [6]	4D (4D,00/4D,00)	77	74 (74,00/74,00)	116	-	-
End [1]	4F (4F,00/4F,00)	74	75 (74,00/74,00)	117	-	-
↓ [2]	50 (50,00/50,00)	80	91* (-/91,00)	145	-	-
PgDn [3]	51 (51,00/51,00)	81	76 (76,00/76,00)	118	-	-
Ins [0]	52 (52,00/52,00)	82	92* (-/92,00)	146	-	-
Del [.]	53 (53,00/53,00)	83	93* (-/93,00)	147	-	-

[1]. NUMLOCK отключен или нажата клавиша SHIFT при включенном NUMLOCK.
[2] Стандартные скан-коды и ASCII-коды.

Таблица 1.20. Расширенный ASCII-код прочих комбинаций

Клавиша	Hex	Dec	Клавиша	Hex	Dec
Alt+Esc	01* (-/01,00)	1	Ctrl+Break	00 (00,00/00,00)	0
Shift+Tab	0F (0F,00/0F,00)	15	Alt+Bakspace	0E* (-/0E,00)	14
Alt+Tab	A5* (-/A5,00)	165	Alt+Enter	1C* (-/1C,00)	28
Ctrl+Tab	94* (-/94,00)	148	Ctrl+Print Screen	72 (72,00/72,00)	114
Ctrl+@	03 (03,00/03,00)	3	Ctrl+^	1E (1E,00/1E,00)	30

1.2.6.5. Функции DOS обработки клавиатуры

Эти функции (подфункции INT 21h) обычно используются программами, которые должны обеспечивать максимальную совместимость со всеми типами ПК, самой операционной системой и ее утилитами. В DOS клавиатура обычно назначается стандартным устройством ввода и ей присваивается указатель 0. Если средствами DOS стандартное устройство ввода переназначено (на файл или последовательный порт), клавиатура становится недоступной для работы через INT 21h.

Ожидание поступления и чтение символа со стандартного устройства ввода с эхом на стандартное устройство вывода и обработкой Ctrl Break

Вход: AH = 01h.

Выход: AL - код символа, клавиши типа F11 необходимо обрабатывать за два вызова.

Чтение со стандартного устройства ввода (без ожидания) или запись на стандартное устройство вывода

Нажатие Ctrl Break не обрабатывается.

Вход: AH = 06h;
DL = FFh, если необходимо прочитать символ,
DL = 0-FEh, если необходимо вывести этот код.

Выход: если ZF сброшен, то символ был прочитан и его код хранится в AL (при вызове с DL=FFh).

Ожидание поступления и чтение символа со стандартного устройства ввода без эха и обработки Ctrl Break

Вход: AH = 07h.

Выход: AL - код символа, клавиши типа F11 необходимо обрабатывать за два вызова.

Ожидание поступления и чтение символа со стандартного устройства ввода без эха, но с обработкой Ctrl Break

Вход: AH = 08h.

Выход: AL - код символа, клавиши типа F11 необходимо обрабатывать за два вызова.

Буферизированный ввод со стандартного устройства ввода

Эта подфункция обеспечивает чтение строки символов с несложным редактированием вводимой строки и обработкой нажатия Ctrl Break.

Вход: AH = 0Ah;
DS:DX - адрес для записи вводимой строки (в первом байте - максимальная длина строки).

Выход: в буфер занесена строка (во втором - байте длина прочитанной строки).

Проверить стандартное устройство ввода

Обрабатывается нажатие Ctrl Break.

Вход: AH = 0Bh.

Выход: AH = 00h, если есть символ,
AH = FFh, если символа нет.

Чистка буфера стандартного устройства ввода и чтение символа с помощью указанной функции

Если указан не допустимый номер функции, то буфер будет очищен, но функция не вызывается.

Вход: AH = 0Ch;
AL - номер функции DOS (01h, 06h, 07h, 08h, 0Ah).

Выход: в соответствии с вызываемой подфункцией.

Прочитать данные с клавиатуры можно также посредством функций DOS 3Fh и 40h.

1.2.7. Примеры программирования клавиатуры

В разделе приводятся примеры работы с клавиатурой с использованием функций прерывания Int16h BIOS и непосредственно через порты адаптера клавиатуры. Программы и процедуры написаны на ассемблере. Приведенные программы рассчитаны на свободный доступ к памяти и аппаратуре, то есть должны выполняться в реальном режиме работы MS-DOS или в режиме эмуляции MS-DOS, который имеется в Windows 95/98. Ряд программных модулей сориентированы на выполнение в среде полноэкранного отладчика типа AFD.

1.2.7.1. Примеры использования функций BIOS.

Ниже приведено несколько программ, иллюстрирующих работу с клавиатурой через функции прерывания Int16h BIOS и ряд процедур, обеспечивающих отображение на экране монитора результатов взаимодействия оператора с программами через клавиатуру.

Большинство программ сориентированы на DOS 6.22 и DOS 7.0. Приводятся также программные модули, сориентированные на выполнение в среде полноэкранного отладчика типа AFD.

1.2.7.1.1. Программа тестирования функции 00h.

Пример, показанный в листинге 1.8, предназначен для демонстрации особенностей функции 00h прерывания Int 16h. Программа Testlntl6_00h выполняет одну простую операцию — отображает на экране ASCII-коды и скан-коды клавиш, которые пользователь нажимает на клавиатуре [6].

Программа использует текстовый видеорежим. Вывод информации осуществляется напрямую в видеопамять, которая в цветном текстовом режиме размещается по адресу B8000h. Каждому символу экрана соответствует байт кода (ASCII) и байт атрибута, описывающий цвет символа и цвет фона знакоместа. Видеопамять при выводе на экран разбивается по строкам: в строке 80 символов, то есть 160 байт информации; вывод выполняется слева направо, сверху вниз. Таким образом, байт видеопамяти с нулевым номером соответствует ASCII-коду первого символа, следующий байт определяет его цвет, третий байт соответствует ASCII-коду второго символа и т. д. Все основные процедуры ввода-вывода, используемые в программе, подключаются из файла list1_02.inc (см. раздел 1.2.7.1.2).

Листинг 1.8. **Тестирование функции 00h ввода с клавиатуры (TestIntl6_00h).**

```
IDEAL
P386
LOCALS
MODEL MEDIUM

SEGMENT sseg para stack 'STACK'
DB 400h DUP(?)
ENDS

DATASEG
; Счетчик операций нажатия/отпускания клавиш
PressCounter DW ?
; Текстовые сообщения
Text DB 0,18
     DB "Тестирование функции ввода с клавиатуры 00h",0
     DB 2,0,"ASCII-код  Скан-код",0
     DB 24,29,"Нажмите любую клавишу",0
ENDS

CODESEG
;*****************************
;* Основной модуль программы *
;*****************************
PROC TestInt16_00h
        mov     AX,DGROUP
        mov     DS,AX
        mov     [CS:MainDataSeg],AX
; Установить текстовый режим и очистить экран
        mov     AX,3
        int     10h
; Скрыть курсор - убрать за нижнюю границу экрана
        mov     [ScreenString],25
        mov     [ScreenColumn],0
        call    SetCursorPosition
; Вывести текстовые сообщения на экран
        mov     CX,3
        mov     SI,offset Text
@@OutText:
        call    ShowString
        loop    @@OutText
; Установить начальную позицию для вывода кодов
; в нулевой колонке пятой строки
        mov     [ScreenString],3
        mov     [ScreenColumn],0
; Инициализировать счетчик операций
; нажатия/отпускания клавиш
        mov     [PressCounter],0
; Принять очередную пару кодов с клавиатуры
@@Next: mov     AH,0
        int     16h
; Отобразить принятые коды в шестнадцатеричном виде
        ; Отобразить ASCII-код (начиная с 3-й колонки)
        mov     [ScreenColumn],3
        call    ShowHexByte
        ; Отобразить скан-код (начиная с 14-й колонки)
        mov     [ScreenColumn],14
        mov     AL,AH
        call    ShowHexByte
        ; Перейти на следующую строку

        inc     [ScreenString]
        ; Увеличить значение счетчика нажатий клавиш
        inc     [PressCounter]
        ; После 20 нажатий выйти из цикла
        cmp     [PressCounter],20
        jb      @@Next

; Переустановить текстовый режим и очистить экран
        mov     AX,3
        int     10h
; Выход в DOS
        mov     AH,4Ch
        int     21h
ENDP TestInt16_00h
ENDS

; Подключить процедуры вывода данных на экран
include "list1_02.inc"

END
```

1.2.7.1.2. Процедуры вывода данных на экран и ввода информации с клавиатуры

В программе TestIntl6_00h [6] используется несколько вспомогательных процедур, не относящихся непосредственно к клавиатурным функциям BIOS, но необходимых для отображения информации на экране. Подобные процедуры общего назначения объединены в отдельную группу, оформленную в виде подключаемого файла (include-файла), приведенного в листинге 1.9. В последующих примерах данный файл также будет использоваться, что позволяет избежать многократного дублирования кодов. Назначение подпрограмм, включенных в листинг 1.9, следующее:

- процедура ShowHexByte отображает в заданную позицию экрана содержимое регистра AL (байт данных) в шестнадцатеричном коде;
- процедура ShowString выводит текстовую строку в заданную область экрана, причем используется цвет символов и фона, заданный по умолчанию;
- процедура ShowText использует процедуру ShowString для вывода текста (группы строк, цвет которых определяется по умолчанию) на экран;
- процедура ShowColorString выводит текстовую строку заданного цвета в заданную область экрана;
- процедура ShowColorText использует процедуру ShowColorString для вывода разноцветного текста (группы строк, цвет каждой из которых задается индивидуально) на экран;
- процедура SetCursorPosition предназначена для управления положением курсора, но в данном примере используется только для того, чтобы убрать курсор за пределы экрана (сделать курсор невидимым);
- процедура GetChar ожидает ввода любого символа с клавиатуры, а затем считывает ASCII-код и скан-код этого символа;
- процедура WaitChar проверяет наличие символа в буфере клавиатуры, и если символ есть — считывает его ASCII-код и скан-код;
- процедура C1earScreen осуществляет очистку экрана;

- процедура Beep предназначена для выдачи оператору звукового сигнала (обычно в случае **какой-**либо ошибки);
- процедура ShowASCIIField служит для вывода на экран текстового поля фиксированного размера;
- процедура FatalError осуществляет выдачу сообщения о фатальной ошибке, после чего производится экстренное (аварийное) прекращение работы программы.

Процедуры вывода информации на экран используют метод прямой записи в видеопамять с учетом особенностей текстового режима:

- каждый символ кодируется в видеопамяти двумя байтами: первый байт содержит код символа, а второй описывает цвет символа и фон знакоместа;
- видеопамять отображена на адресное пространство процессора до адреса 0B8000h (для доступа к ней обычно применяются сегментный регистр ES, в который записывается число B800h, и любой индексный регистр, в который заносится смещение символа от начала видеопамяти);
- длина одной строки составляет 80 символов (160 байт);
- на экране помещается 25 текстовых строк.

Листинг 1.9. Процедуры ввода-вывода общего назначения для работы в цветном текстовом режиме (подключаемый файл list1_02.inc).

```
DATASEG
; Цвет и фон выводимого текста (по умолчанию установим
; вывод белого текста по черному фону)
TextColorAndBackground DB 0Fh
; Начальная позиция для вывода текстовой строки на экран
ScreenString DW ?
ScreenColumn DW ?
ENDS

CODESEG
; Адрес основного сегмента данных
MainDataSeg DW ?
;*******************************************************
;*    ВЫВОД БАЙТА НА ЭКРАН В ШЕСТНАДЦАТЕРИЧНОМ КОДЕ    *
;* Подпрограмма выводит содержимое регистра AL         *
;* в шестнадцатеричном коде в заданную позицию экрана. *
;* Координаты позиции передаются через глобальные      *
;* переменные ScreenString и ScreenColumn. После       *
;* выполнения операции вывода происходит автомати-     *
;* ческое приращение значений этих переменных.         *
;*******************************************************
PROC ShowHexByte near
        pusha
        push    DS
        push    ES
; Настроить DS на глобальный сегмент данных
        mov     DI,[CS:MainDataSeg]
        mov     DS,DI
        cld

; Настроить пару ES:DI для прямого вывода в видеопамять
        push    AX
        ; Загрузить адрес сегмента видеоданных в ES
        mov     AX,0B800h
        mov     ES,AX
        ; Умножить номер строки на длину строки в байтах
        mov     AX,[ScreenString]
        mov     DX,160
        mul     DX
        ; Прибавить к полученному произведению номер
        ; колонки (дважды)
        add     AX,[ScreenColumn]
        add     AX,[ScreenColumn]
        ; Переписать результат в индексный регистр
        mov     DI,AX
        pop     AX

; Использовать цвет символов, заданный по умолчанию
        mov     AH,[TextColorAndBackground]
; Вывести старший разряд числа
        push    AX
        ; Выделить старший разряд
        shr     AL,4
        ; Преобразовать старший разряд в код ASCII
        add     AL,'0'
        cmp     AL,'9'
        jbe     @@M0
        add     AL,'A'-'9'- 1
        ; Вывести разряд числа на экран
@@M0:   stosw
        pop     AX
; Вывести младший разряд числа
        ; Выделить младший разряд числа
        and     AL,0FH
        ; Преобразовать младший разряд в код ASCII
        add     AL,'0'
        cmp     AL,'9'
        jbe     @@M1
        add     AL,'A'-'9'- 1
        ; Вывести разряд числа на экран
@@M1:   stosw

; Подготовка для вывода следующих байтов
        ; Перевести текущую позицию на 2 символа влево
        add     [ScreenColumn],2
        ; Проверить пересечение правой границы экрана
        cmp     [ScreenColumn],80
        jb      @@End
        ; Если достигнута правая граница экрана -
        ; перейти на следующую строку
        sub     [ScreenColumn],80
        inc     [ScreenString]
@@End:  pop     ES
        pop     DS
        popa
        ret
ENDP ShowHexByte
;*******************************************************
;*          ВЫВОД ТЕКСТОВОЙ СТРОКИ НА ЭКРАН            *
;* Все параметры передаются через одну структуру:      *
;* первый байт - номер начальной строки (0-24);        *
;* второй байт - номер начальной колонки (0-79);       *
;* далее идет строка, ограниченная нулем.              *
;* Адрес структуры передается через регистры DS:SI.    *
;*******************************************************
PROC ShowString near
        push    AX
        push    BX
        push    DI
        push    ES
; Настроить регистр ES на глобальный сегмент данных
```

```
        mov     AX,[CS:MainDataSeg]
        mov     ES,AX
; Запомнить цвет текста в BL
        mov     BL,[ES:TextColorAndBackground]
; Настроить регистр ES на видеопамять
        mov     AX,0B800h
        mov     ES,AX
        cld
; Вычислить адрес для строки в видеопамяти
        ; Загрузить номер строки экрана в AL и
        ; умножить его на длину строки в байтах
        lodsb
        ; Проверка: номер строки не должен превышать
        ; предела нижней границы экрана
        cmp     AL,24
        ja      @@Error
        mov     AH,160
        mul     AH
        ; Переписать результат в индексный регистр DI
        mov     DI,AX
        ; Загрузить номер столбца и дважды
        ; прибавить его к DI
        lodsb
        cmp     AL,79   ;номер колонки не должен
        ja      @@Error ;превышать ширины экрана
        mov     BH,AL   ;запомнить номер колонки
        xor     AH,AH   ;обнулить AH
        add     DI,AX
        add     DI,AX
        ; Загрузить атрибут цвета в AH
        mov     AH,BL
@@L1:   ; Загрузить очередной символ строки в AL
        lodsb
        ; Проверка на 0 (на конец строки)
        and     AL,AL
        jz      @@L2
        ; Проверить номер колонки символа
        cmp     BH,79
        ja      @@Error ;нарушена правая граница экрана
        ; Вывести символ на экран
        stosw
        inc     BH      ;увеличить номер колонки
        jmp     @@L1
@@L2:   pop     ES
        pop     DI
        pop     BX
        pop     AX
        ret
@@Error: ;Немедленный выход в DOS по ошибке
        mov     AH,4Ch
        int     21h
ENDP ShowString

;*********************************************************
;*         ВЫВОД ТЕКСТА (ГРУППЫ СТРОК) НА ЭКРАН          *
;* ShowText использует процедуру ShowString для вывода   *
;* на экран группы строк.                                *
;* Параметры:                                            *
;* CX - количество строк;                                *
;* DS:SI - адрес первой строки в группе.                 *
;* Строки должны иметь заданный для ShowString формат    *
;* и располагаться в памяти последовательно.             *
;* При выводе текста используются принятые по            *
;* умолчанию цвет и фон.                                 *
;*********************************************************
PROC ShowText near
; Цикл вывода строк
@@NextString:
        call    ShowString
        loop    @@NextString
; Процедура не сохраняет значения в CX и SI
        ret
ENDP ShowText

;*************************************************
;*  ВЫВОД ТЕКСТОВОЙ СТРОКИ ЗАДАННОГО ЦВЕТА НА ЭКРАН  *
;* Все параметры передаются через одну структуру:    *
;* первый байт - атрибут цвета и фона для строки;    *
;* второй байт - номер начальной строки (0-24);      *
;* третий байт - номер начальной колонки (0-79);     *
;* далее идет строка, ограниченная нулем.            *
;* Адрес структуры передается через регистры DS:SI.  *
;*************************************************
PROC ShowColorString near
        push    AX
        ; Запомнить цвет, используемый по умолчанию
        mov     AL,[TextColorAndBackground]
        push    AX
        ; Установить цвет строки
        cld
        lodsb
        mov     [TextColorAndBackground],AL
        ; Использовать функцию ShowString
        call    ShowString
        ; Восстановить цвет, используемый по умолчанию
        pop     AX
        mov     [TextColorAndBackground],AL
        pop     AX
        ret
ENDP ShowColorString

;*****************************************************
;*    ВЫВОД ЦВЕТНОГО ТЕКСТА (ГРУППЫ СТРОК) НА ЭКРАН    *
;* ShowColorText использует процедуру ShowColorString  *
;* для вывода на экран группы разноцветных строк.      *
;* Параметры:                                          *
;* CX - количество строк;                              *
;* DS:SI - адрес первой строки в группе.               *
;* Строки должны иметь заданный для ShowColorString    *
;* формат и располагаться в памяти последовательно.    *
;* При выводе текста используются принятые по          *
;* умолчанию цвет и фон.                               *
;*****************************************************
PROC ShowColorText near
; Цикл вывода строк
@@NextColorString:
        call    ShowColorString
        loop    @@NextColorString
; Процедура не сохраняет значения в CX и SI
        ret
ENDP ShowColorText

;*******************************
;*  УСТАНОВИТЬ ПОЗИЦИЮ КУРСОРА  *
;* Входные параметры:          *
;* ScreenString - номер строки  *
;* ScreenColumn - номер столбца *
;*******************************
PROC SetCursorPosition NEAR
        pusha
; Вычисление линейного адреса курсора
        mov     AX,[ScreenString]
        mov     BX,80
        mul     BX
        add     AX,[ScreenColumn]
        mov     BL,AL ;запомнить младший байт
; Прямой вывод позиции курсора
; в регистры видеоконтроллера
        mov     DX,3D4h
        ; Вывести старший байт адреса курсора
        mov     AL,0Eh
        out     DX,AX
        ; Вывести младший байт адреса курсора
        inc     AL
        mov     AH,BL
        out     DX,AX
```

```
        popa
        ret
ENDP SetCursorPosition

;*********************************************
;*       ПРИНЯТЬ СИМВОЛ ОТ КЛАВИАТУРЫ        *
;* Процедура осуществляет ввод символа с     *
;* помощью функции 00h прерывания Int16h.    *
;* Для "текстовых" управляющих клавиш вместо *
;* скан-кодов используются ASCII-коды.       *
;* Входных параметров нет.                   *
;* Функция возвращает:                       *
;* AL - код символа;                         *
;* AH - управляющий код, если в AL ноль.     *
;*********************************************
PROC GetChar NEAR
; Очистить буфер клавиатуры
@@ClearBuffer:
        mov     AH,1
        int     16h
        jz      @@WaitChar
        mov     AH,0
        int     16h
        jmp short @@ClearBuffer
; Ожидать нажатия клавиши и принять код символа
@@WaitChar:
        mov     AH,0
        int     16h
; Обработать принятый код
        and     AL,AL
        jnz     @@Get1
        ret     ;(в AL - ноль, в AH - управляющий код)
@@Get1: cmp     AL,32
        jnb     @@Get2
        ; Переписать в AH управляющий код
        xchg    AL,AH
        mov     AL,0
        ret     ;(в AL - ноль, в AH - управляющий код)
@@Get2: mov     AH,0
        ret     ;(в AL - код буквы, в AH - ноль)
ENDP GetChar

;*********************************************
;*       ПРИНЯТЬ СИМВОЛ ОТ КЛАВИАТУРЫ,       *
;*           ЕСЛИ ОН ЕСТЬ В БУФЕРЕ           *
;* Процедура проверяет наличие символа в буфере *
;* клавиатуры и считывает его, если он есть.  *
;* Входных параметров нет.                   *
;* Функция возвращает:                       *
;* AL - код символа;                         *
;* AH - управляющий код, если в AL ноль.     *
;* Если в AL и AH нули - нажатий не было.    *
;*********************************************
PROC WaitChar NEAR
; Проверить наличие символа в буфере клавиатуры
        mov     AH,1
        int     16h
        jz      @@NoInput
; Принять символ от клавиатуры
        mov     AX,0
        int     16h
        and     AL,AL
        jnz     @@GET1
        ret     ;в AL - ноль, в AH - управляющий код
@@GET1: cmp     AL,32
        jnb     @@GET2
        mov     AH,AL ;переписать в AH управляющий код
        mov     AL,0
        ret     ;в AL - ноль, в AH - управляющий код
@@GET2: mov     AH,0
        ret     ;в AL - код буквы, в AH - ноль
@@NoInput:
        xor     AX,AX
        ret     ;в AL и AH - нули
ENDP WaitChar

;*********************************
;* ОЧИСТКА ЭКРАНА В ТЕКСТОВОМ РЕЖИМЕ *
;*  (процедура параметров не имеет)  *
;*********************************
PROC ClearScreen NEAR
        pusha
        push    ES
; Настроить ES:DI на "текстовую" область видеопамяти
        mov     AX,0B800h
        mov     ES,AX
        cld
        mov     DI,0
; Вывести 2000 "пустых" символов (ASCII-код 0) с
; атрибутом "белый цвет, черный фон"
        mov     CX,2000
         mov    AX,0F00H
         rep    stosw
         pop    ES
         popa
         ret
 ENDP ClearScreen

 ;*****************************************************
 ;* ПОДАЧА ЗВУКОВОГО СИГНАЛА ЧЕРЕЗ ВСТРОЕННЫЙ ДИНАМИК *
 ;*          (процедура параметров не имеет)          *
 ;*****************************************************
 PROC Beep NEAR
         push   AX
         push   DX
 ; Послать на терминал код "звонок" (07h)
         mov    AH,2
         mov    DL,7
         int    21h
         pop    DX
         pop    AX
         ret
 ENDP Beep

 ;*****************************************************
 ;*      ВЫВОД НА ЭКРАН ТЕКСТОВОГО ПОЛЯ ДАННЫХ        *
 ;* Передаваемые параметры:                           *
 ;* DS:SI - указатель на структуру данных;            *
 ;* BX - смещение поля от начала структуры;           *
 ;* CX - длина поля в байтах.                         *
 ;* Цвет задается переменной TextColorAndBackground.  *
 ;* Координаты позиции передаются через глобальные    *
 ;* переменные ScreenString и ScreenColumn.           *
 ;*****************************************************
 PROC ShowASCIIField near
         pusha
         push   ES
         mov    AX,0B800h    ;Настроить ES для прямого
         mov    ES,AX        ;вывода на экран
 ; Установить указатель на начало поля
         add    SI,BX
 ; Вычислить начальную позицию в видеопамяти
         mov    AX,[ScreenString]
         mov    DI,160
         mul    DI
         add    AX,[ScreenColumn]
         add    AX,[ScreenColumn]
         mov    DI,AX
 ; Использовать цвет, заданный по умолчанию
         mov    AH,[TextColorAndBackground]
 ; Вывести поле на экран
 @@NextChar:
         lodsb
         stosw
         loop   @@NextChar
         pop    ES
```

```
        popa
        ret
ENDP ShowASCIIField

;*************************************************
;*       ВЫДАЧА СООБЩЕНИЯ О ФАТАЛЬНОЙ ОШИБКЕ    *
;*         И ЭКСТРЕННЫЙ ВЫХОД ИЗ ПРОГРАММЫ       *
;* Параметры:                                    *

;* DS:SI - указатель на строку сообщения об ошибке, *
;* представленную в формате ShowColorString.     *
;*************************************************
PROC FatalError near
; Переустановить текстовый режим и очистить экран
        mov     AX,3
        int     10h
; Настроить DS на глобальный сегмент данных
        mov     DI,[CS:MainDataSeg]
        mov     DS,DI
; Вывести сообщение об ошибке красным цветом
        mov     [TextColorAndBackground],12
        call    ShowString
; Переместить курсор в нижнюю часть экрана
        mov     [ScreenString],24

        mov     [ScreenColumn],0
        call    SetCursorPosition
; Аварийный выход из программы
        mov     AH,4Ch
        int     21h
ENDP FatalError
ENDS
```

1.2.7.1.3. Мнемоническое обозначение управляющих клавиш и цветовых кодов

Приведенная в листинге 1.9 процедура ввода символа GetChar выполняет определенные преобразования над данными, выдаваемыми функцией 00h по прерыванию Int 16h (прерывание Int 16h обеспечивает работу с клавиатурой с использованием функций BIOS). Дело в том, что некоторые из управляющих символов, перечисленных в табл. 1.21, традиционно имеют как ASCII-коды, так и скан-коды, причем значения этих кодов не совпадают.

Таблица 1.21. Управляющие символы ASCII-кода

Код символа	Мнемоническое обозначение	Назначение символа
00h	NUL	Пустой символ
01h	SOH	Начало заголовка (начало блока данных)
02h	STX	Начало текста
03h	ETX	Конец текста
04h	EOT	Конец передачи
05h	ENQ	Запрос подтверждения
06h	ACK	Подтверждение
07h	BEL	Звонок (звуковой сигнал)
08h	BS	Забой (возврат на одну позицию влево)
09h	HT	Горизонтальная табуляция
0Ah	LF	Перевод строки
0Bh	VT	Вертикальная табуляция
0Ch	FF	Перевод формата (переход к новой странице)
0Dh	CR	Возврат каретки
0Eh	SO	Переход на нижний регистр
0Fh	SI	Переход на верхний регистр
10h	DLE	Завершение сеанса связи
11h	DC1	Управление устройством № 1
12h	DC2	Управление устройством № 2
13h	DC3	Управление устройством № 3
14h	DC4	Управление устройством № 4
15h	NAK	Ошибка передачи
16h	SYN	Холостой ход передатчика
17h	ETB	Конец передачи блока
18h	CAN	Отмена
19h	EM	Конец носителя данных
1Ah	SUB	Подстановка (замена символа)
1Bh	ESC	Переход (посылка сложной команды)
1Ch	FS	Разделитель файлов
1Dh	GS	Разделитель групп
1Eh	RS	Разделитель записей
1Fh	US	Разделитель элементов
7Fh	DEL	Удаление символа

С целью упрощения последующего анализа кодов введенных символов процедура GetChar переносит ASCII-коды символов со значениями от 0 до 20h из регистра AL в регистр AH, заменяя соответствующие скан-коды (AL при этом обнуляется).

В листинге 1.10. даны мнемонические обозначения для кодов наиболее часто применяемых управляющих клавиш, которые процедура GetChar сохраняет в регистре AH (символ является управляющим, если в регистре AL был возвращен код 0). Кроме того, в листинг 1.10 включены мнемонические обозначения цветовых оттенков, которые будут применяться в программах, работающих в 16-цветных и 256-цветных режимах [6].

Листинг 1.10 Мнемонические обозначения кодов управляющих клавиш и цветовых оттенков (подключаемый файл list1_03.inc).

```
; КОДЫ УПРАВЛЯЮЩИХ КЛАВИШ
; Для "текстовых" управляющих клавиш вместо скан-кодов
; используются ASCII-коды:
B_RUBOUT equ   8 ;забой
B_TAB    equ   9 ;табуляция
B_LF     equ  10 ;перевод строки
B_ENTER  equ  13 ;возврат каретки
B_ESC    equ  27 ;"Esc"
; Скан-коды функциональных клавиш:
F1       equ  59
F2       equ  60
F3       equ  61
F4       equ  62
F5       equ  63
F6       equ  64
F7       equ  65
F8       equ  66
F9       equ  67
F10      equ  68
; Скан-коды клавиш дополнительной клавиатуры:
B_HOME   equ  71 ;перейти в начало
B_UP     equ  72 ;стрелка вверх
B_PGUP   equ  73 ;на страницу вверх
B_BS     equ  75 ;стрелка влево
B_FWD    equ  77 ;стрелка вправо
B_END    equ  79 ;перейти в конец
B_DN     equ  80 ;стрелка вниз
B_PGDN   equ  81 ;на страницу вниз
B_INS    equ  82 ;переключить режим (вставка/замещение)
B_DEL    equ  83 ;удалить символ над курсором

; МНЕМОНИЧЕСКИЕ ОБОЗНАЧЕНИЯ ЦВЕТОВ
; "Темные" цвета (можно использовать для фона и текста)
BLACK        equ  0 ;черный
BLUE         equ  1 ;темно-синий
GREEN        equ  2 ;темно-зеленый
CYAN         equ  3 ;бирюзовый (циан)
RED          equ  4 ;темно-красный
MAGENTA      equ  5 ;темно-фиолетовый
BROWN        equ  6 ;коричневый
LIGHTGREY    equ  7 ;серый
; "Светлые" цвета (только для текста)
DARKGREY     equ  8 ;темно-серый
LIGHTBLUE    equ  9 ;синий
LIGHTGREEN   equ 10 ;зеленый
LIGHTCYAN    equ 11 ;голубой
LIGHTRED     equ 12 ;красный
LIGHTMAGENTA equ 13 ;фиолетовый
YELLOW       equ 14 ;желтый
WHITE        equ 15 ;белый
```

1.2.7.1.4. Макрокоманды для вывода данных на экран.

Листинг 1.11 содержит набор макрокоманд, делающих более наглядными и компактными участки программного кода, в которых осуществляются операции ввода-вывода [6]. В макросах используются процедуры ввода-вывода общего назначения для работы в цветном текстовом режиме, приведенные в листинге 1.9.

Листинг 1.11. Макрокоманды для вывода данных на экран (подключаемый файл list1_04.inc).

```
;****************************************
;* ВЫВОД БАЙТА В ШЕСТНАДЦАТЕРИЧНОМ КОДЕ *
;* Параметры:                           *
;* SString - номер строки экрана;       *
;* SColumn - номер колонки экрана;      *
;* BData - отображаемый байт данных.    *
;****************************************
MACRO MShowHexByte SString,SColumn,BData
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     AL,BData
        call    ShowHexByte
ENDM
;******************************
;*   ВЫВОД ТЕКСТОВОЙ СТРОКИ   *
;* Параметры:                 *
;* TOffset - смещение строки. *
;******************************
MACRO MShowString TOffset
        mov     SI,offset TOffset
        call    ShowString
ENDM
;**********************************
;* ВЫВОД ЦВЕТНОЙ ТЕКСТОВОЙ СТРОКИ *
;* Параметры:                     *
;* TOffset - смещение строки.     *
;**********************************
MACRO MShowColorString TOffset
        mov     SI,offset TOffset
        call    ShowColorString
ENDM
;******************************************
;*        ВЫВОД ОДНОЦВЕТНОГО ТЕКСТА       *
;* Параметры:                             *
;* TStrings - количество строк в тексте;  *
;* TOffset - смещение первой строки текста. *
;******************************************
MACRO MShowText TStrings,TOffset
        mov     CX,TStrings
        mov     SI,offset TOffset
        call    ShowText
ENDM
;******************************************
;*        ВЫВОД РАЗНОЦВЕТНОГО ТЕКСТА      *
;* Параметры:                             *
;* TStrings - количество строк в тексте;  *
;* TOffset - смещение первой строки текста. *
;******************************************
MACRO MShowColorText TStrings,TOffset
        mov     CX,TStrings
        mov     SI,offset TOffset
        call    ShowColorText
ENDM
;******************************************
;*        ВЫВОД ТЕКСТОВОГО ПОЛЯ ДАННЫХ    *
;* Параметры:                             *
;* SString - номер строки экрана;         *
;* SColumn - номер колонки экрана;        *
;* DOffs - смещение поля от начала структуры; *
;* DSize - длина поля в байтах;           *
;* DS:SI - указатель на структуру данных. *
;******************************************
```

```
MACRO MShowASCIIField SString,SColumn,DOffs,DSize
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     BX,DOffs
        mov     CX,DSize
        call    ShowASCIIField
ENDM
;***************************************
;*       ВЫВОД СООБЩЕНИЯ ОБ ОШИБКЕ      *
;* Параметры:                          *
;* TOffset - смещение строки сообщения. *
;***************************************
```

```
MACRO MFatalError TOffset
        mov     SI,offset TOffset
        call    FatalError
ENDM
```

1.2.7.1.5. Программа тестирования функции 01h прерывания Int16h, обрабатываемого BIOS.

Программа Testlntl6_10h [6], показанная в листинге 1.12, демонстрирует особенности функции 10h. Основное отличие от программы в листинге 1.8 — в другом номере функции. В программе используются те же самые вспомогательные процедуры из листинга 1.9, но текстовые сообщения окрашены в разные цвета. Выделение цветом — простой и удобный прием, позволяющий оператору быстрее отыскивать на экране нужную информацию. Кроме того, различная окраска изображения позволяет избежать недоразумений при поочередном выполнение программ Testlntl6_00h и Testlntl6_10h, поскольку текст сообщений в этих процедурах практически совпадает (поочередное выполнение программ Testlntl6_00h и Testlntl6_10h необходимо для сравнения между собой функций 00h и 10h).

Листинг 1.12. Тестирование функции 10h ввода с клавиатуры (Testlntl6_10h)

```
IDEAL
P386
LOCALS
MODEL MEDIUM

; Подключить файл мнемонических обозначений
; кодов управляющих клавиш и цветовых кодов
include "list1_03.inc"
; Подключить файл макросов
include "list1_04.inc"

SEGMENT sseg para stack 'STACK'
DB 400h DUP(?)
ENDS

DATASEG
; Счетчик операций нажатия/отпускания клавиш
PressCounter DW ?
; Текстовые сообщения
Text DB LIGHTMAGENTA,0,18
     DB "Тестирование функции ввода с клавиатуры 10h",0
     DB YELLOW,2,0,"ASCII-код  Скан-код",0
     DB LIGHTRED,24,29,"Нажмите любую клавишу",0
ENDS

CODESEG
;*****************************
;* Основной модуль программы *
;*****************************
PROC TestInt16_10h
        mov     AX,DGROUP
        mov     DS,AX
        mov     [CS:MainDataSeg],AX
; Установить текстовый режим и очистить экран
        mov     AX,3
        int     10h
; Скрыть курсор - убрать за нижнюю границу экрана
        mov     [ScreenString],25
        mov     [ScreenColumn],0
        call    SetCursorPosition
; Вывести текстовые сообщения на экран
        MShowColorText 3,Text
; Установить белый цвет символов и черный фон
        mov     [TextColorAndBackground],WHITE
; Установить начальную позицию для вывода кодов
; в нулевой колонке пятой строки
        mov     [ScreenString],3
        mov     [ScreenColumn],0
; Инициализировать счетчик операций
; нажатия/отпускания клавиш
        mov     [PressCounter],0
; Принять очередную пару кодов с клавиатуры
@@Next: mov     AH,10h
        int     16h
; Отобразить принятые коды в шестнадцатеричном виде
        ; Отобразить ASCII-код
        mov     [ScreenColumn],3
        call    ShowHexByte
        ; Отобразить скан-код
        mov     [ScreenColumn],14
        mov     AL,AH
        call    ShowHexByte
        ; Перейти на следующую строку
        inc     [ScreenString]
        ; Увеличить значение счетчика нажатий клавиш
        inc     [PressCounter]
        ; После 20 нажатий выйти из цикла
        cmp     [PressCounter],20
        jb      @@Next
; Переустановить текстовый режим и очистить экран
        mov     AX,3
        int     10h
; Выход в DOS
        mov     AH,4Ch
        int     21h
ENDP TestInt16_10h
ENDS
; Подключить процедуры вывода данных на экран
include "list1_02.inc"

END
```

1.2.7.1.6. Программные модули, выполняемые в среде отладчика AFD.

Приведенный ниже листинг представляет пример программного модуля, выполняемого в среде AFD. Программа использует функцию 01h прерывания Int 16h для получения скан-кодов и ASCII-кодов клавиш клавиатуры. Получаемые коды размещаются в буфере памяти, начиная с адреса ES:0000h. Размер буфера 32 байта. После выполнения последней команды Int3 содержимое буфера отображается в окне памяти отладчи-

ка. То есть этот модуль реализует те же действия, что и программа Int16_01h. Если во второй строке заменить 10h на 00h, то будет реализована функция программы Int16_00h, а если поставить значение 20h, то будут обрабатываться дополнительные клавиши 122-клавишной клавиатуры.

```
2528:0100  BF0000          MOV    DI,0000
2528:0103  B410            MOV    AH,10
2528:0105  CD16            INT    16
2528:0107  268825          MOV    ES:[DI],AH
2528:010A  47              INC    DI
2528:010B  268805          MOV    ES:[DI],AL
2528:010E  47              INC    DI
2528:010F  F7C72000        TEST   DI,0020
2528:0113  74EE            JZ     0103
2528:0115  CC              INT3
```

Для получения значений ASCII-кодов знаков русского алфавита необходимо предварительно загрузить резидентную программу-русификатор, который будет перехватывать прерывание Int 09h и конвертировать скан-коды в ASCII коды в соответствие с русифицированной кодовой таблицей. При работе в Windows 95/98/Me в режиме загрузки Command Prompt Only для русификации видеосистемы и клавиатуры необходимо в файл Autoexec.bat внести следующие строчки:

MODE CON CODEPAGE PREPARE=((866) C:\WINDOWS\COMMAND\EGA3.CPI)
MODE CON CODEPAGE SELECT=866
KEYB RU,,C:\WINDOWS\COMMAND\KEYBRD3.SYS

а в файл Config.sys внести строчки:

device=C:\WINDOWS\COMMAND\display.sys con=(ega,,1)
Country=007,866,C:\WINDOWS\COMMAND\country.sys

Этот программный модуль использовался для уточнения значений ASCII-кодов, приведенных в таблицах 1.14 - 1.20.

1.2.7.2. Примеры работы с клавиатурой непосредственно через адаптер клавиатуры

Реальная необходимость в непосредственной работе с клавиатурой возникает в том случае, если вы создаете программу, которая переводит процессор из реального режима в защищенный, а затем выполняет в защищенном режиме всю дальнейшую работу. Перевод в защищенный режим приводит к тому, что функции BIOS, рассчитанные на реальный режим, становятся непригодными для использования. Самопроверка адаптера и программирование основных параметров его работы, а также самотестирование и программирование параметров клавиатуры (и мыши PS/2) производятся при включении компьютера и могут быть изменены с помощью прерываний BIOS, пока процессор еще не переведен из реального режима в защищенный. В защищенном режиме необходимо выполнять только два типа операций: считывание кодов нажимаемых клавиш и зажигание/гашение светодиодов на клавиатуре.

Примеры программ, работающих непосредственно через регистры адаптера клавиатуры, приведены в листингах 1.13, 1.14 и 1.15.

1.2.7.2.1. Программа отображения потока данных от клавиатуры в шестнадцатеричных кодах

Программа KeyboardTest [6] из листинга 1.13 предназначена для отображения на экран потока кодов, поступающих с клавиатуры. Она позволяет наблюдать реакцию клавиатуры на нажатие, удержание и отпускание клавиш различных типов. Процедура обработки клавиатурного прерывания IRQ1 KeyboardInterrupt отображает принятый от клавиатуры код на экран в шестнадцатеричном представлении, а затем производит подготовку к выводу следующего кода (изменяет значение текущих экранных координат ScreenColumn и ScreenSring). Для установки вектора прерывания на эту процедуру служит подпрограмма SetKeyboardInterrupt, а для восстановления старого вектора прерывания после завершения работы основной программы используется подпрограмма Restore0ldKeyboardInterrupt. В приложении 1 приводится краткая информация о подсистеме аппаратных прерываний ПК типа PC/AT, которая позволяет понять текст процедур KeyboardInterrupt, SetKeyboardInterrupt и Restore0ldKeyboardInterrupt.

Листинг 1.13. Отображение потока данных от клавиатуры в шестнадцатеричных кодах

```
IDEAL
P386
LOCALS
MODEL MEDIUM
; Подключить файл мнемонических обозначений
; кодов управляющих клавиш и цветовых кодов
include "list1_03.inc"
; Подключить файл макросов
include "list1_04.inc"

SEGMENT sseg para stack 'STACK'
DB 400h DUP(?)
ENDS
DATASEG
; Счетчик операций нажатия/отпускания клавиш
PressCounter DW ?
```

```
; Область сохранения старого вектора прерывания клавиатуры
OldKeyboardInterruptOffset  DW ?
OldKeyboardInterruptSegment DW ?
; Текстовые сообщения
Text DB LIGHTGREEN,0,19
     DB "Непосредственный прием данных с клавиатуры",0
     DB YELLOW,4,0,"Отображение потока данных "
     DB "в шестнадцатеричных кодах:",0
     DB LIGHTBLUE,24,29,"Нажмите любую клавишу",0
ENDS

CODESEG
;*****************************
;* Основной модуль программы *
;*****************************
PROC KeyboardTest
        mov     AX,DGROUP
        mov     DS,AX
        mov     [CS:MainDataSeg],AX
; Установить текстовый режим и очистить экран
        mov     AX,3
        int     10h
; Скрыть курсор - убрать за нижнюю границу экрана
        mov     [ScreenString],25
        mov     [ScreenColumn],0
        call    SetCursorPosition
; Вывести текстовые сообщения на экран
        MShowColorText 3,Text
; Установить белый цвет символов и черный фон
        mov     [TextColorAndBackground],WHITE
; Установить начальную позицию для вывода кодов
; в нулевой колонке пятой строки
        mov     [ScreenString],5
        mov     [ScreenColumn],0
; Инициализировать счетчик операций
; нажатия/отпускания клавиш
        mov     [PressCounter],0
; Установить новый обработчик прерывания
        call    SetKeyboardInterrupt

; Ожидать, пока не будет произведено 200
; операций нажатия/отпускания
@@Next: cmp     [PressCounter],200
        jb      @@Next
; Восстановить исходный обработчик прерывания
        call    RestoreOldKeyboardInterrupt
; Переустановить текстовый режим и очистить экран
        mov     AX,3
        int     10h
; Передать управление DOS
        mov     AH,4Ch
        int     21h
ENDP KeyboardTest

;*********************************************
;* НОВЫЙ ОБРАБОТЧИК ПРЕРЫВАНИЯ ОТ КЛАВИАТУРЫ *
;*********************************************
proc KeyboardInterrupt far
        pusha
        push    DS
        mov     AX,[CS:MainDataSeg]
        mov     DS,AX
; Получить скан-код
        in      AL,60h
; Разрешить прерывания с более низким приоритетом
        push    AX
        mov     AL,20h  ;команда EOI
        out     20h,AL
        pop     AX
        sti             ;разрешить прерывания
; Отобразить на экране монитора принятый от клавиатуры
; код в шестнадцатеричном представлении
        ; Отобразить принятый байт
        call    ShowHexByte
        ; Перевести текущую позицию на 2 символа влево
        ; (подготовка для вывода следующего кода)
        add     [ScreenColumn],2
        ; Проверить пересечение правой границы экрана
        cmp     [ScreenColumn],80
        jb      @@EndShowCode
        ; Если достигнута правая граница
        ; экрана - перейти на следующую строку
        sub     [ScreenColumn],80
        inc     [ScreenString]
@@EndShowCode:
; Завершение обработки прерывания
        ; Увеличить значение счетчика нажатий/отпусканий
        inc     [PressCounter]
        pop     DS
        popa
        iret
endp KeyboardInterrupt

;*************************************************
;* УСТАНОВИТЬ НОВЫЙ ОБРАБОТЧИК ПРЕРЫВАНИЯ КЛАВИАТУРЫ *
;*************************************************
PROC SetKeyboardInterrupt NEAR
        pusha
        push    ES
        mov     AX,0
        mov     ES,AX
; Запомнить прежний вектор обработчика
; прерывания клавиатуры
        mov     AX,[ES:9*4]
        mov     [OldKeyboardInterruptOffset],AX
        mov     AX,[ES:9*4+2]
        mov     [OldKeyboardInterruptSegment],AX
; Установка вектора прерывания на обработчик клавиатуры
        cli             ;запретить прерывания
        mov     AX,offset KeyboardInterrupt
        mov     [ES:9*4],AX
        mov     AX,CS
        mov     [ES:9*4+2],AX
        sti             ;разрешить прерывания
        pop     ES
        popa
        ret
ENDP SetKeyboardInterrupt
;*****************************************
;* ВОССТАНОВИТЬ ИСХОДНЫЙ ВЕКТОР ПРЕРЫВАНИЯ *
;*****************************************
PROC RestoreOldKeyboardInterrupt NEAR
        pusha
; Настроить регистр ES на таблицу векторов прерываний
        push    ES
        mov     AX,0
        mov     ES,AX
; Восстановить прежний вектор обработчика прерывания
        cli
        mov     AX,[OldKeyboardInterruptOffset]
        mov     [ES:9*4],AX
        mov     AX,[OldKeyboardInterruptSegment]
        mov     [ES:9*4+2],AX
        sti
        pop     ES
        popa
        ret
ENDP RestoreOldKeyboardInterrupt
ENDS

; Подключить процедуры вывода данных на экран
include "list1_02.inc"

END
```

1.2.7.2.2. Программа упрощенного драйвера клавиатуры.

Программа KeyboardDriver [6] из листинга 1.14 демонстрирует выполнение следующих типовых операций над данными, поступающими от адаптера клавиатуры системной платы:

- преобразование скан-кодов в ASCII-коды;
- обработка управляющих клавиш;
- управление светодиодами клавиатуры.

Кроме процедур общего назначения, в программе KeyboardDriver используются следующие подпрограммы:

- процедура KeyboardInterrupt принимает данные от клавиатуры, обрабатывает их и при необходимости запоминает введенный символ в выделенной для него ячейке или переключает состояние светодиодов на клавиатуре;
- процедура SetKeyboardInterrupt служит для установки вектора прерывания от клавиатуры на подпрограмму KeyboardInterrupt;
- процедура Restore0ldKeyboardInterrupt служит для восстановления прежнего вектора прерывания после завершения работы программы;
- процедура Wait8042BufferEmpty выполняет цикл ожидания освобождения буфера адаптера клавиатуры.

Эти процедуры взаимодействуют с подсистемой аппаратных прерываний ПК. Пояснения правил работы с аппаратными прерываниями приведено в Приложении 1.

Листинг 1.14. Упрощенный вариант драйвера клавиатуры

```
IDEAL
P386
LOCALS
MODEL MEDIUM

; Подключить файл мнемонических обозначений
; кодов управляющих клавиш и цветовых кодов
include "list1_03.inc"
; Подключить файл макросов
include "list1_04.inc"

SEGMENT sseg para stack 'STACK'
DB 400h DUP(?)
ENDS

DATASEG
; Счетчик операций нажатия/отпускания клавиш
PressCounter DW ?
; Область сохранения старого вектора
; прерывания клавиатуры
OldKeyboardInterruptOffset  DW ?
OldKeyboardInterruptSegment DW ?
; Байт состояния клавиатуры (бит 0 - ScrollLock,
; бит 1 - NumLock, бит 2 - CapsLock, бит 3 - левый
; Shift, бит 4 - правый Shift)
KeyboardStatus DB 0
; Байт состояния светодиодов (бит 0 - ScrollLock,
; бит 1 - NumLock, бит 2 - CapsLock)
LEDStatus DB 0
; Признак ввода очередного символа
CharInputFlag DB 0
; ASCII-код введенного символа
ASCIICode DB 0
; Текстовые сообщения
Text DB LIGHTGREEN,0,21
     DB "Упрощенный вариант драйвера клавиатуры",0
     DB YELLOW,10,0
     DB "Отображение потока данных в ASCII-кодах:",0
     DB LIGHTGREY,24,29,"Нажмите любую клавишу",0
; Перекодировочные таблицы (для алфавитно-цифровой
; клавиатуры)
label RusNorm byte
  DB  0 , 27,'1','2','3','4','5','6'
  DB '7','8','9','0','-','=', 8 , 9
  DB 'й','ц','у','к','е','н','г'
  DB 'ш','щ','з','х','ъ', 13, 0
  DB 'ф','ы','в','а','п','р','о'
  DB 'л','д','ж','э','ё', 0 ,'\'
  DB 'я','ч','с','м','и','т','ь'
  DB 'б','ю','.', 0 ,'*', 0 ,' '
label RusNormShift byte
  DB  0 , 27,'!','"','№',';','%',':'
  DB '?','*','(',')','_','+', 8 ,
  DB 'Й','Ц','У','К','Е','Н','Г'
  DB 'Ш','Щ','З','Х','Ъ', 13, 0
  DB 'Ф','Ы','В','А','П','Р','О'
  DB 'Л','Д','Ж','Э','Ё', 0 ,'/'
  DB 'Я','Ч','С','М','И','Т','Ь'
  DB 'Б','Ю',',', 0 ,'*', 0 ,' '
label RusCaps byte
  DB  0 , 27,'1','2','3','4','5','
  DB '7','8','9','0','-','=', 8 ,
  DB 'Й','Ц','У','К','Е','Н','Г'
  DB 'Ш','Щ','З','Х','Ъ', 13, 0
  DB 'Ф','Ы','В','А','П','Р','О'
  DB 'Л','Д','Ж','Э','Ё', 0 ,'\'
  DB 'Я','Ч','С','М','И','Т','Ь'
  DB 'Б','Ю',',', 0 ,'*', 0 ,' '
label RusCapsShift byte
  DB  0 , 27,'!','"','№',';','%',':'
  DB '?','*','(',')','_','+', 8 , 9
  DB 'й','ц','у','к','е','н','г'
  DB 'ш','щ','з','х','ъ', 13, 0
  DB 'ф','ы','в','а','п','р','о'
  DB 'л','д','ж','э','ё', 0 ,'/'
  DB 'я','ч','с','м','и','т','ь'
  DB 'б','ю',',', 0 ,'*', 0 ,' '
ENDS

CODESEG
;****************************
;* Основной модуль программы *
;****************************
PROC KeyboardDriver
        mov     AX,DGROUP
        mov     DS,AX
        mov     [CS:MainDataSeg],AX
; Установить текстовый режим и очистить экран
        mov     AX,3
        int     10h
; Вывести текстовые сообщения на экран
        MShowColorText 3,Text
; Инициализировать счетчик операций
; нажатия/отпускания клавиш
        mov     [PressCounter],0
```

```
; Установить новый обработчик прерывания
        call    SetKeyboardInterrupt
; Установить курсор в начало строки
        mov     [ScreenString],12
        mov     [ScreenColumn],0
        call    SetCursorPosition
; Настроить ES:DI на видеопамять
        mov     AX,0B800h ;текстовый видеосегмент
        mov     ES,AX
        mov     AH,WHITE  ;белый цвет, черный фон
        ; Выполнять вывод символов в двенадцатой строке
        mov     DI,12*160
; Ввести и отобразить на экране 80 символов
@@NextChar:
        cmp     [CharInputFlag],0 ;Введен символ?
        je      @@NextChar
        ; Сбросить флаг ввода символа
        mov     [CharInputFlag],0
        ; Отобразить символ
        mov     AL,[ASCIICode]
        stosw
        ; Передвинуть курсор в следующую позицию
        inc     [ScreenColumn]
        call    SetCursorPosition
        ; Увеличить счетчик
        inc     [PressCounter]
        ; Проверить условие завершения цикла
        cmp     [PressCounter],80
        jb      @@NextChar
; Восстановить исходный обработчик прерывания
        call    RestoreOldKeyboardInterrupt
; Переустановить текстовый режим и очистить экран
        mov     AX,3
        int     10h
; Передать управление DOS
        mov     AH,4Ch
        int     21h
ENDP KeyboardDriver

;*****************************************
;* НОВЫЙ ОБРАБОТЧИК ПРЕРЫВАНИЯ ОТ КЛАВИАТУРЫ *
;*****************************************
proc KeyboardInterrupt far
;Сохранить регистры
        pusha
        push    DS
        mov     AX,[CS:MainDataSeg]
        mov     DS,AX
; Получить скан-код
        in      AL,60h
        ;Разрешить прерывания
        push    AX          ;сохранить скан-код
        mov     AL,20h      ;послать команду EOI
        out     20h,AL      ;в ведущий контроллер
        pop     AX          ;восстановить скан-код
        sti

; Игнорировать код подтверждения команды ACK
        cmp     AL,0FAh     ;код подтверждения команды?
        je      @@End
; Обработать Shift
        cmp     AL,2Ah      ;левый Shift нажат
        je      @@LeftShiftON
        cmp     AL,0AAh     ;левый Shift отпущен
        je      @@LeftShiftOFF
        cmp     AL,36h      ;правый Shift нажат
        je      @@RightShiftON
        cmp     AL,0B6h     ;правый Shift отпущен
        je      @@RightShiftOFF
; Проверить нажатие клавиш статуса
        cmp     AL,3Ah      ;нажатие клавиши CapsLock?
        je      @@CapsLockLED_ON
        cmp     AL,45h      ;нажатие клавиши NumLock?
        je      @@NumLockLED_ON
        cmp     AL,46h      ;нажатие клавиши ScrollLock?
        je      @@ScrollLockLED_ON
; Обработать алфавитно-цифровые клавиши
        cmp     AL,39h
        ja      @@End
        ; Использовать скан-код как индекс
        ; элемента в массиве перекодировки
        xor     BX,BX
        mov     BL,AL
        ; Выбрать массив перекодировки
        test    [KeyboardStatus],100b ;CapsLock активен?
        jnz     @@CapsON
        test    [KeyboardStatus],11000b ;Shift нажат?
        jnz     @@ShiftON1
        add     BX,offset RusNorm
        jmp short @@SaveChar
@@ShiftON1:
        add     BX,offset RusNormShift
        jmp short @@SaveChar
@@CapsON:
        test    [KeyboardStatus],11000b ;Shift нажат?
        jnz     @@ShiftON2
        add     BX,offset RusCaps
        jmp short @@SaveChar
@@ShiftON2:
        add     BX,offset RusCapsShift
@@SaveChar:
; Сохранить ASCII-код символа и установить флаг приема
        mov     AL,[BX]
        cmp     AL,32   ;это алфавитно-цифровой символ?
        jb      @@End
        mov     [ASCIICode],AL
        mov     [CharInputFlag],1
        jmp short @@End

; Установить признак Shift
@@LeftShiftON:
        or      [KeyboardStatus],1000b
        jmp short @@End
@@LeftShiftOFF:
        and     [KeyboardStatus],11110111b
        jmp short @@End
@@RightShiftON:
        or      [KeyboardStatus],10000b
        jmp short @@End
@@RightShiftOFF:
        and     [KeyboardStatus],11101111b
        jmp short @@End
; При нажатии клавиши статуса зажечь или погасить
; соответствующий светодиод
@@CapsLockLED_ON:
        xor     [LEDStatus],100b
        jmp short @@SetKeyboardLED
@@NumLockLED_ON:
        xor      [LEDStatus],10b
        jmp short @@SetKeyboardLED
@@ScrollLockLED_ON:
        xor     [LEDStatus],1
@@SetKeyboardLED:
        ; Сбросить незначащие разряды
        and     [LEDStatus],111b
        ; Скопировать LEDStatus в KeyboardStatus
        and     [KeyboardStatus],11111000b
        mov     AL,[LEDStatus]
        or      [KeyboardStatus],AL
; Переустановить состояние светодиодов на клавиатуре
        call    Wait8042BufferEmpty
        ; Послать команду установки светодиодов
        mov     AL,0EDh
        out     60h,AL
        call    Wait8042BufferEmpty
        ; Передать байт состояния светодиодов
        mov     AL,[LEDStatus]
        out     60h,AL
@@End:  pop     DS
        popa
```

```
        iret
endp KeyboardInterrupt

;*************************************************

;* УСТАНОВИТЬ НОВЫЙ ОБРАБОТЧИК ПРЕРЫВАНИЯ КЛАВИАТУРЫ *
;*************************************************
PROC SetKeyboardInterrupt NEAR
        pusha
        push    ES
        mov     AX,0
        mov     ES,AX
; Запомнить прежний вектор обработчика
; прерывания клавиатуры
        mov     AX,[ES:9*4]
        mov     [OldKeyboardInterruptOffset],AX
        mov     AX,[ES:9*4+2]
        mov     [OldKeyboardInterruptSegment],AX
; Установка вектора прерывания на обработчик клавиатуры
        cli                 ;запретить прерывания
        mov     AX,offset KeyboardInterrupt
        mov     [ES:9*4],AX
        mov     AX,CS
        mov     [ES:9*4+2],AX
        sti                 ;разрешить прерывания
        pop     ES
        popa
        ret
ENDP SetKeyboardInterrupt

;*****************************************
;* ВОССТАНОВИТЬ ИСХОДНЫЙ ВЕКТОР ПРЕРЫВАНИЯ *
;*****************************************
PROC RestoreOldKeyboardInterrupt NEAR
        pusha
; Настроить регистр ES на таблицу векторов прерываний
        push    ES
        mov     AX,0
        mov     ES,AX
; Восстановить прежний вектор обработчика прерывания
        cli
        mov     AX,[OldKeyboardInterruptOffset]
        mov     [ES:9*4],AX
        mov     AX,[OldKeyboardInterruptSegment]
        mov     [ES:9*4+2],AX
        sti
        pop     ES
        popa
        ret
ENDP RestoreOldKeyboardInterrupt

;*********************************************
;*   ОЖИДАНИЕ ОЧИСТКИ ВХОДНОГО БУФЕРА I8042    *
;* При выходе из процедуры:                  *
;* флаг ZF установлен - нормальное завершение, *
;* флаг ZF сброшен - ошибка тайм-аута.       *
;*********************************************
proc Wait8042BufferEmpty near
        push    CX
        mov     CX,0FFFFh  ;задать число циклов
@@kb:   in      AL,64h     ;получить статус
        test    AL,10b     ;буфер i8042 свободен?
        loopnz  @@kb       ;если нет, то цикл
        pop     CX
        ; (если при выходе сброшен флаг ZF - ошибка)
        ret
endp Wait8042BufferEmpty
ENDS

; Подключить процедуры вывода данных на экран
include "list1_02.inc"

END
```

Следует иметь в виду, что приведенный пример был для наглядности сильно упрощен: драйвер обрабатывает только коды клавиш основной клавиатуры и управляющих клавиш Mum Lock и Scroll Lock. Функциональные клавиши, клавиши дополнительной и цифровой клавиатур игнорируются. Из трех клавиш, состояние которых отображается светодиодами, влияние на коды вводимых символов оказывает только Caps Lock, а реакция на Mum Lock и Scroll Lock сводится к зажиганию или гашению соответствующих индикаторов.

1.2.7.2.3. Программа чтения скан-кодов, сориентированная на выполнение в среде отладчика AFD

Программа, читающая скан-коды из выходного буфера адаптера клавиатуры (адрес 60h) и помещающая их в буфер, начинающийся с адреса DS:0000 и имеющий размер в 16 байт. На время работы программы запрещается обработка аппаратных прерываний.

Программа использует:

- подпрограмму P3, разрешающую или запрещающую конвертирование скан-кодов в адаптере;
- подпрограмму P4, реализующую переключение режимов работы клавиатур
- подпрограмму P2, реализующую ожидание заполнения выходного буфера адаптера

Листинг 1.15. Программа чтения скан-кодов клавиатуры, работающей в режиме 1, 2 или 3

```
232E:016C  FA        CLI            ;Запрет маскируемых прерываний.
232E:016D  B301      MOV    BL,01   ;Запрет преобразования скан-кодов
232E:016F  B7BF      MOV    BH,BF   ;адаптером (бит 6 командн. байта
232E:0171  E8A4FF    CALL   0118    ;уст. в 0) (выполняет П/П P3).
232E:0174  B303      MOV    BL,03   ;Переключение клавиатуры в режим 3
232E:0176  E8D1FF    CALL   014A    ;подпрограммой (П/П) P4.
232E:0179  BF0000    MOV    DI,0000 ;Установка на начало буфера памяти.
232E:017C  8CD8      MOV    AX,DS   ;Содержимое сегментного регистра DS
232E:017E  8EC0      MOV    ES,AX   ;переписывается в регистр ES.
232E:0180  E885FF    CALL   0108    ;Поступил скан-код в вых. буфер?
232E:0183  E460      IN     AL,[60] ;Чтение скан-кода из вых. буфера
232E:0185  268805    MOV    ES:[DI],AL    ;и запись его в буф. памяти.
```

```
232E:0188  47        INC    DI      ;Установка на след. байт буфера.
232E:0189  F7C71000  TEST   DI,0010 ;Буфер памяти заполнен?
232E:018D  74F1      JZ     0180    ;если нет - чтение следующего кода.
232E:018F  B302      MOV    BL,02   ;Переключение клавиатуры в реж.2
232E:0191  E8B6FF    CALL   014A    ;подпрограммой P4.
232E:0194  B302      MOV    BL,02   ;Разрешение преобразования скан-
232E:0196  B740      MOV    BH,40   ;кодов в системные подпрограммой P3.
232E:0198  E87DFF    CALL   0118
232E:019B  FB        STI            ;Разрешение маскируемых прерываний.
232E:019C  CC        INT3           ;Передача управления отладчику AFD.
```

Подпрограмма P1

Ожидание очистки входного буфера адаптера клавиатуры.

```
232E:0100  E464      IN     AL,[64] ;Чтение регистра состояния.
232E:0102  A802      TEST   AL,02   ;Разряд 1 равен 0?
232E:0104  75FA      JNZ    0100    ;если нет, то новый цикл чтения,
232E:0106  C3        RET            ;если да, то возврат из П/П.
232E:0107  90        NOP
```

Подпрограмма P2

Ожидание заполнения выходного буфера адаптера клавиатуры.

```
232E:0108  E464      IN     AL,[64] ;Чтение регистра состояния.
232E:010A  A801      TEST   AL,01   ;Разряд 0 равен 0?
232E:010C  74FA      JZ     0108    ;если да, то новый цикл чтения,
232E:010E  C3        RET            ;если нет, то возврат из П/П.
```

Подпрограмма P21

Проверка положительного кода подтверждения клавиатуры FAh (ACK).

```
232E:010F  E460      IN     AL,[60] ;Чтение кода ответа клавиатуры.
232E:0111  3CFA      CMP    AL,FA   ;Сравнение с кодом FAh.
232E:0113  7501      JNZ    0116    ;Коды совпали?
232E:0115  C3        RET            ;если совпали - возврат из П/П,
232E:0116  CC        INT3           ;если не совпали - выход в AFD.
232E:0117  90        NOP
```

Подпрограмма P3

Изменение командного байта адаптера клавиатуры.

Перед вызовом подпрограммы необходимо задать входные параметры:
В младший байт регистра BX (BL) необходимо занести:

- 01h - если с текущим командным байтом выполняется логическая операция AND;
- 02h - логическая операция OR.

В старший байт регистра BX (BH) необходимо занести:

- для операции AND - байт с нулями в разрядах, которые необходимо обнулить в командном байте;
- для операции OR - ,байт с единицами в разрядах, которые необходимо установить в командном байте.

В подпрограмме используются вспомогательные подпрограммы:

- подпрограмма P1, реализующая процедуру ожидания очистки входного буфера адаптера;
- подпрограмма P2, реализующая процедуру ожидания заполнения выходного буфера адаптера.

```
232E:0118  E8E5FF    CALL   0100    ;Ожидание очистки вх. буфера.
232E:011B  B020      MOV    AL,20   ;Команда чтения командного байта (20h)
232E:011D  E664      OUT    [64],AL ;передается в адаптер.
232E:011F  E8E6FF    CALL   0108    ;Ожидание заполнения вых. буфера.
232E:0122  E460      IN     AL,[60] ;Чтение командного байта.
232E:0124  80FB01    CMP    BL,01   ;Операция "END"?
```

```
232E:0127  7407      JZ     0130    ;если "да", то переход на ее выполнение.
232E:0129  80FB02    CMP    BL,02   ;Операция "OR"?
232E:012C  7408      JZ     0136    ;если "да", то переход на ее выполнение,
232E:012E  EB18      JMP    0148    ;если "нет", то выход из П/П.
232E:0130  20F8      AND    AL,BH   ;Выполнение операции "END" и занесение
232E:0132  88C3      MOV    BL,AL   ;результата в регистр BL.
232E:0134  EB04      JMP    013A    ;Переход на запись нового командн. байта.
232E:0136  08F8      OR     AL,BH   ;Выполнение операции "OR" и занесение
232E:0138  88C3      MOV    BL,AL   ;результата в регистр BL.
232E:013A  B060      MOV    AL,60   ;Команда 60h записи командного байта
232E:013C  E664      OUT    [64],AL ;передается в адаптер через порт 64h.
232E:013E  E8BFFF    CALL   0100    ;Ожидание очистки входного буфера.
232E:0141  88D8      MOV    AL,BL   ;Значение нового командного байта
232E:0143  E660      OUT    [60],AL ;передается в адаптер через порт 60h.
232E:0145  E8B8FF    CALL   0100    ;Ожидание очистки входного буфера.
232E:0148  C3        RET            ;Выход из подпрограммы.
232E:0149  90        NOP
```

Подпрограмма P4

Переключения режимов работы клавиатуры.

Перед вызовом подпрограммы необходимо задать входные параметры:
в регистр BL необходимо занести константу:

- 00h - чтение текущего режима работы клавиатуры;
- 01h - установить режим 1 (PC/XT);
- 02h - установить режим 2 (PC/AT-совместимые ПК);
- 03h - установить режим 3 (ПК типа PS/2).

При BL=00h подпрограмма возвращает в регистре AL значение текущего режима работы клавиатуры.

В подпрограмме используются вспомогательные подпрограммы:

- подпрограмма P1, реализующая процедуру ожидания очистки входного буфера адаптера;
- подпрограмма P2, реализующая процедуру ожидания заполнения выходного буфера адаптера;
- подпрограмма P21, реализующая процедуру проверки значения кода подтверждения клавиатуры FAh (ACK).

```
232E:014A  E8B3FF    CALL   0100    ;Ожидание очистки входного буфера.
232E:014D  B0F0      MOV    AL,F0   ;Команда F0h установки режима клавиатуры
232E:014F  E660      OUT    [60],AL ;передается в клавиатуру через порт 60h.
232E:0151  E8B4FF    CALL   0108    ;Ожидание поступления кода в вых. буфер.
232E:0154  E8B8FF    CALL   010F    ;Чтение вых. буфера и сравнение с FAh.
232E:0157  88D8      MOV    AL,BL   ;Передача нового кода режима в клавиатуру
232E:0159  E660      OUT    [60],AL ;через порт 60h адаптера.
232E:015B  E8AAFF    CALL   0108    ;Ожидание поступления кода в вых. буфер.
232E:015E  E8AEFF    CALL   010F    ;Чтение вых. буфера и сравнение с FAh.
232E:0161  80FB00    CMP    BL,00   ;Код режима равен 0?
232E:0164  7505      JNZ    016B    ;если нет, то возврат из подпрограммы,
232E:0166  E89FFF    CALL   0108    ;если да, то чтение кода текущего режима
232E:0169  E460      IN     AL,[60] ;работы клавиатуры из вых. буфера.
232E:016B  C3        RET            ;Возврат из подпрограммы.
```

Программа запускается командой G 16C из командной строки отладчика AFD. Предварительно необходимо загрузить коды команд программы и подпрограмм по указанным в листинге адресам. В этих адресах содержимое регистра кодового сегмента (CS) определяет сам отладчик AFD при его запуске, а значения смещения должны быть такими, как указано в листинге 1.15.

Выше приведенная программа была использована для определения значений скан-кодов, представленных в таблицах 1.4 - 1.9.

В данном разделе использовались материалы из [4 - 6, 8, 9, 11, 13 - 15, 18, 19].

Контрольные вопросы

1. Охарактеризуйте клавиатуру и принципы ее взаимодействия с системой.
2. Охарактеризуйте три основных типа клавиатур ПК семейства PC.
3. Охарактеризуйте возможности 101/102-клавишной клавиатуры и ее развитие.
4. Охарактеризуйте основные группы клавиш 101/102-клавишной клавиатуры.
5. Что входит в состав модуля управления клавиатурой?
6. Охарактеризуйте конструктивные компоненты 101/102-клавишной клавиатуры.
7. Дайте описание блок-схемы клавиатуры (Рис. 1.14).
8. Какие типы датчиков клавиш могут применяться в клавиатурах?
9. Каковы функции устройства управления клавиатурой (контроллера клавиатуры)?
10. Какими типами данных обмениваются клавиатура и система?
11. Охарактеризуйте буфер клавиатуры, организованный в устройстве управления клавиатурой (в контроллере клавиатуры).
12. Как устройство управления клавиатурой (контроллер) обрабатывает состояние клавиш?
13. Дайте краткую характеристику компонентам электрической схемы 101-клавишной клавиатуры фирмы Xring.
14. Охарактеризуйте контрольные точки электрической схемы клавиатуры.
15. Какой тип клавиатуры может принимать команды от системы?
16. Можно ли подключить USB-клавиатуру к разъему PS/2-клавиатуры?
17. Кратко охарактеризуйте интерфейс 101-клавишной клавиатуры.
18. Кратко опишите взаимодействие 101-клавишной клавиатуры с системой.
19. Как взаимодействовала XT-клавиатура с системой?
20. Как выполняется базовый тест клавиатуры (BAT)?
21. Охарактеризуйте использование линий синхронизации и данных интерфейса клавиатуры.
22. Кратко охарактеризуйте потоки данных между клавиатурой и системой.
23. Опишите особенности ввода данных с клавиатуры.
24. Опишите особенности передачи данных в клавиатуру.
25. Охарактеризуйте осциллограммы, представленные на рис. 1.19.
26. Охарактеризуйте программу, представленную в листинге KLEE1.TXT.
27. Охарактеризуйте режим 1 101-клавишной клавиатуры.
28. Охарактеризуйте режим 2 101-клавишной клавиатуры.
29. Охарактеризуйте режим 3 101-клавишной клавиатуры.
30. Охарактеризуйте скан-коды 101-клавишной клавиатуры.
31. Скан-коды каких клавиш добавлены в 122-клавишной клавиатуре?
32. В чем заключается программирование клавиатуры PC/XT?
33. В чем заключается программирование 101-клавишной клавиатуры?
34. Как системой обрабатывается нажатие и отпускание клавиш?
35. Как обрабатываются специальные клавиши?
36. Охарактеризуйте особенности взаимодействия с клавиатурой PC/XT через порты 60h и 61h.
37. Охарактеризуйте особенности взаимодействия с 101-клавишной клавиатурой через порты 60h и 64h.
38. Какие правила необходимо соблюдать при программирование 101-клавишной клавиатуры на уровне ее команд?
39. Перечислите команды, поддерживаемые клавиатурой.
40. Охарактеризуйте команду клавиатуры EDh.
41. Охарактеризуйте команду клавиатуры EEh и F2h.
42. Охарактеризуйте команду клавиатуры F4h.
43. Охарактеризуйте коды ответов клавиатуры.
44. Где расположен буфер клавиатуры обработчика прерывания INT 09h?
45. Как организован буфер клавиатуры в области переменных BIOS?
46. Охарактеризуйте назначение флаговых разрядов, устанавливаемых обработчиком INT 9h в ячейках памяти 0040:0017h и 0040:0018h.
47. Для чего используется перехват прерывания INT 9h?
48. Дайте общую характеристику программному прерыванию INT 16h.
49. Охарактеризуйте функции 00h, 10h и 20h прерывания INT 16h.
50. Охарактеризуйте функции 01h, 11h и 21h прерывания INT 16h.
51. Охарактеризуйте функции 02h, 12h и 22h прерывания INT 16h.

52. Охарактеризуйте функции 03h прерывания INT 16h.
53. Охарактеризуйте функции 05h прерывания INT 16h.
54. Охарактеризуйте ASCII-коды основной клавиатуры.
55. Охарактеризуйте ASCII-коды цифровой клавиатуры.
56. Охарактеризуйте ASCII-коды функциональных клавиш.
57. Охарактеризуйте ASCII-коды Alt-комбинаций символьных клавиш.
58. Охарактеризуйте ASCII-коды дополнительной клавиатуры.
59. Охарактеризуйте расширенные ASCII-коды цифровой клавиатуры.
60. Охарактеризуйте расширенные ASCII-коды комбинаций клавиш.
61. Охарактеризуйте подфункции обработки клавиатуры прерыванием INT 21h DOS.
62. Охарактеризуйте программу, представленную в листинге 1.8.
63. Охарактеризуйте программу, представленную в листинге 1.9.
64. Что выполняет процедура GetChar входящая в состав модулей, представленных в листинге 1.9?
65. Охарактеризуйте информацию, представленную в листинге 1.10.
66. Охарактеризуйте макросы, представленные в листинге 1.11.
67. Охарактеризуйте программу, представленную в листинге 1.12.
68. Охарактеризуйте программный модуль, представленный в разделе 1.2.7.1.5.
69. Когда возникает реальная необходимость в непосредственной работе с клавиатурой через порты 60h и 64h?
70. Охарактеризуйте программу, представленную в листинге 1.13.
71. Охарактеризуйте программу, представленную в листинге 1.14.
72. Охарактеризуйте программу, представленную в разделе 1.2.7.2.3.

1.3. PS/2 MOUSE.

Манипулятор PS/2 Mouse появился с компьютерами PS/2 и относится к указательным устройствам (pointing devices). С их помощью пользователь может задать компьютеру местоположение курсора и подать одну из нескольких команд. Для этого манипулятор передает в систему информацию о своем перемещении по плоской поверхности и нажатии двух - трех или более кнопок. PS/2 Mouse подключается к системе через двунаправленный последовательный интерфейс, аналогичный интерфейсу клавиатуры и обслуживается, как правило, тем же адаптером. (Более полная общая информация о манипуляторах мышь представлена в разделе 2).

Устройства-указатели с интерфейсом PS/2 (мышь PS/2) имеют *поддержку BIOS,* обеспечивающую настройку мыши на различные режимы работы посылкой ей соответствующих команд. Поддержка мыши вызывается через BIOS программным прерыванием Int 15h с указанием кода функции (C200-C209h) в регистре AX. Собственно драйвер мыши (обработчик прерывания по номеру вектора 74h от запроса IRQ12), обрабатывающий ее информационные посылки, входит лишь в состав ОС или загружается отдельно.

1.3.1. Конструктивы PS/2 Mouse.

Манипуляторы PS/2 Mouse выпускаются в двух основных вариантах:

- манипуляторы, использующие механические компоненты в блоке преобразования перемещения мыши в электрические сигналы (мышь обычной конструкции);
- манипуляторы, использующие электронно-оптические (видеокамеру с процессором и устройством подсветки) компоненты в блоке преобразования перемещения в электрический сигнал (оптическая мышь).

Общий вид обычной (не оптической) 3D PS/2 Mouse представлен на рис. 1.20, а ее внутренняя конструкция - на рис. 1.21.

Обычная конструкция PS/2-мыши имеет свободно вращающийся массивный обрезиненный шарик (1) в днище корпуса, передающий вращение на два координатных диска (2 и 3) с фотоэлектрическими датчиками. Датчики для каждой координаты представляют собой две открытые оптопары (светодиод (4 и 5) и сдвоенные фотодиоды (6 и 7)), в оптический канал которых входит вращающийся диск с прорезями (2 или 3).

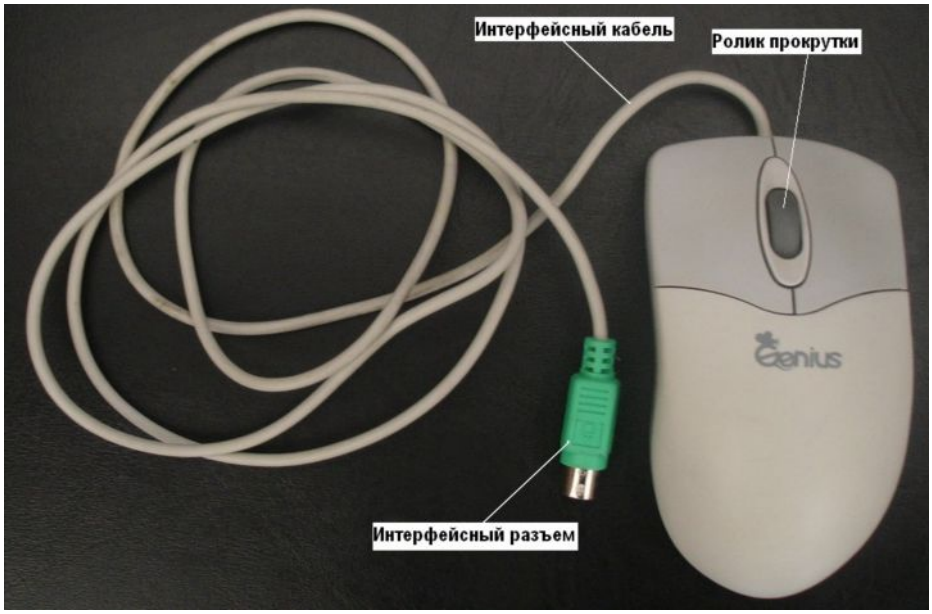

Рис. 1.20. Общий вид 3D PS/2 Mouse.

В оптическом канале каждого датчика два фотодиода оптопары геометрически расположены таким образом, что генерируют фотоэлектрические сигналы сдвинутые по фазе примерно на 90 градусов. Знак сдвига фаз и позволяет определить направление перемещения по координате. Оптопары датчиков могут оформляться в виде монолитных конструкций или быть просто отдельными элементами, установленными на печатной плате. Помимо датчиков перемещения в конструкции имеются два кнопочных датчика (8 и 9), срабатывающих при нажатие и отпускание соответствующей клавиши мыши.

Особенностью 3D мыши (трех координатная мышь) является возможность имитировать перемещение по третьей координате с помощью дополнительного ролика (10), вращаемого пальцем (ролика прокрутки). Этот ролик, как правило, приводит в движение трещотки с контактными датчиками или имеет прорези, действующие в оптическом канале дополнительной двойной оптопары (светодиод 11 и сдвоенный фотодиод 12). С блоком ролика прокрутки, как правило, совмещен третий кнопочный датчик (13), срабатывающий при нажатие и отпускание ролика.

В мыши имеется микроконтроллер (14), который обрабатывает сигналы с датчиков и посылает в ПК

информацию о перемещениях и состоянии кнопок по интерфейсному кабелю (15). Интерфейсный кабель с одной стороны подключается к разъему (16) мыши, а с другой - к интерфейсному разъему USB или через переходник к разъему PS/2 Mouse, расположенному с тыльной стороны системного блока ПК.

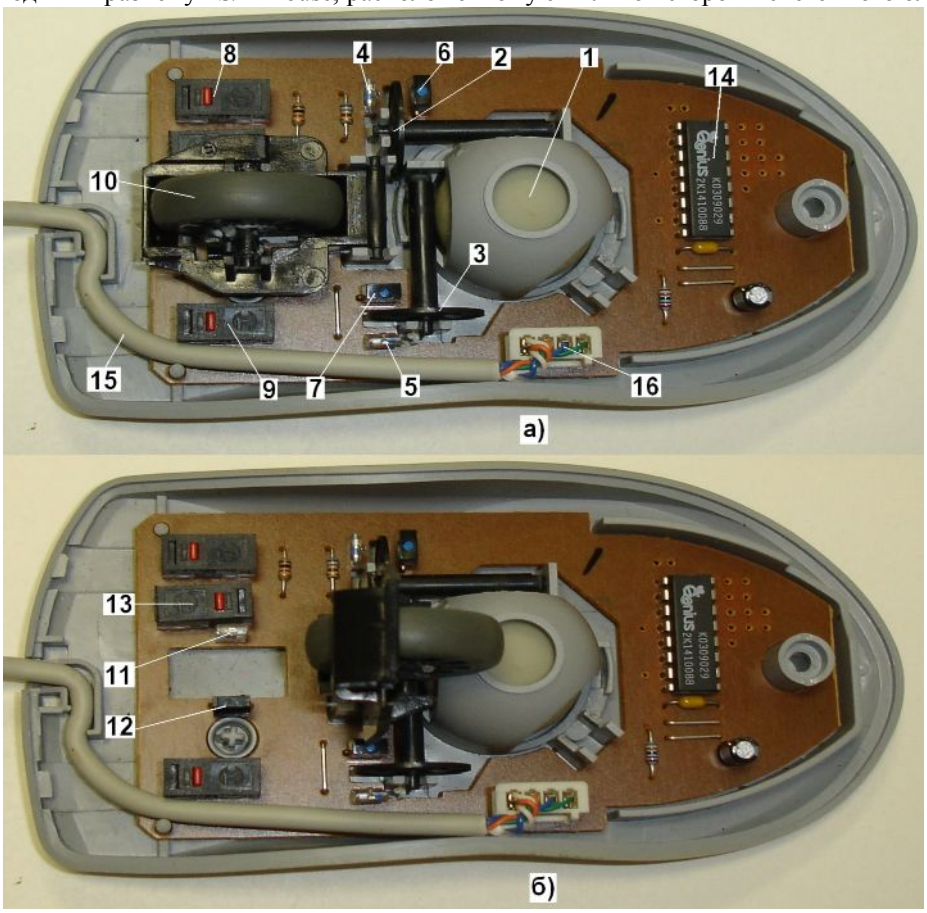

Рис. 1.21. Внутренняя конструкция обычной 3D PS/2 Mouse: а) - ролик прокрутки (10) находится в рабочем состояние; б) - ролик прокрутки приподнят.

На рис. 1.22 приведен вид внутренней конструкции оптической 3D USB/PS/2 мыши, а на рис. 1.23 представлены основные ее компоненты. Оптическая мышь (optical mouse), не имеет механических частей, подверженных загрязнению и износу. Первые модели оптических мышей ориентировались по лучам, отраженным от специального коврика с сетчатым рисунком. Теоретически это надежнее, но загрязнения и царапины на коврике приводят к неожиданным «прыжкам», наклон оси мыши относительно оси коврика сильно искажает отображение траектории движения. Современные оптические мыши имеют встроенную видеокамеру с процессором (1), обрабатывающим полученное изображение, осветительный светодиод 2 и систему оптических призм и линз (3). Эта мышь не требует специального коврика и может функционировать на любой поверхности — оптические «неровности», за которые она может «зацепиться взглядом», есть практически везде. Первые оптические мыши из-за слабости графического процессора обладали низким быстродействием — слишком быстрые перемещения сбивали их с толку. Современные оптические мыши имеют хорошие параметры: разрешение — 800 dpi, скорость — до 1 м/с, ускорение — до 10g.

Как и в обычной 3D PS/2 мыши в оптической 3D USB/PS/2 мыши помимо оптического устройства слежения за ее перемещением имеется:

- два кнопочных датчика (4 и 5), срабатывающих при нажатие и отпускание соответствующих клавиш мыши; -
- ролик прокрутки (6) с прорезями, функционирующий совместно с двойной открытой оптопарой (светодиод (7) и сдвоенный фотодиод (8)) и третьим кнопочным датчиком (9);
- микроконтроллер (10), обрабатывающий сигналы от кнопочных датчиков и от процессора видеокамеры, а так же осуществляющий взаимодействие с системой через интерфейсный кабель (11).

Оптическая система мыши состоит из двух частей:

- осветительного светодиода (2) с системой оптических призм и линз (3), расположенных на электронном блоке мыши (рис. 1.23а);
- дополнительного оптического блока (рис. 1.23в) с дополнительными призмами подсветки (12) и линзой видеокамеры (13), располагающегося под электронным блоком мыши.

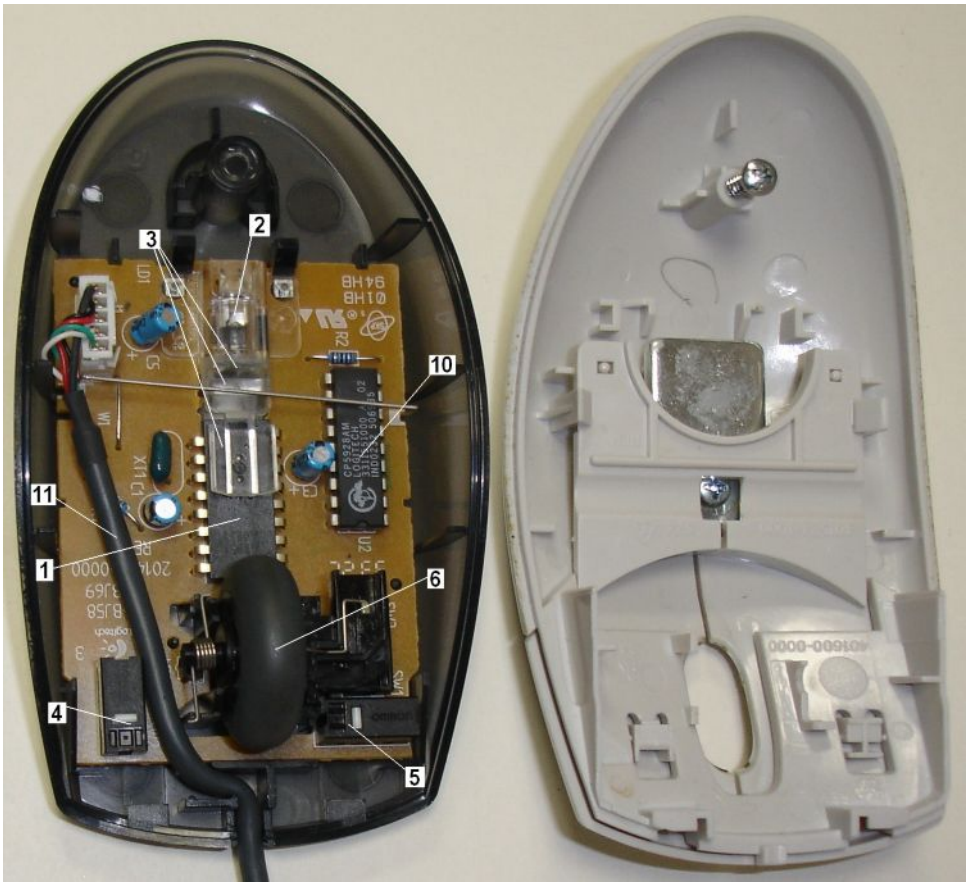

Рис. 1.22. Внутренняя конструкция оптической 3D USB/PS/2 мыши.

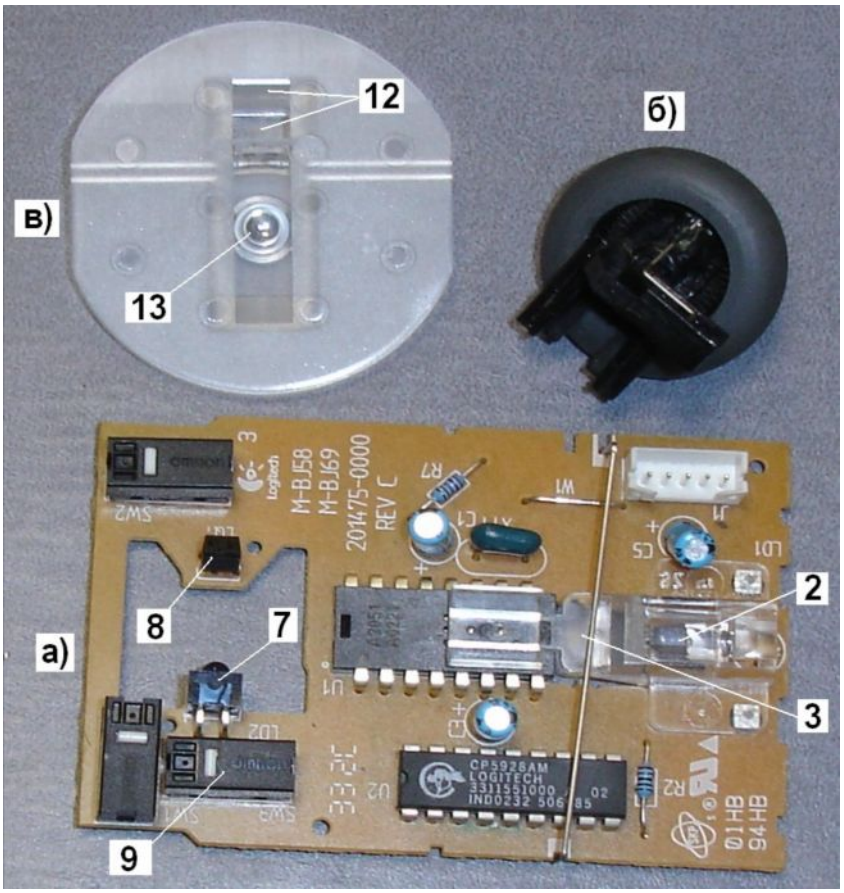

Рис. 1.23. Основные компоненты оптической 3D USB/PS/2 мыши: а) - электронный блок; б) - блок ролика прокрутки; в) - дополнительный оптический блок.

1.3.2. Схемотехника манипуляторов 3D PS/2 Mouse.

В данном разделе рассмотрены общие электрические схемы трех основных разновидностей манипуляторов типа 3D PS/2 Mouse:

- мышь с датчиком ролика прокрутки трещоточного типа;
- мышь сдатчиком ролика прокрутки на сдвоенных открытых оптопарах;
- оптическая мышь.

На рис. 1.24 представлена общая электрическая схема 3D PS/2 мыши с преобразователем перемещения по третьей координате, выполненным на базе контактного датчика типа трещотка.

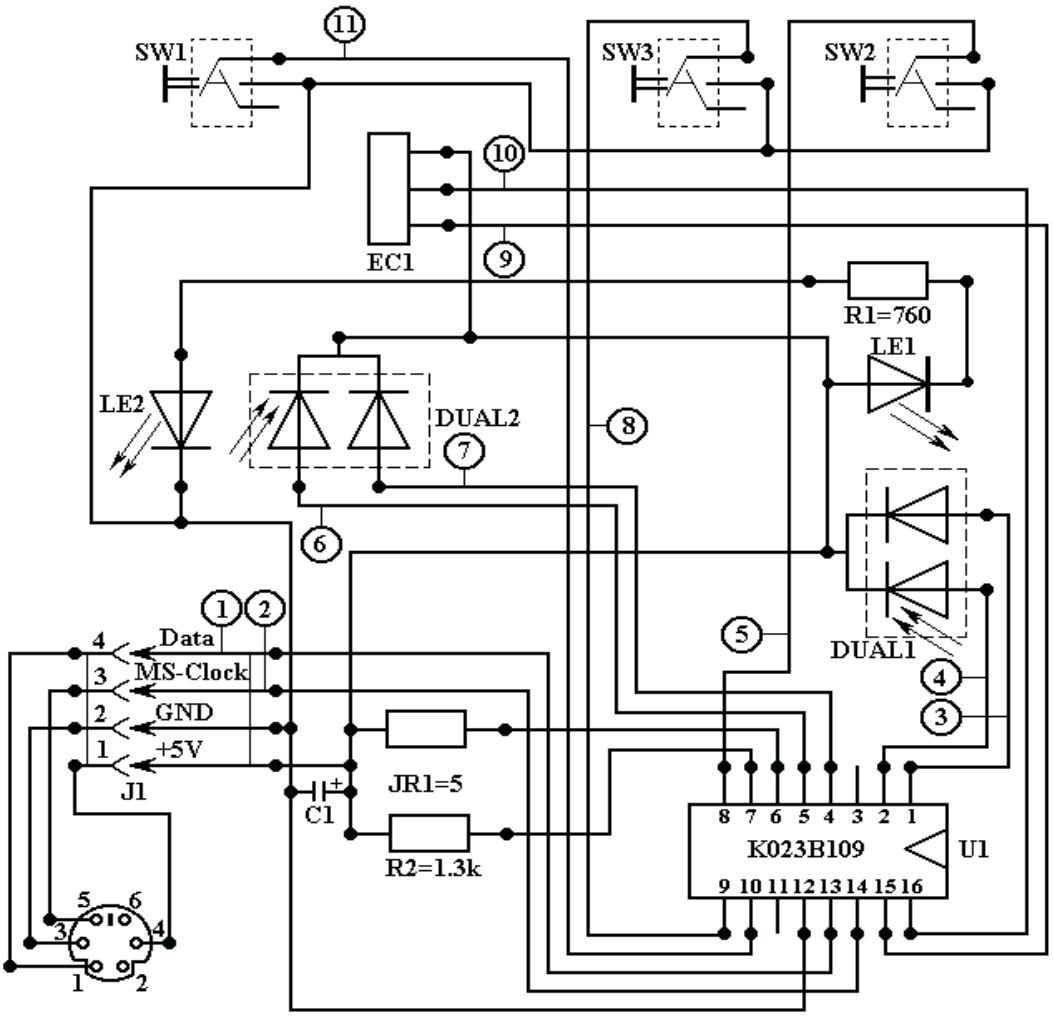

Рис. 1.24. Общая электрическая схема 3D PS/2 мыши с преобразователем трещоточного типа.

Компоненты электрической схемы.

- **EC1** - контактный датчик типа трещотка перемещения по третьей координате.
- **Интерфейсный кабель**, содержащий четыре линии, подключенные к контактам разъема **J1**:
  - контакт 1 - GND - сигнальная земля,
  - контакт 2 - +5V - шина питания,
  - контакт 3 - MS-Clock - линия синхронизации;
  - контакт 4 - Data - линия передачи данных.
- **JR1** - сопротивление - предохранитель величиной 5 Ом, через которое питание +5V подается на контакт 6 контроллера мыши U1.
- **R2** - сопротивление (резистор) величиной 1.3 кОм, через которое контакт 7 контроллера U1 подключается к шине питания +5V.
- **U1** - микроконтроллер мыши типа К023В109.
- **DUAL1** - сдвоенный фотодиод оптопары датчика перемещения по вертикали.
- **LE1** - светодиод этого же датчика.
- **DUAL2** - сдвоенный фотодиод оптопары датчика перемещения по горизонтали.
- **LE2** - светодиод этого же датчика.
- **R1** - сопротивление величиной 760 Ом, задающее величину тока через светодиоды LE1 и LE2.
- **SW1** - кнопочный датчик левой клавиши мыши.

- **SW2** - кнопочный датчик правой клавиши мыши.
- **SW3** - кнопочный датчик блока ролика прокрутки.
- **C1** - конденсатор фильтра развязки цепи питания.

Контрольные точки электрической схемы.

Контрольные точки обозначают цепи электрической схемы мыши, в которых можно контролировать соответствующие электрические сигналы.

1 - сигналы на линии данных интерфейса.
2 - сигналы на линии синхронизации интерфейса.
3, 4 - сигналы с фотодиодов датчика перемещения по вертикали.
5 - сигнал с кнопочного датчика правой клавиши мыши.
6, 7 - сигналы с фотодиодов датчика перемещения по горизонтали.
8 - сигнал с кнопочного датчика блока ролика прокрутки.
9, 10 - сигналы с датчика типа трещотка.
11 - сигнал с кнопочного датчика левой клавиши мыши.

На рис. 1.25 представлена общая электрическая схема 3D PS/2 мыши с преобразователем перемещения по третьей координате, выполненным на базе двойной открытой оптопары.

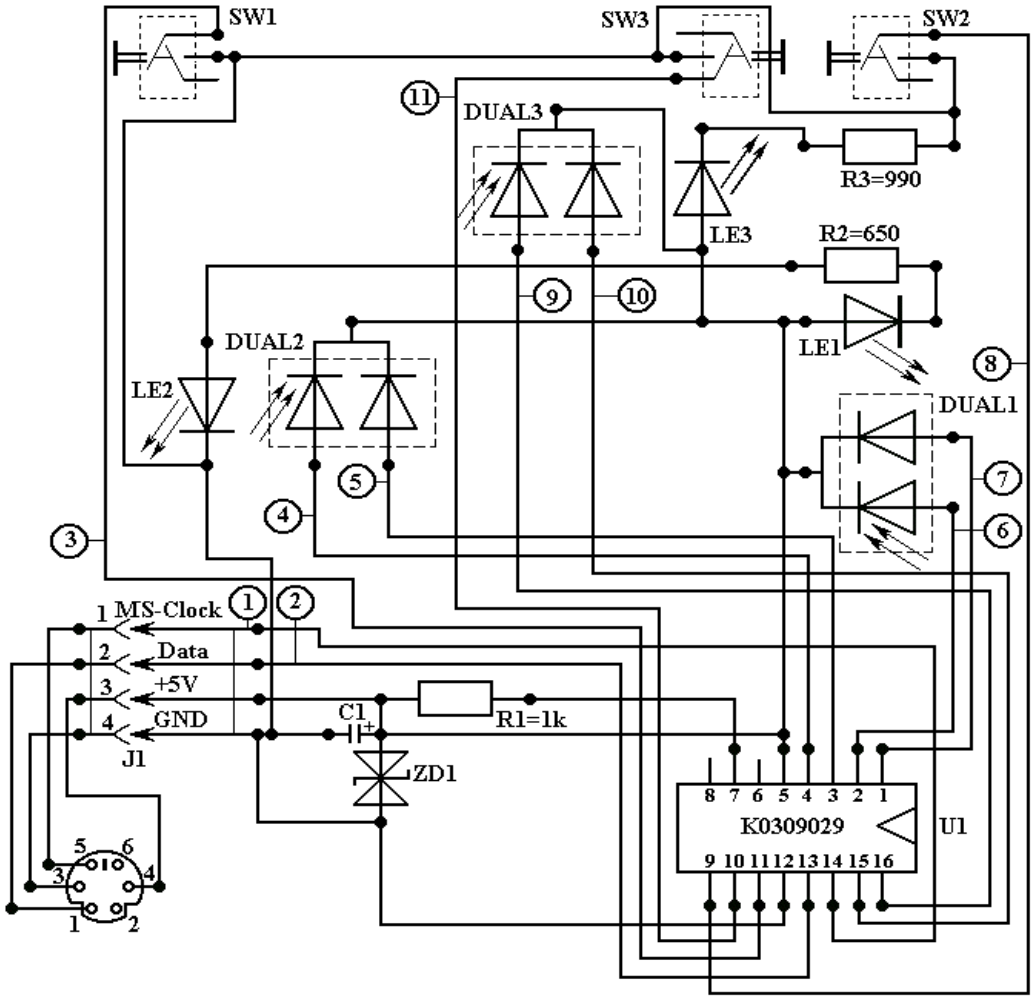

Рис. 1.25. Общая электрическая схема 3D PS/2 мыши с тремя сдвоенными открытыми оптопарами.

Компоненты электрической схемы.

- **Интерфейсный кабель**, содержащий четыре линии, подключенные к контактам разъема **J1**:
  - контакт 1 - MS-Clock - линия синхронизации;
  - контакт 2 - Data - линия передачи данных
  - контакт 3 - +5V - линия питания,
  - контакт 4 - GND - сигнальная земля.
- **R1** - сопротивление (резистор) величиной 1 кОм, через которое контакт 7 контроллера U1 подключается к шине питания +5V.
- **U1** - микроконтроллер мыши типа K0309029.
- **ZD1** - ограничительный стабилитрон цепи питания.
- **C1** - конденсатор фильтра развязки цепи питания.

- **DUAL1** - сдвоенный фотодиод оптопары датчика перемещения по вертикали.
- **LE1** - светодиод этого же датчика.
- **DUAL2** - сдвоенный фотодиод оптопары датчика перемещения по горизонтали.
- **LE2** - светодиод этого же датчика.
- **DUAL3** - сдвоенный светодиод оптопары датчика ролика прокрутки.
- **LE3** - светодиод этого же датчика.
- **R2** - сопротивление величиной 650 Ом, задающее величину тока через светодиоды LE1 и LE2.
- **R3** - сопротивление величиной 990 Ом, задающее величину тока через светодиод LE3.
- **SW1** - кнопочный датчик левой клавиши мыши.
- **SW2** - кнопочный датчик правой клавиши мыши.
- **SW3** - кнопочный датчик блока ролика прокрутки.

Контрольные точки электрической схемы.

Контрольные точки обозначают цепи электрической схемы мыши, в которых можно контролировать соответствующие электрические сигналы.

1 - сигналы на линии синхронизации интерфейса.
2 - сигналы на линии данных интерфейса.
3 - сигнал с кнопочного датчика левой клавиши мыши.
4, 5 - сигналы с фотодиодов датчика перемещения по горизонтали.
6, 7 - сигналы с фотодиодов датчика перемещения по вертикали.
8 - сигнал с кнопочного датчика правой клавиши мыши.
9, 10 - сигналы с фотодиодов датчика ролика прокрутки.
11 - сигнал с кнопочного датчика блока ролика прокрутки.

На рис. 1.26 представлена общая электрическая схема оптической 3D USB/PS/2 мыши, использующей в качестве датчиков перемещения по освещаемой поверхности встроенную видеокамеру с процессором, а в качестве датчика перемещения по третьей координате - сдвоенную оптопару.

Компоненты электрической схемы.

- **Интерфейсный кабель**, содержащий четыре линии и защитный экран, подключенные к контактам разъема **J1**:
  - контакт 5 - MS-Clock - линия синхронизации;
  - контакт 1 - Корпус - оплетка защитного экрана;
  - контакт 4 - Data - линия передачи данных;
  - контакт 3 - +5V - линия питания;
  - контакт 2 - GND - сигнальная земля.
- **R14** - ограничительное сопротивление цепи питания +5V.
- **R12** - сопротивление "подтягивания" линии данных к потенциалу земли.
- **R13** - сопротивление "подтягивания" линии синхронизации к потенциалу земли.
- **C9** - конденсатор защиты линии синхронизации от высокочастотных помех.
- **C8** - конденсатор защиты линии данных от высокочастотных помех.
- **R2** - сопротивление, через которое вход 8 контроллера U2 подключается к линии данных интерфейса. **LD1** - светодиод подсветки поверхности перемещения.
- **U2 -** основной контроллер мыши, поддерживающий протоколы обмена шины USB и интерфейса PS/2-mouse.
- **U1** - видеокамера и процессор видеокамеры манипулятора.
- **C4 -** конденсатор развязки цепи питания по высоким частотам.
- **C5** - конденсатор фильтра развязки цепи питания по низким частотам.
- **C2 и C3 -** конденсаторы высокочастотной и низкочастотной развязки внутренних цепей процессора видеокамеры.
- **R17 -** низкоомный резистор-перемычка подключения светодиода LD2 оптопары ролика прокрутки к цепи питания +5V.
- **R16 и R15 -** резисторы цепи задания тока через светодиод LD2, управляемой контроллером U2.
- **C10 и C11 -** конденсаторы высокочастотной развязки цепей взаимодействия процессора видеокамеры U1 и основного контроллера мыши U2.
- **R4 и R5 -** резисторы делителя, включенного в цепь управления контроллером U2 (контакт 18) транзистором Q1, задающим напряжение в цепи питания процессора U1 (контакт 13) и светодиода подсветки LD1.
- **C1 и C6** - конденсаторы низкочастотной и высокочастотной развязки цепи питания процессора U1.
- **R6** - балластное сопротивление цепи питания процессора U1.
- **R8** - токоограничивающее сопротивление в цепи контакта 15 процессора U1.
- **R7** - резистор подключения контакта 14 процессора U1 к шине сигнальной земли.
- **X1** - кварцевый резонатор.
- **SW1** - кнопочный датчик левой клавиши мыши.
- **SW2** - кнопочный датчик правой клавиши мыши.

- **SW3** - кнопочный датчик блока ролика прокрутки.

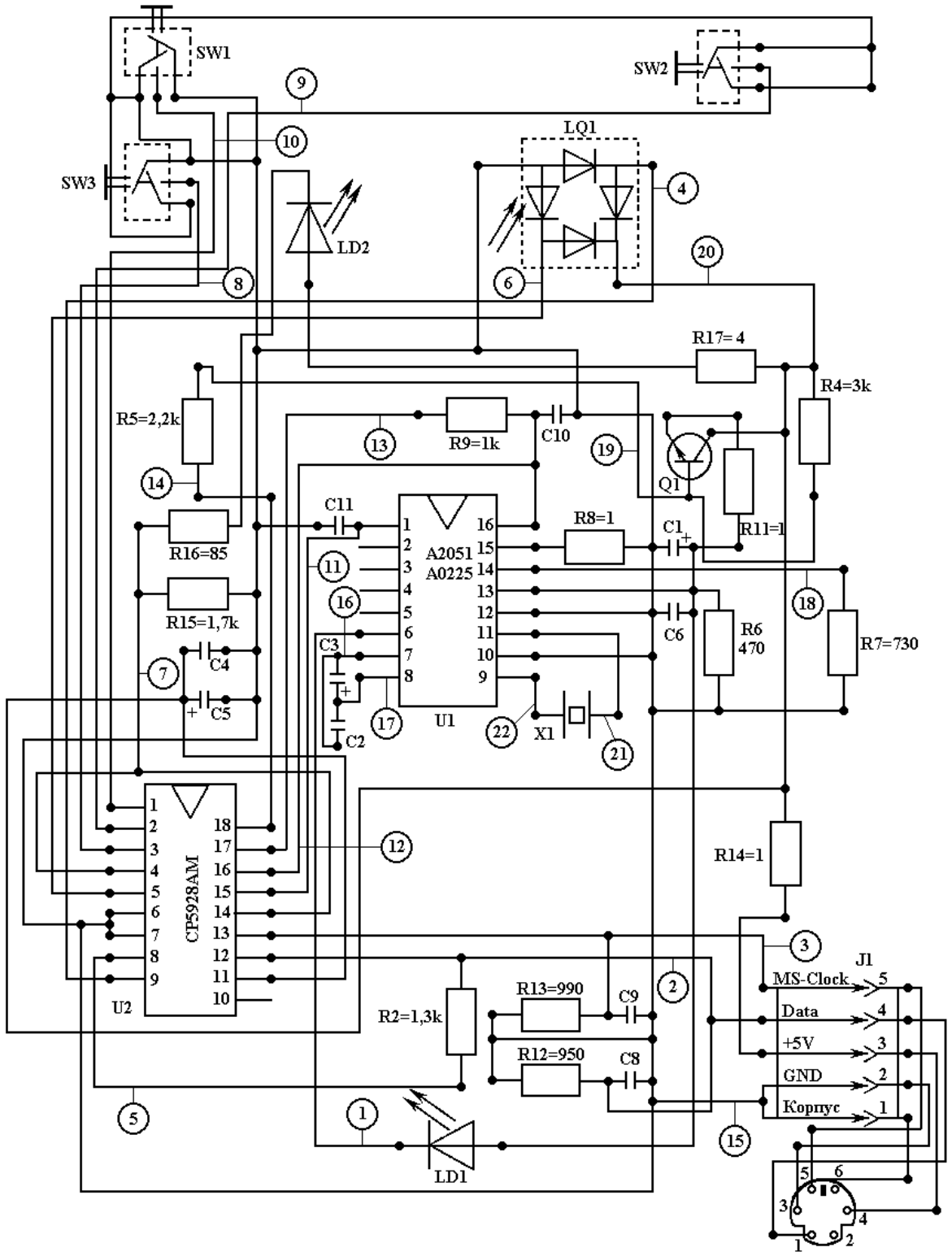

Рис. 1.26. Общая электрическая схема оптической 3D USB/PS/2 Mouse.

Контрольные точки электрической схемы.

Контрольные точки обозначают цепи электрической схемы оптической мыши, в которых можно контролировать характерные электрические сигналы.

1 - сигнал управления светодиодом LD1 подсветки, поступающий с контакта 6 процессора видеокамеры U1.

2 - сигналы на линии данных интерфейса мыши MS-Data.

3 - сигналы на линии синхронизации мыши MS-Clock.

4 - сигнал с правого плеча фотодиодного моста LQ1 датчика ролика прокрутки.

5 - сигналы, поступающие на контакт 8 контроллера U2 с линии данных интерфейса мыши.

6 - сигнал с левого плеча фотодиодного моста LQ1 датчика ролика прокрутки.

7 - сигнал управления током через светодиод LD2 датчика ролика прокрутки; поступает с контактов 4 и 14 контроллера U2.

8 - сигнал с кнопочного датчика SW3 блока ролика прокрутки.

9 - сигнал с кнопочного датчика SW2 правой клавиши мыши.

10 - сигнал с кнопочного датчика SW1 левой клавиши мыши.

11 - сигналы цепи взаимодействия U1 и U2 через контакты U1-1 и U2-15.

12 - сигналы цепи взаимодействия U1 и U2 через контакты U1-16 и U2-16.

13 - сигналы, поступающие на контакт 17 контроллера U2 из цепи, обозначенной контрольной точкой 12.

14 - сигнал управления транзистором Q1, поступающий с контакта 18 контроллера U2: он через делитель R5, R4 поступает на базу транзистора Q1.

15 - сигнальная "Земля" (GND)

16 - сигналы на контакте 7 процессора U1, соединенного через параллельно включенные конденсаторы C2 и C3 (C2 - электролитический, C3 - керамический) с его контактом 8.

17 - сигнал контакта 8 процессора U1.

18 - сигнал контакта 14 процессора U1.

19 - сигнал на базе транзистора Q1.

20 - потенциал на шине питания мыши.

21 - сигнал кварцевого резонатора, поступающий на вывод 11 U1.

22 - сигнал кварцевого резонатора, поступающий на вывод 9 U1.

1.3.3. Интерфейс PS/2 Mouse.

PS/2-Mouse подключается к системе через двунаправленный последовательный интерфейс, аналогичный интерфейсу клавиатуры и обслуживается, как правило, тем же адаптером.

На физическом уровне адаптер обеспечивает передачу и прием по линиям данных и синхронизации интерфейса последовательностей электрических сигналов, с помощью которых реализуется передача и прием последовательностей битов информации, синхронизация передачи и приема, а также асинхронный протокол обмена, предусматривающий запросы и квитирование передач, и управление потоком данных. Отличия от интерфейса клавиатуры выражаются только во временных параметрах используемых сигналов и правилами подачи смещения на линии данных и синхронизации.

Для определения конкретных временных параметров сигнального протокола интерфейса PS/2 Mouse можно воспользоваться простой программой, представленной в листинге 1.16 (программа CPSD4E.TXT), выполняемой в среде отладчика DOS (Debug или AFD).

Листинг 1.16. Циклическое выполнение команды передачи в PS-Mouse, работающей в эхо режиме, кода EEh (программа CPSD4E.TXT).

```
2C47:0100  BB0011    MOV    BX,1100 ;Число циклов выполнения программы.
2C47:0103  FA        CLI            ;Запрет аппаратных прерываний.
2C47:0104  E464      IN     AL,[64] ;Чтение рег. состояния адаптера и
2C47:0106  A802      TEST   AL,02   ;проверка: Вх. буфер пуст? Если нет,
2C47:0108  75FA      JNZ    0104    ;то новый цикл проверки.
2C47:010A  B0D4      MOV    AL,D4   ;Код команды D4h передается
2C47:010C  E664      OUT    [64],AL ;в адаптер.
2C47:010E  E464      IN     AL,[64] ;Чтение рег. состояния адаптера и
2C47:0110  A802      TEST   AL,02   ;проверка: Вх. буфер пуст? Если нет,
2C47:0112  75FA      JNZ    010E    ;то новый цикл проверки.
2C47:0114  B0ED      MOV    AL,EE   ;Код команды EEh "Установить эхо-режим"
2C47:0116  E660      OUT    [60],AL ;передается в PS/2 мышь.
2C47:0118  E464      IN     AL,[64] ;Чтение рег. состояния адаптера и
2C47:011A  A801      TEST   AL,01   ;проверка признака "вых. буфер полон":
2C47:011C  74FA      JZ     0118    ;если пуст, то на новый цикл.
2C47:011E  E460      IN     AL,[60] ;Чтение кода подтверждения мыши
2C47:0120  88C4      MOV    AH,AL   ;и пересылка его в регистр AH.
2C47:0122  E464      IN     AL,[64] ;Чтение рег. состояния адаптера и
2C47:0124  A802      TEST   AL,02   ;проверка: Вх. буфер пуст? Если нет,
2C47:0126  75FA      JNZ    0122    ;то новый цикл проверки.
2C47:0128  E464      IN     AL,[64] ;Чтение рег. состояния адаптера и
2C47:012A  A801      TEST   AL,01   ;проверка признака "вых. буфер полон":
2C47:012C  7406      JZ     0134    ;если пуст, то продолжение программы,
2C47:012E  E460      IN     AL,[60] ;иначе чтение вых. буфера,
2C47:0130  89C6      MOV    SI,AX   ;запоминание его содержимого в регистре SI
2C47:0132  EBF4      JMP    0128    ;и переход на новый цикл проверки.
2C47:0134  B0D4      MOV    AL,D4   ;Код команды D4h передается
```

```
2C47:0136  E664      OUT    [64],AL ;в адаптер.
2C47:0138  E464      IN     AL,[64] ;Чтение рег. состояния адаптера и
2C47:013A  A802      TEST   AL,02   ;проверка: Вх. буфер пуст? Если нет,
2C47:013C  75FA      JNZ    0138    ;то новый цикл проверки.
2C47:013E  B0EE      MOV    AL,EE   ;Код EEh передается
2C47:0140  E660      OUT    [60],AL ;в PS/2 мышь.
2C47:0142  E464      IN     AL,[64] ;Чтение рег. состояния адаптера и
2C47:0144  A801      TEST   AL,01   ;проверка признака "вых. буфер полон":
2C47:0146  74FA      JZ     0142    ;если пуст, то на новый цикл проверки.
2C47:0148  E460      IN     AL,[60] ;Чтение кода ответа мыши
2C47:014A  88C4      MOV    AH,AL   ;и запоминание его в регистре AH.
2C47:014C  B90F00    MOV    CX,000F ;Вложенные
2C47:014F  51        PUSH   CX      ;циклы
2C47:0150  B9FFFF    MOV    CX,FFFF ;программной
2C47:0153  E2FE      LOOP   0153    ;задержки
2C47:0155  59        POP    CX
2C47:0156  E2F7      LOOP   014F
2C47:0158  4B        DEC    BX      ;Уменьшение счетчика циклов BX.
2C47:0159  75C7      JNZ    0122    ;Если BX не 0, то новый цикл программы.
2C47:015B  FB        STI            ;Разрешение прерываний.
2C47:015C  CC        INT3           ;Остановка и передача упр. отладчику.
```

В программе реализовано циклическое выполнение команды D4h передачи байта данных в PS/2 Mouse, работающей в режиме "Эхо". В качестве байта данных выбран код EEh для того, чтобы можно было определить отличия сигнального протокола интерфейса PS/2 Mouse от сигнального протокола интерфейса клавиатуры, представленного в разделе 1.2.5.4.2 (осциллограммы (рис. 1.19) передачи в клавиатуру команды EEh "Эхо"). Программой CPSD4E.BIN код данных EEh передается через адаптер и интерфейс в PS/2 мышь, которая в ответ передает через интерфейс и адаптер код ответа EEh. Подключившись осциллографом к линиям MS-Clock и MS-Data интерфейса мыши можно фиксировать электрические сигналы, реализующие передачу этих кодов на физическом уровне в соответствие с сигнальным протоколом аппаратного интерфейса мыши. На рис 1.27 представлены осциллограммы соответствующих сигналов интерфейса мыши.

После окончания работы программы PS/2 мышь остается в режиме работы "Эхо" а разряд D5 (выходной буфер мыши полон") в единичном состоянии.

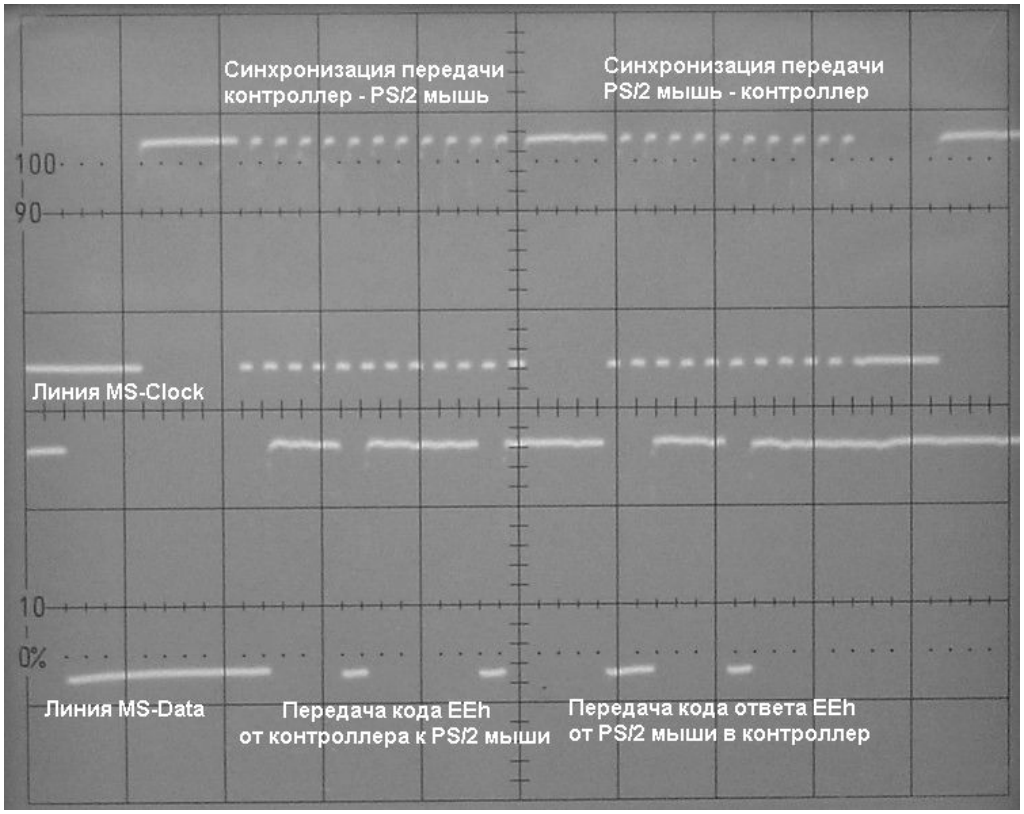

Рис. 1.27. Осциллограммы сигналов интерфейса PS/2 мыши, зарегистрированные при выполнении программы CPSD4E.BIN: развертка - 0,32 мс/дел., чувствительность по напряжению - 2 В/дел.; развертка запускается отрицательным фронтом сигнала линии MS-Clock.

На канальном уровне адаптер обеспечивает обмен байтами команд и данных между системой и мышью,

используя те же порты ввода-вывода, что и клавиатура (60h и 64h). Отличие заключается в том, что:

- при передаче команд и данных в мышь перед записью каждого байта (и кода команды, и ее параметра) в порт 60h необходимо предварительно в порт 64h записывать команду адаптера D4h - передать мыши байт данных;
- при приеме байта данных от мыши адаптер устанавливает в регистре состояния флаг "выходной буфер мыши полон" (разряд D5) после чего примерно через 15 мкс передает полученный байт в выходной буфер адаптера и устанавливает флаг "выходной буфер полон" (разряд D0). При установке разряда D5 вырабатывается прерывание IRQ12, если оно не запрещено командным байтом адаптера, а при установке разряда D0 - прерывание IRQ1, если оно не запрещено командным байтом. Прерывание IRQ12 необходимо закрепить за мышью с помощью процедуры BIOS SETUP, иначе оно может быть захвачено каким-либо другим устройством и станет для мыши недоступным.

Более подробная информация о взаимодействии PS/2 Mouse и системы на канальном уровне приводится ниже в разделе 1.3.4.

Интерфейсный разъем PS/2 Mouse аналогичен разъему PS/2 клавиатуры (6-pin mini-DIN). Вид разъема (со стороны задней панели системного блока) и назначение контактов приведены на рис. 1.28.

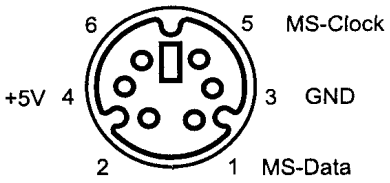

Рис. 1.28. Разъем PS/2-Mouse

1.3.4. Функционирование PS/2 Mouse.

Мышь PS/2 может находиться в одном из указанных ниже режимов работы.

- Потоковый режим - мышь выдает 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, то есть по команде считывания данных Read Data (EBh).
- Эхо-режим - любой байт данных, переданный компьютером (кроме ECh и FFh), мышь отсылает обратно.

После включения напряжения питания или получения команды Reset (FFh) мышь осуществляет самотестирование и примерно через 0,5 секунд посылает компьютеру последовательность байтов: AAh, 00h -в случае нормального завершения теста, или FCh, 00h - в случае обнаружения ошибки. В обоих случаях второй байт 00h является идентификатором устройства типа PS/2 Mouse. 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 (F4h) (которая активизирует мышь).

Управление режимами работы мыши и обменом байтами данных с ней может осуществляется непосредственно через порты 60h и 64h адаптера, через функции прерывания INT 15h, обрабатываемые BIOS, функциями программного прерывания INT 33h, обрабатываемого DOS и средствами стандартных прикладных программных интерфейсов, например Win32 API в среде современных ОС.

1.3.4.1. Группа форматов PS/2 Mouse.

При работе PS/2 мыши в потоковом режиме или в режиме дистанционного управления она передает информацию о своем перемещении, о прокрутке ролика третьей координаты и об изменение состояния своих кнопочных датчиков байтовыми пакетами, имеющими определенный формат.

Стандартный формат передачи данных для мыши PS/2 типа, разработанный фирмой IBM, показан в таблице 1.22.

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).

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, то есть противоположно оси Y дисплея. Преимуществом формата PS2 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).

Таблица 1.22. 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[1]	R	L
2	X7	X6	X5	X4	X3	X2	X1	X0
3	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

[1] У стандартных двухкнопочных манипуляторов этот бит всегда равен нулю.

В течение длительного времени область применения устройств PS/2-типа была ограниченной, однако с внедрением стандарта ATX началось их интенсивное распространение и развитие. В частности, появились новые, четырехбайтные протоколы передачи данных для трехкоординатных устройств, показанные в таблицах 1.23 и 1.24. При включение питания такие устройства начинают работать в стандартном режиме PS/2 со стандартным трехбайтным протоколом, а переключение в режим 3D с четырехбайтным протоколом происходит по специальной команде от драйвера мыши.

Формат 3D PS/2 Mouse предусматривает передачу координаты Z в четвертом байте пакета - в виде восьмиразрядного двоичного числа (старший разряд числа содержит знак). Формат пятикнопочной мыши Wheel Mouse отличается от предыдущего тем, что для передачи перемещения по Z применяются только четыре разряда, а еще два кодируют состояние специальных кнопок B4 и B5.

Таблица 1.23. Формат 3D PS/2 Mouse

Номер байта в посылке	Номер бита							
	7	6	5	4	3	2	1	0
1	0	0	YS	XS	1	M	R	L
2	X7	X6	X5	X4	X3	X2	X1	X0
3	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
4	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0

Таблица 1.24. Формат данных 5-кнопочной мыши Wheel Mouse

Номер байта в посылке	Номер бита							
	7	6	5	4	3	2	1	0
1	0	0	YS	XS	1	M	R	L
2	X7	X6	X5	X4	X3	X2	X1	X0
3	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
4	0	0	B5	B4	Z3	Z2	Z1	Z0

1.3.4.2. Непосредственная работа с PS/2 Mouse на уровне ее команд.

Манипулятор PS/2 Mouse, по сравнению COM Mouse, является достаточно "интеллектуальным" устройством и может поддерживать выполнение ряда команд, используемых системой для управления ее режимами работы. Как и команды управления клавиатурой, команды для PS/2 Mouse передаются в мышь через порт 60h, но с предварительной отработкой адаптером команды D4h - передать мыши байт данных.

Список команд, выполняемых мышью PS/2, приведен в таблице 1.25. Правила передачи команд в мышь и получения данных из мыши в общих чертах аналогичны правилам взаимодействия с клавиатурой. Как и при работе с клавиатурой каждая передача байта в PS/2 Mouse должна подтверждаться ей кодами положительного FAh (ACK), отрицательного FEh (NAC) подтверждения или другими байтовыми посылками. На взаимодействие системы с мышью через порт 60h распространяются также и правила взаимодействия системы с адаптером клавиатуры/мыши.

Программа, реализующая передачу байта в манипулятор PS/2 Mouse должна предусматривать выполнение следующих основных действий:

1. Запретить обработку аппаратных прерываний командой CLI.

2. Проверить состояние выходного буфера адаптера (выходного буфера клавиатуры) и, если он не пуст, очистить его командой чтения выходного буфера (IN AL, [60]).

3. Проверить состояние входного буфера и, если он занят, дождаться его освобождения адаптером. После освобождения входного буфера переслать по адресу 64h команду D4h -передать мыши байт данных.

4. После освобождения входного буфера от кода команды D4h по адресу 60h записать байт, предназначенный для передачи в мышь.

5. Затем организовать цикл ожидания активного признака "выходной буфер (адаптера) полон" (D0=1 в регистре состояния адаптера). Этот признак устанавливается, когда мышь передаст в адаптер ответный байт (код положительного подтверждения FAh, код отрицательного подтверждения FEh или какой то другой код, предусмотренный режимом работы мыши), который сначала фиксируется в выходном буфере мыши, уста-

навливая признак "выходной буфер мыши полон" (разряд D5 регистра состояния адаптера), а затем (примерно через 15 микросекунд) переписывается в выходной буфер адаптера, устанавливая признак "выходной буфер (адаптера, клавиатуры) полон" (разряд D0 регистра состояния адаптера) и становится доступным для чтения через порт 60h.

6. После установки активного признака "выходной буфер полон" выполнить чтение из порта 60h, что сбрасывает признак "выходной буфер полон", но признак "выходной буфер мыши полон" не сбрасывается. На этом цикл передачи байта в мышь заканчивается.

7. Если дальнейших операций с мышью в программе не предусматривается, то необходимо сбросить признак "выходной буфер мыши полон" любой командой адаптера, предусматривающей возвращение в выходной буфер адаптера байта результата выполнения этой команды (см. раздел 1.1.8.3.5).

8. Если дальнейших операций в программе не предусмотрено, то необходимо командой STI разрешить обработку аппаратных прерываний и передать управление ведущей программе.

Таблица 1.25. Команды, выполняемые мышью PS/2

Код	Команда	Выполняемое действие
E6h	Reset Scaling	Отменить масштабирование перемещения
E7h	Set Scaling 2:1	Установить масштабирование перемещения для увеличения чувствительности мыши (см. табл. 1.26)
E8h	Set Resolution	Установить разрешение
E9h	Status Request	Получить информацию о состоянии устройства
EAh	Set Stream Mode	Установить потоковый режим
EBh	Read Data	Выдать пакет данных
ECh	Reset Wrap Mode	Отменить эхо-режим
EEh	Set Wrap Mode	Установить эхо-режим
F0h	Set Remote Mode	Установить режим дистанционного управления
F2h	Read Device Type	Определить тип устройства
F3h	Set Sampling Rate	Установить частоту дискретизации
F4h	Enable	Разрешить передачу данных от мыши
F5h	Disable	Остановить передачу данных от мыши
F6h	Set Default	Установить значение параметров работы мыши, используемое по умолчанию
FEh	Resend	Повторить передачу данных
FFh	Reset	Осуществить сброс и произвести самотестирование устройства

Чтение поступающих от PS/2 Mouse байтов данных почти ни чем не отличается от приема данных от клавиатуры. Отличить данные, поступающие с PS/2 Mouse, от данных клавиатуры можно по состоянию признака "выходной буфер мыши полон" (разряд D5 регистра состояния адаптера): если D5=1, то байт получен от мыши. Байт, поступающий с клавиатуры, сбрасывает разряд D5 и устанавливает разряд D0.

Если чтение данных от мыши и клавиатуры осуществляется обработчиками их прерываний, то синхронизация их работы также осуществляется однозначно. Так, при поступление байта от мыши устанавливается признак "выходной буфер мыши полон", на основании которого формируется запрос на прерывание по линии IRQ12 (номер вектора 74h - INT 74h) и активизируется обработчик прерывания мыши. Этот обработчик блокирует выработку запроса прерывания по линии IRQ 1 (номер вектора 09h - INT 09h), который может активизироваться при установке признака "выходной буфер (клавиатуры) полон" и осуществляет чтение байта из выходного буфера адаптера по адресу 60h, сбрасывает признак "выходной буфер мыши полон" и разрешает прерывание по линии IRQ 1. При поступление байта с клавиатуры устанавливается признак D0=1 в регистре состояния адаптера, генерируется запрос IRQ 1 и активизируется обработчик прерываний клавиатуры, который считывает байт из выходного буфера адаптера.

Одновременное поступление байтов от клавиатуры и мыши невозможно, поскольку при активизации передачи по одному из интерфейсов адаптер блокирует работу другого интерфейса и наоборот.

Рассмотрим представленные в таблице 9 команды более подробно.

Команда E6h (Reset Scaling) отменяет режим масштабирования перемещения (устанавливает масштаб 1:1). Пример программного модуля, выполняемого в среде отладчика AFD и реализующего алгоритм отмены режима масштабирования в PS/2 мыши, приведен в листинге 1.17.

Листинг 1.17. Программа, реализующая отмену режима масштабирования командой E6h (PSE6.TXT).

```
2C47:0100  FA        CLI            ;Запрет прерываний.
2C47:0101  E464      IN     AL,[64] ;Чтение регистра состояния.
2C47:0103  A801      TEST   AL,01   ;Проверка: выходной буфер пуст?
2C47:0105  7404      JZ     010B    ;Если пуст, то на проверку входн.буфера.
2C47:0107  E460      IN     AL,[60] ;Очистка вых.буфера чтением.
2C47:0109  EBF6      JMP    0101    ;На новый цикл проверки.
2C47:010B  E464      IN     AL,[64] ;Чтение регистра состояния.
2C47:010D  A802      TEST   AL,02   ;Проверка: входной буфер пуст?
2C47:010F  75FA      JNZ    010B    ;Если "нет", то на новый цикл проверки.
2C47:0111  B0D4      MOV    AL,D4   ;Код команды "передать байт данных в мышь"
```

```
2C47:0113  E664      OUT    [64],AL ;посылается в адаптер.
2C47:0115  E464      IN     AL,[64] ;Чтение регистра состояния.
2C47:0117  A802      TEST   AL,02   ;Входной буфер пуст?
2C47:0119  75FA      JNZ    0115    ;Если не пуст, то на новый цикл проверки.
2C47:011B  B0E6      MOV    AL,E6   ;Код команды "отменить масштабирование"
2C47:011D  E660      OUT    [60],AL ;передается в PS/2 мышь.
2C47:011F  E464      IN     AL,[64] ;Чтение регистра состояния.
2C47:0121  A820      TEST   AL,20   ;Проверка: выходной буфер мыши заполнен?
2C47:0123  74FA      JZ     011F    ;Если пуст, то новый цикл проверки.
2C47:0125  E464      IN     AL,[64] ;Чтение регистра состояния.
2C47:0127  A801      TEST   AL,01   ;Выходной буфер (адаптера) заполнен?
2C47:0129  74FA      JZ     0125    ;Если пуст, то новый цикл проверки.
2C47:012B  E460      IN     AL,[60] ;Чтение кода ответа мышь
2C47:012D  88C4      MOV    AH,AL   ;и передача его в регистр AH.
2C47:012F  FB        STI            ;Разрешение прерываний.
2C47:0130  CC        INT3           ;Передача управления отладчику.
```

В среде AFD программа запускается на выполнение командой G100, набираемой в командной строке отладчика. Эта команда инициируется на выполнение нажатием клавиши "Enter". Время между моментом поступления в адаптер клавиатуры скан-кода нажатия клавиши "Enter" и до момента передачи адаптером кода команды E6h в мышь составляет примерно 35 миллисекунд, при работе на компьютере Pentium III с частотой процессора 500 МГц. После завершения выполнения программы в регистре AH будет находиться код FAh - нормальное завершение команды. Разряд D5 регистра состояния адаптера (признак "выходной буфер мыши заполнен") будет сброшен адаптером при получении им от клавиатуры кода отпускания клавиши "Enter", если клавиша удерживалась нажатой более 35 миллисекунд, или остается в активном состоянии в противном случае (см. пункт 1.1.8.3.6).

Команда E7h (Set Scaling 2:1) устанавливает режим масштабирования перемещения 2:1 для увеличения чувствительности мыши (таблица 1.26).

Таблица 1.26. Масштабирование перемещения в режиме 2:1

Реальное перемещение	Информация о перемещении, выдаваемая мышью
0	0
1	1
2	1
3	3
4	6
5	9
N(≥6)	2N

Выполнение команды E7h можно реализовать с помощью программы, приведенной в листинге 1.17, изменив код E6 на E7 в строке с адресом :011B.

Команда E8h (Set Resolution) служит для установки используемого разрешения, код которого передается в следующем за командой байте данных и определяет число отсчетов на миллиметр (00h - 1 отсчет, 01h - 2 отсчета, 02h - 4 отсчета, 03h - 8 отсчетов). Для реализации выполнения команды E8h можно доработать программу PSE6.TXT (листинг 1.17), добавив в нее после адреса :012D команды, реализующие передачу в мышь байта, определяющего число отсчетов на миллиметр. В листинге 1.18 представлен текст этой модифицированной программы (программа PSE8.TXT).

Листинг 1.18. Установка разрешения 8 отсчетов на миллиметр в PS/2 мыши (PSE8.TXT)

```
2C47:0100  FA        CLI            ;Запрет прерываний.
2C47:0101  E464      IN     AL,[64] ;Чтение регистра состояния.
2C47:0103  A801      TEST   AL,01   ;Проверка: выходной буфер пуст?
2C47:0105  7404      JZ     010B    ;Если пуст, то на проверку входн.буфера.
2C47:0107  E460      IN     AL,[60] ;Очистка вых.буфера чтением.
2C47:0109  EBF6      JMP    0101    ;На новый цикл проверки.
2C47:010B  E464      IN     AL,[64] ;Чтение регистра состояния.
2C47:010D  A802      TEST   AL,02   ;Проверка: входной буфер пуст?
2C47:010F  75FA      JNZ    010B    ;Если "нет", то на новый цикл проверки.
2C47:0111  B0D4      MOV    AL,D4   ;Код команды "передать байт данных в мышь"
2C47:0113  E664      OUT    [64],AL ;посылается в адаптер.
2C47:0115  E464      IN     AL,[64] ;Чтение регистра состояния.
2C47:0117  A802      TEST   AL,02   ;Входной буфер пуст?
2C47:0119  75FA      JNZ    0115    ;Если не пуст, то на новый цикл проверки.
2C47:011B  B0E8      MOV    AL,E8   ;Код команды "установить разрешение"
2C47:011D  E660      OUT    [60],AL ;передается в PS/2 мышь.
```

```
2C47:011F  E464      IN     AL,[64] ;Чтение регистра состояния.
2C47:0121  A820      TEST   AL,20   ;Выходной буфер мыши заполнен?
2C47:0123  74FA      JZ     011F    ;Если пуст, то новый цикл проверки.
2C47:0125  E464      IN     AL,[64] ;Чтение регистра состояния.
2C47:0127  A801      TEST   AL,01   ;Выходной буфер (адаптера) заполнен?
2C47:0129  74FA      JZ     0125    ;Если пуст, то новый цикл проверки.
2C47:012B  E460      IN     AL,[60] ;Чтение кода ответа мышь
2C47:012D  88C4      MOV    AH,AL   ;и передача его в регистр AH.
2C47:012F  B0D4      MOV    AL,D4   ;Код команды "передать байт данных в мышь"
2C47:0131  E664      OUT    [64],AL ;посылается в адаптер.
2C47:0133  E464      IN     AL,[64] ;Чтение регистра состояния.
2C47:0135  A802      TEST   AL,02   ;Проверка: входной буфер пуст?
2C47:0137  75FA      JNZ    0133    ;Если не пуст, то на новый цикл проверки.
2C47:0139  B003      MOV    AL,03   ;Код разрешения "8 отсчетов" (03h)
2C47:013B  E660      OUT    [60],AL ;передается в PS/2 мышь.
2C47:013D  E464      IN     AL,[64] ;Чтение регистра состояния.
2C47:013F  A820      TEST   AL,20   ;Выходной буфер мыши заполнен?
2C47:0141  74FA      JZ     013D    ;Если пуст, то новый цикл проверки.
2C47:0143  E464      IN     AL,[64] ;Чтение регистра состояния.
2C47:0145  A801      TEST   AL,01   ;Выходной буфер (адаптера) заполнен?
2C47:0147  74FA      JZ     0143    ;Если пуст, то новый цикл проверки.
2C47:0149  E460      IN     AL,[60] ;Чтение кода ответа мыши
2C47:014B  88C3      MOV    BL,AL   ;и передача его в регистр BL.
2C47:014D  FB        STI            ;Разрешение прерываний.
2C47:014E  CC        INT3           ;Конец и передача управления отладчику.
```

После выполнения программы в регистрах AH и BL должны находиться коды подтверждения FAh.

Команда E9h (Status Request) вызывает выдачу специального трехбайтного пакета данных, содержащего информацию о статусе устройства (то есть о текущих значениях параметров работы): формат первого байта показан в таблице 1.27, второй байт содержит код разрешения, третий - код частоты дискретизации.

Таблица 1.27. Формат первого байта статуса

Номер бита	Значение
0	Состояние правой кнопки (0 - отпущена, 1 - нажата)
1	Состояние средней кнопки (0 - отпущена, 1 - нажата)
2	Состояние левой кнопки (0 - отпущена, 1 - нажата)
3	Зарезервирован (всегда 0)
4	Масштабирование (0 - режим 1:1, 1 - режим 2:1)
5	Передача данных (0 - запрещена, 1 - разрешена)
6	Режим (0 - потоковый, 1 - дистанционного управления)
7	Зарезервирован (всегда 0)

Программа PSE9N.TXT, представленная в листинге 1.19, реализует выполнение манипулятором PS/2 Mouse команды E9h "Получить информацию о состоянии устройства". Программа работает совместно с набором подпрограмм, представленных в листинге 1.20 (подпрограммы PSPPN.TXT).

PS/2 Mouse в ответ на команду E9h возвращает программе четыре байта:

1. - байт кода подтверждения принятия команды - код FAh;
2. - байт, характеризующий состояние манипулятора (Таблица 1.27);
3. - байт кода разрешения;
4. - байт кода частоты дискретизации.

В среде отладчика AFD программа PSE9N.BIN загружается с относительного адреса 0200h а набор подпрограмм PSPPN.BIN - с относительного адреса 0100h.

В программе PSE9N после окончания очередного цикла чтения четырех байт, поступивших от мыши в ответ на команду E9h, предусмотрен сброс признака "выходной буфер мыши полон" (разряд D5 регистра состояния адаптера) выполнением команды адаптера 20h - прочитать командный байт адаптера.

После окончания работы программы в буфере памяти DS:0000 - DS:0003 находятся четыре, поступивших от мыши байта: FAh, 00h, 02h, 64h.

На рис. 1.29 приведены временные диаграммы сигналов интерфейса PS/2 мыши, зарегистрированные во время выполнения программы PSE9N.BIN и иллюстрирующие процессы взаимодействия адаптера и PS/2 мыши во время реализации выполнения команды E9h.

Листинг 1.19. Программа циклического выполнения мышью команды E9h (программа PSE9N.TXT).

```
2C47:0200  BBFE00    MOV    BX,00FE ;Число циклов выполнения программы.
```

```
2C47:0203  8CD8      MOV    AX,DS   ;Задание начального адреса буфера
2C47:0205  8EC0      MOV    ES,AX   ;памяти.
2C47:0207  BF0000    MOV    DI,0000 ;Позиционирование на начало буфера.
2C47:020A  FA        CLI            ;Запрет прерываний.
2C47:020B  E82CFF    CALL   013A    ;Очистка выходного буфера адаптера.
2C47:020E  E8EFFE    CALL   0100    ;Ожидание освобождения вх.буф.адаптера.
2C47:0211  B0D4      MOV    AL,D4   ;Посылка в адаптер команды D4h-передать
2C47:0213  E664      OUT    [64],AL ;мыши байт данных.
2C47:0215  E8E8FE    CALL   0100    ;Ожидание освобождения вх.буф.адаптера.
2C47:0218  B0E9      MOV    AL,E9   ;Передача в мышь команды E9h-получить
2C47:021A  E660      OUT    [60],AL ;информацию о состоянии устройства.
2C47:021C  E8F9FE    CALL   0118    ;Ожидание заполнения вых.буфера мыши.
2C47:021F  E8EAFE    CALL   010C    ;Ожидание заполн.вых.буфера адаптера.
2C47:0222  E460      IN     AL,[60] ;Чтение кода, полученного от мыши и
2C47:0224  E8FDFE    CALL   0124    ;запись его в буфер памяти.
2C47:0227  83FF04    CMP    DI,0004 ;Буфер памяти заполнен?
2C47:022A  7402      JZ     022E    ;Если "ДА", то чтение закончено, иначе -
2C47:022C  EBEE      JMP    021C    ;на новый цикл чтения байтов от мыши.
2C47:022E  E8CFFE    CALL   0100    ;Ожидание освобождения вх.буф.адаптера.
2C47:0231  B020      MOV    AL,20   ;Команда 20h-чтение командного байта
2C47:0233  E664      OUT    [64],AL ;передается адаптеру.
2C47:0235  E8D4FE    CALL   010C    ;Ожидание заполн.вых.буфера адаптера.
2C47:0238  E460      IN     AL,[60] ;Чтение командного байта.
2C47:023A  E8EEFE    CALL   012B    ;Организация программной задержки.
2C47:023D  4B        DEC    BX      ;Декремент счетчика циклов BX.
2C47:023E  75C3      JNZ    0203    ;Если BX не "нуль", то новый цикл прогр,
2C47:0240  FB        STI            ;иначе - разрешить прерывания, остановить
2C47:0241  CC        INT3           ;программу и передать управл. отладчику.
```

Листинги 1.20. Подпрограмм PSPPN.TXT, использующихся программой PSE9N, представленной в листинге 1.19.

Подпрограмма 1. Реализует циклы ожидания освобождения входного буфера адаптера клавиатуры/мыши.

```
2C47:0100  B9FFFF    MOV    CX,FFFF ;Установка счетчика таймаута.
2C47:0103  E464      IN     AL,[64] ;Чтение регистра состояния.
2C47:0105  A802      TEST   AL,02   ;Проверка: входной буфер пуст?
2C47:0107  E0FA      LOOPNZ 0103    ;Если не пуст, то новый цикл проверки.
2C47:0109  C3        RET            ;Возврат из подпрограммы.
2C47:010A  90        NOP            ;Пустые
2C47:010B  90        NOP            ;операции.
```

Подпрограмма 2. Реализует циклы ожидания заполнения выходного буфера адаптера клавиатуры/мыши.

```
2C47:010C  B9FFFF    MOV    CX,FFFF ;Установка счетчика таймаута.
2C47:010F  E464      IN     AL,[64] ;Чтение регистра состояния.
2C47:0111  A801      TEST   AL,01   ;Проверка: выходной буфер пуст?
2C47:0113  E1FA      LOOPZ  010F    ;Если пуст, то новый цикл проверки.
2C47:0115  C3        RET            ;Возврат из подпрограммы.
2C47:0116  90        NOP            ;Пустые
2C47:0117  90        NOP            ;операции.
```

Подпрограмма 3. Реализует циклы ожидания заполнения выходного буфера PS/2 Mouse в адаптере клавиатуры/мыши.

```
2C47:0118  B9FFFF    MOV    CX,FFFF
2C47:011B  E464      IN     AL,[64]
2C47:011D  A820      TEST   AL,20
2C47:011F  E1FA      LOOPZ  011B
2C47:0121  C3        RET
2C47:0122  90        NOP
2C47:0123  90        NOP
```

Подпрограмма 4. Реализует запись байта, передаваемого регистром AL, в память по адресу ES:[DI].

```
2C47:0124  268805    MOV    ES:[DI],AL ;Запись байта в память.
2C47:0127  47        INC    DI      ;Инкремент DI
2C47:0128  C3        RET            Возврат из подпрограммы.
```

```
2C47:0129  90        NOP            ;Пустые
2C47:012A  90        NOP            ;операции.
```

Подпрограмма 5. Реализует циклы программной задержки. На Pentium II / 500 мГц задержка составляет 2,1 мс.

```
2C47:012B  B90500    MOV    CX,0005 ;Число внешних циклов.
2C47:012E  51        PUSH   CX      ;Содержимое CX записывается в стек.
2C47:012F  B9FFFF    MOV    CX,FFFF ;Число внутренних циклов.
2C47:0132  E2FE      LOOP   0132    ;Зацикливание пока CX больше нуля.
2C47:0134  59        POP    CX      ;Извлечение CX из стека.
2C47:0135  E2F7      LOOP   012E    ;Зацикливание пока CX больше нуля.
2C47:0137  C3        RET            ;Возврат из подпрограммы.
2C47:0138  90        NOP            ;Пустые
2C47:0139  90        NOP            ;операции.
```

Подпрограмма 6. Реализует очистку выходного буфера адаптера клавиатуры/мыши.

```
2C47:013A  E464      IN     AL,[64] ;Чтение регистра состояния.
2C47:013C  A801      TEST   AL,01   ;Проверка: выходной буфер пуст?
2C47:013E  7404      JZ     0144    ;Если пуст, то на выход из подпрограммы.
2C47:0140  E460      IN     AL,[60] ;Очистка вых.буфера чтением порта 60h.
2C47:0142  EBF6      JMP    013A    ;На новый цикл проверки.
2C47:0144  C3        RET            ;Возврат из подпрограммы.
2C47:0145  90        NOP            ;Пустые
2C47:0146  90        NOP            ;операции
```

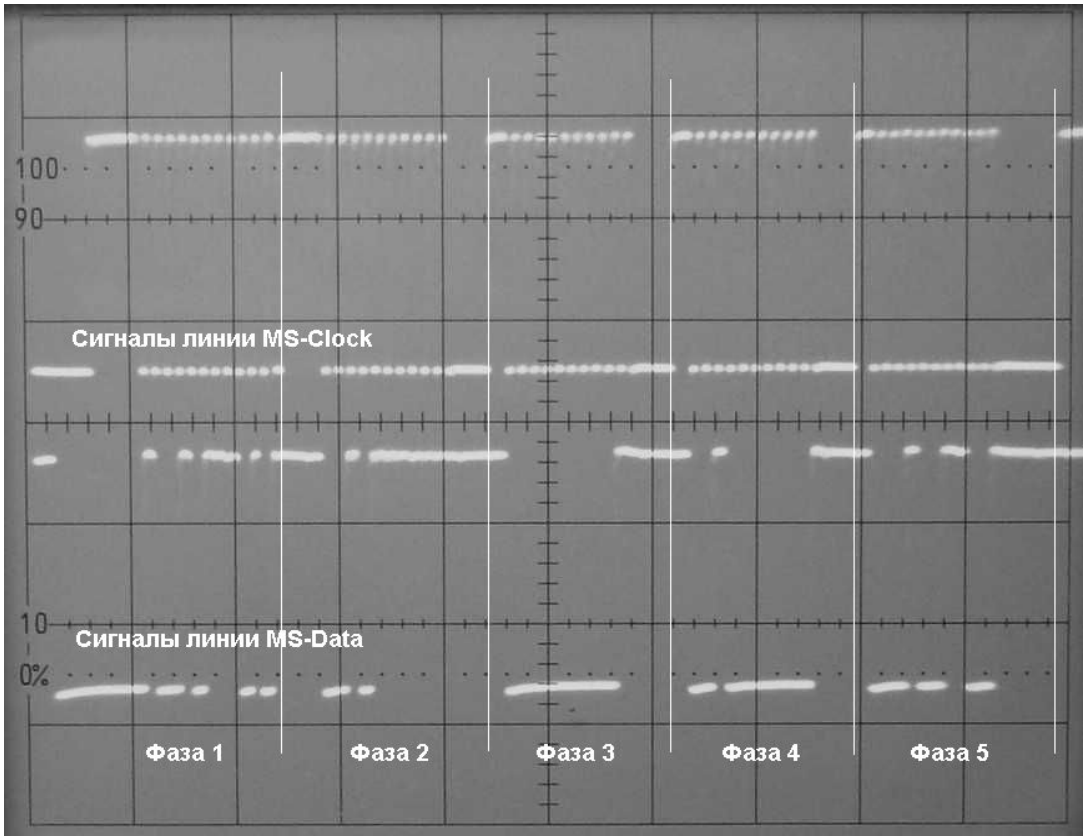

Рис. 1.29. Временные диаграммы сигналов интерфейса PS/2 мыши, зарегистрированные во время выполнения программы PSE9N.BIN.

На временной диаграмме можно выделить пять фаз активности интерфейса:

- фаза 1 отражает передачу в PS/2 мышь кода команды E9h;
- фаза 2 соответствует передаче PS/2 мышью в адаптер кода FAh (ACK) подтверждения получения и начала выполнения команды;
- в фазе 3 в адаптер передается первый байт состояния мыши 00h;
- в фазе 3 передается второй байт 02h;
- в фазе 3 - третий байт 64h.

Длительность пакета синхронизации каждой передачи составляет примерно 0,83 миллисекунды.

Команда EAh (Set Stream Mode) переключает мышь в потоковый режим: передача пакетов данных будет происходить при каждом изменении состояния устройства. Выполнение команды EAh можно реализовать с помощью программы, приведенной в листинге 1.17, изменив код E6 на EA в строке с адресом :011B.

Команда EBh (Read Data) позволяет получить из PS/2 мыши текущее значение счетчиков перемещения по осям X и Y, а также информацию о состоянии ее кнопок. После успешной передачи данных счетчики перемещения сбрасываются в нуль. Данные мышью передаются даже в том случае, если координаты устройства и состояние кнопок не изменялись. Если мышь находится в дистанционном режиме, то выполнение данной команды является единственным способом получения данных от нее. Для реализации выполнения данной команды можно воспользоваться программой PSE9N, представленной в листинге 1.19. Для этого в ней необходимо произвести изменения в двух командах:

- в команде MOV по адресу :0200h необходимо в регистр BX занести константу 0001h, т.е. программа должна выполняться один раз за один пуск;
- в команде MOV по адресу :0218 код E9h необходимо заменить кодом EBh команды "Чтение данных".

После выполнения программы в буфере памяти DS:0000h - DS:0003h будет находиться четыре байта, полученных от мыши. Первый байт является кодом подтверждения FAh, а три остальных интерпретируются в соответствие со стандартным форматом передач PS/2 Mouse (см. таблицу 1.22).

Команда ECh (Reset Wrap Mode) отключает эхо-режим, то есть служит для отмены команды EEh. Выполнение команды ECh можно реализовать с помощью программы, приведенной в листинге 1.17, изменив код E6 на EC в строке с адресом :011B.

Команда EEh (Set Wrap Mode) переводит мышь в эхо-режим. В эхо-режиме мышь отсылает обратно в компьютер каждый переданный ей байт данных (кроме ECh и FFh). Применяется данный режим для тестирования интерфейса мыши. Отменить его можно посылкой команды ECh или FFh. Пример использования команды EEh приведен в листинге 1.16. Выполнение команды EEh можно реализовать с помощью программы, приведенной в листинге 1.17, изменив код E6 на EE в строке с адресом :011B.

Команда F0h (Set Remote Mode) позволяет установить режим дистанционного управления: мышь будет передавать пакеты данных только по запросу со стороны компьютера (по команде EBh). Выполнение команды F0h можно реализовать с помощью программы, приведенной в листинге 1.17, изменив код E6 на F0 в строке с адресом :011B.

Команда F2h (Read Device Type) предназначена для определения типа подключенного координатного устройства. Мышь должна выдавать в ответ на эту команду однобайтный код 00h. Реализовать выполнение данной команды можно программой PSE9N, представленной в листинге 1.19. Для этого в ней необходимо произвести изменения в трех командах:

- в команде MOV по адресу :0200h необходимо в регистр BX занести константу 0001h, т.е. программа должна выполняться один раз за один пуск;
- в команде MOV по адресу :0218 код E9 необходимо заменить кодом F0 команды "Установить режим дистанционного управления";
- в команде CMP по адресу :0227 необходимо вместо константы 0004h нужно поставить константу 0002h, т.е в буфер памяти будет занесено два байта.

После выполнения программы в буфере памяти DS:0000h - DS:0001h будет находиться два байта, полученных от мыши. Первый байт является кодом подтверждения FAh, а второй - кодом идентификатора PS/2 мыши 00h.

Команда F3h (Set Sampling Rate) служит для установки частоты дискретизации. Следующий за ней байт данных должен содержать код, задающий максимальное количество пакетов данных, которое может быть передано за одну секунду (таблица 1.28).

Таблица 1.28. Коды для установки частоты дискретизации

Код	Частота дискретизации, пакет/с
0Ah	10
14h	20
28h	40
3Ch	60
50h	80
64h	100
C8h	200

Реализовать выполнение команды F3h можно с помощью программы PSE8.TXT, представленной в листинге 1.18. Для этого в команде MOV, находящейся по адресу :011Bh, вместо кода E8 необходимо поставить код F3.

Команда F4h (Enable) разрешает возобновить передачу данных, если мышь работает в потоковом режиме. Выполнение команды F4h можно реализовать с помощью программы, приведенной в листинге 1.17, изменив код E6 на F4 в строке с адресом :011Bh.

Команда F5h (Disable) используется в потоковом режиме, чтобы остановить передачу данных, инициированную мышью. Если мышь работает в потоковом режиме, то передачу данных нужно запрещать перед подачей любой команды, которая предусматривает ответ (передачу данных) со стороны мыши. После получения команды F5h мышь возвращает адаптеру код подтверждения FAh и запрещает передачу данных в потоковом режиме. Выполнение команды F5h можно реализовать с помощью программы, приведенной в листинге 1.17, изменив код E6 на F5 в строке с адресом :011B.

Команда F6h (Set Default) применяется, чтобы установить параметры работы по умолчанию. Получив эту команду мышь возвращает код подтверждения FAh и переводит свои параметры в то состояние, в которое они устанавливаются обычно после включения электропитания: масштаб - 1:1, разрешение - 4 отсчета на миллиметр, частота дискретизации - 100 выборок и потоковый режим. Выполнение команды F6h можно реализовать с помощью программы, приведенной в листинге 1.17, изменив код E6 на F6 в строке с адресом :011B.

Команда FEh (Resend) применяется при обнаружение любой ошибки передачи данных от мыши: получив эту команду, мышь выполняет повторную передачу предыдущего пакета данных. Однако - сигнал помехи, способный вызвать сбой при передаче данных от мыши, должен быть настолько мощным, что, скорее всего, нарушит и внутренние установки параметров встроенного в мышь микроконтроллера, поэтому при обнаружении ошибки лучше подать команду Reset и все параметры переустановить.

Команда FFh (Reset) используется при обнаружение серьезных сбоев в работе мыши. По этой команде мышь возвращает код FAh, выполняет процедуры сброса и самотестирования, которые длятся примерно 0,5 секунды. После этого мышь выдает идентификационную последовательность байтов: AAh, 00h (как при включение напряжения питания). Реализовать циклическое выполнение данной команды можно программой PSE9N, представленной в листинге 1.19. Для этого в ней необходимо произвести изменения в трех командах:

- в команде MOV по адресу :0200h необходимо в регистр BX занести константу 000Fh, т.е. программа должна выполняться шестнадцать раз за один пуск;
- в команде MOV по адресу :0218 код E9h необходимо заменить кодом FFh команды "Reset";
- в команде CMP по адресу :0227 необходимо вместо константы 0004h нужно поставить константу 0002h, т.е. в буфер памяти будет занесено три байта;
- в подпрограмме 2 по адресу :0113 команду LOOPZ надо заменить командой JZ, отключив тем самым тайм-аут (он установлен примерно на 60 миллисекунд, а ждать результат тестирования приходится примерно 500 миллисекунд).

После выполнения программы в буфере памяти DS:0000h - DS:0002h будет находиться три байта, полученных от мыши. Первый байт является кодом подтверждения FAh, второй - кодом нормального завершения теста AAh и третий - кодом идентификатора PS/2 мыши 00h.

1.3.4.3. Работа с PS/2 Mouse через функции BIOS прерывания INT 15h.

Для обслуживания мыши PS/2 имеется специальная функция C2h прерывания BIOS Int 15h - интерфейс координатного устройства. К сожалению, данный интерфейс носит вспомогательный характер - он служит только для управления параметрами мыши, а драйвер в его состав не входит.

После выполнения любой функции данной группы в случае успешного ее завершения флаг CF будет сброшен, в случае ошибки - установлен. Кроме того, в любом случае в регистр AH будет помещена дополнительная информация о результате операции:

- 00h - успешное завершение;
- 01h - недопустимый номер функции;
- 02h - недопустимое значение параметра;
- 03h - ошибка интерфейса;
- 04h - необходимо выполнить повторную передачу информации;
- 05h - драйвер устройства не установлен.

Рассмотрим набор подфункций интерфейса координатного устройства более подробно.

Прерывание INT 15h, функция C2h, подфункция 00h: разрешить или запретить работу мыши PS/2.

Подфункция позволяет разрешить или запретить работу мыши или другого координатного устройства, использующего интерфейс PS/2.

Перед вызовом прерывания INT 15h необходимо занести в регистры следующие значения:

- в AX - значение C200h;
- в BH - код команды (0 - запретить работу, 1 - разрешить работу).

После выполнения подфункции следует проверить флаг CF (флаг переноса). В случае возникновения ошибки флаг переноса будет установлен в 1 а в регистр AH будет помещен код ошибки. Ниже приведен пример использования подфункции 00h для запрещения работы мыши:

```
2C47:0100  B800C2    MOV    AX,C200 ;Помещение в AX номера функции.
2C47:0103  B700      MOV    BH,00   ;В AH заносим признак отключения мыши.
2C47:0105  CD15      INT    15      ;Вызов прерывания BIOS.
2C47:0107  CC        INT3           ;
```

При выполнении этого фрагмента программы без загруженного драйвера мыши приводит к ошибке с кодом 05h - драйвер устройства не установлен.

Прерывание INT 15h, функция C2h, подфункция 01h: произвести сброс устройства.

С помощью этой подфункции осуществляется перевод мыши или другого координатного устройства в исходное состояние.

Перед вызовом прерывания необходимо занести в регистр AX значение C201h.

Ниже приведен пример использования подфункции 01h для сброса PS/2 мыши.

```
2C47:0110  B801C2          MOV    AX,C201
2C47:0113  CD15            INT    15
2C47:0115  CC              INT3
```

После успешного выполнения этого примера в среде AFD параметры PS/2 мыши будут установлены в следующее состояние:

- частота передачи данных: 100 пакетов в секунду;
- разрешение: 4 отсчета на миллиметр;
- масштабирование 1:1;
- работа устройства запрещена.

Подфункция после завершения работы устанавливает флаг CF = 0, а регистр BX записывает следующие значения, переданные мышкой: старший байт BH - идентификатор устройства 00h, младший байт BL - значение AAh. Для выполнения этой подфункции не требуется загруженного драйвера.

Прерывание INT 15h, функция C2h, подфункция 02h: установить частоту дискретизации.

Подфункция позволяет изменить частоту дискретизации (частоту передачи пакетов данных от устройства к компьютеру).

Перед вызовом прерывания необходимо занести в регистры следующие значения:

- в AX - значение C202h;
- в BH - код частоты дискретизации (0 - 10 посылок/с, 1 - 20 посылок/с, 2 - 40 посылок/с, 3 - 60 посылок/с, 4 - 80 посылок/с, 5 - 100 посылок/с, 6 - 200 посылок/с).

Подфункция выполняется и без загруженного драйвера мыши. Пример ее использования приведен ниже.

```
2C47:0120  B802C2          MOV    AX,C202
2C47:0123  B704            MOV    BH,04
2C47:0125  CD15            INT    15
2C47:0127  CC              INT3
```

После выполнения примера в мыши будет установлена частота дискретизации 80 посылок в секунду. Подфункция возвращает признак CF = 0 и код ошибки в регистре AH = 00h.

Прерывание INT 15h, функция C2h, подфункция 03h: установить разрешение.

Подфункция позволяет регулировать чувствительность мыши к перемещению.

Перед вызовом прерывания необходимо занести в регистры следующие значения:

- в AX - значение C203h;
- в BH - код устанавливаемого разрешения (0 - 1 отсчет на мм, 1 - 2 отсчета на мм, 2 - 4 отсчета на мм, 3 - 8 отсчетов на мм).

Подфункция может выполняться без загруженного драйвера мыши. Пример ее использования приведен ниже.

```
2C47:0130  B803C2          MOV    AX,C203
2C47:0133  B702            MOV    BH,02
2C47:0135  CD15            INT    15
2C47:0137  CC              INT3
```

После установки в мыши разрешения 4 отсчета на мм подфункция возвращает значение CF=0 и код ошибки в регистре AH=00h.

Прерывание INT 15h, функция C2h, подфункция 04h: получить идентификатор устройства.

Подфункция предназначена для определения типа подключенного к компьютеру координатного устройства.

Перед вызовом прерывания необходимо занести в регистр AX значение C204h.

В случае успешного выполнения функции в регистре BH будет возвращен идентификационный код устройства.

Подфункция может выполняться без загруженного драйвера мыши. Пример ее использования приведен ниже.

```
2C47:0140  B804C2          MOV     AX,C204
2C47:0143  CD15            INT     15
2C47:0145  CC              INT3
```

Подфункция возвращает значение CF=0, код ошибки в регистре AH=00h и код устройства 00h (манипулятор PS/2 Mouse) в регистре BH.

Прерывание INT 15h, функция C2h, подфункция 05h: инициализировать интерфейс координатного устройства

Подфункция инициализирует интерфейс координатного устройства. После ее выполнения устройство мыши будет иметь следующие параметры: выключено, разрешение - 4 отсчета на миллиметр, частота дискретизации - 100 выборок в секунду, масштабирование - 1 : 1.

Перед вызовом прерывания необходимо занести в регистры следующие значения:

- в AX - значение C205h;
- в BH - размер пакета данных в байтах (от 1 до 8).

Подфункция может выполняться без загруженного драйвера мыши. Пример ее использования приведен ниже. В нем помимо стандартных значений параметров мыши устанавливается длина пакета 6 байт

```
2C47:0150  B805C2          MOV     AX,C205
2C47:0153  B706            MOV     BH,06
2C47:0155  CD15            INT     15
2C47:0157  CC              INT3
```

После успешного выполнения подфункция 05h возвращает значение CF=0, код ошибки в регистре AH=00h и код устройства 00h (манипулятор PS/2 Mouse) в регистре BH.

Прерывание INT 15h, функция C2h, подфункция 06h: получить состояние или установить масштаб.

Подфункция позволяет определить состояние устройства или установить режим масштабирования.

Перед вызовом прерывания необходимо занести в регистры следующие значения:

- в AX - значение C206h;
- в BH - код выполняемой операции (0 - определить состояние устройства, 1 - установить масштаб 1:1, 2 - установить масштаб 2:1).

После завершения выполнения операций с кодами 1 или 2 функция возвращает только общий код состояния координатного устройства в регистре AH. В случае успешного завершения операции с кодом 0 в регистрах будут размещены следующие значения:

- в AH - общий код состояния;
- в BL - байт состояния устройства;
- в CL - код текущего разрешения устройства (0 - 1 отсчет на мм, 1 - 2 отсчета на мм, 2 - 4 отсчета на мм, 3 - 8 отсчетов на мм);
- в DL - частоту дискретизации (0Ah - 10 посылок/с, 14h - 20 посылок/с, 28h - 40 посылок/с, 3Ch - 60 посылок/с, 50h - 80 посылок/с, 64h - 100 посылок/с, C8h - 200 посылок/с).

Разряды байта состояния устройства, возвращаемого в BL, имеют следующее значение:

- бит 0 - состояние правой кнопки (0 - отпущена, 1 - нажата);
- бит 1 - состояние средней кнопки (0 - отпущена, 1 - нажата);
- бит 2 - состояние левой кнопки (0 - отпущена, 1 - нажата);
- бит 3 - зарезервирован (всегда 0);
- бит 4 - масштабирование (0 - режим 1:1, 1 - режим 2:1);
- бит 5 - передача данных от устройства к компьютеру (0 - запрещена, 1 - разрешена);
- бит 6 - режим (0 - потоковый, 1 - дистанционного управления);
- бит 7 - зарезервирован (всегда 0).

Подфункция может выполняться без загруженного драйвера мыши. Пример ее использования, с кодом выполняемой операции 00h, приведен ниже.

```
2C47:0160  B806C2          MOV     AX,C206
2C47:0163  B700            MOV     BH,00
2C47:0165  CD15            INT     15
```

```
2C47:0167  CC              INT3
```

После успешного выполнения подфункция 06h с кодом операции 00h она возвращает значение CF=0, код ошибки в регистре AH=00h и код устройства 00h (манипулятор PS/2 Mouse) в регистре BH. Помимо этого она возвращает в регистре BL байт состояния устройства 00h, в регистре CL значение 02h - 4 отсчета на мм и в регистре DL код 64h - 100 посылок в секунду.

Прерывание INT 15h, функция C2h, подфункция 07h: установить адрес обработчика прерываний. Функция позволяет программисту установить свою собственную процедуру обработки прерываний от координатного устройства.

Перед вызовом прерывания необходимо занести в регистры следующие значения:

- в AX - значение C207h;
- в ES:BX - сегментный адрес и смещение пользовательского обработчика прерываний от координатного устройства. Для отмены пользовательского обработчика следует указать 0000h:0000h.

Подфункция может выполняться без загруженного драйвера мыши. Пример ее использования приведен ниже. В примере устанавливается адрес обработчика прерывания, расположенного в рабочей области отладчика начиная с адреса 2C47h:0200h.

```
2C47:0170  B8072C          MOV     AX,2C47
2C47:0173  8EC0            MOV     ES,AX
2C47:0175  BB0000          MOV     BX,0200
2C47:0178  B807C2          MOV     AX,C207
2C47:017B  CD15            INT     15
2C47:017D  CC              INT3
```

К сожалению, обработчик прерываний не входит в данный набор функций, и писать его должен сам программист.

1.3.4.4. Примеры программирования PS/2 Mouse.

В разделе приводятся примеры работы с PS/2 мышью непосредственно через адаптер клавиатуры/мыши. Программы и процедуры написаны на ассемблере. Приведенные программы рассчитаны на свободный доступ к памяти и аппаратуре, то есть должны выполняться в реальном режиме работы MS-DOS или в режиме эмуляции MS-DOS, который имеется в Windows 95/98. Ряд программных модулей сориентированы на выполнение в среде полноэкранного отладчика типа AFD.

1.3.4.4.1. Программы, выполняемые в среде DOS.

В листинге 1.20 приведены две подпрограммы общего назначения, которые будут использованы в программах, представленных ниже в листингах 1.21 и 1.22. Процедура Wait8042BufferEmpty выполняет операцию ожидания поступления сигнала очистки входного буфера адаптера клавиатуры (ее нужно вызывать перед началом операции записи любого байта во входной буфер адаптера по адресам 60h или 64h), а процедура WaitMouseData - операцию ожидания поступления данных от мыши в выходной буфер мыши [6].

Листинг 1.20. Процедуры общего назначения (в составе файла List5_03.inc)

```
CODESEG
;***********************************************
;*   ОЖИДАНИЕ ОЧИСТКИ ВХОДНОГО БУФЕРА I8042    *
;* При выходе из процедуры:                    *
;* флаг ZF установлен - нормальное завершение, *
;* флаг ZF сброшен - ошибка тайм-аута.         *
;***********************************************
proc Wait8042BufferEmpty near
        push    CX
        mov     CX,0FFFFh  ;задать число циклов ожидания
@@kb:
        in      AL,64h     ;получить статус
        test    AL,10b     ;буфер i8042 свободен?
        loopnz  @@kb       ;если нет, то цикл
        pop     CX
        ;Если при выходе из подпрограммы сброшен
        ;флаг ZF - ошибка
        ret              ;возврат в подпрограмму
endp Wait8042BufferEmpty
```

```
;*************************************
;* ОЖИДАНИЕ ПОСТУПЛЕНИЯ ДАННЫХ ОТ МЫШИ *
;*************************************
proc WaitMouseData near
        push    CX
        mov     CX,0FFFFh  ;задать число циклов ожидания
@@mouse:
        in      AL,64h     ;опросить регистр статуса
        test    AL,100000b ;данные поступили?
        loopz   @@mouse    ;если нет, то цикл
        pop     CX
        ;Если при выходе из подпрограммы установлен
        ;флаг ZF - ошибка
        ret
endp WaitMouseData
ENDS
```

Для корректной работы программ, приведенных в листингах 1.21 и 1.22 необходимо пользоваться еще одной процедурой общего назначения "Ожидание поступления байта в выходной буфер i8042". WaitOutputBufferI8042 (см. листинги 1.17 и 1.18 в разделе "Непосредственная работа с PS/2 Mouse").

```
;*************************************
;*Ожидание поступления байта в выходной *
;*буфер I8042                          *
;*************************************
```

```
pros WaitOutputBufferI8042 near
        push  cx
        mov   cx,FFFFh    ;задать число циклов ожибания
@@outb:
        in    al,64h      ;получить статус
        test  al,01b      ;выходной буфер i8042 заполнен?
        loopz @@outb      ;если нет, новый цикл проверки
        pop   cx
        ;если при выходе из подпрограммы
        ;установлен флаг ZF - ошибка
        ret               ;возврат из подпрограммы
endp WaitOutputBufferI8042
```

Эта процедура должна выполняться всегда перед чтением любой информации из выходного буфера адаптера. Так, в программах, приведенных в листингах 1.21 и .1.22, эту процедуру необходимо выполнять после передачи адаптеру команд, возвращающих результаты их выполнения в выходной буфер с адресом 60h, а также после поступления очередного байта от PS/2 мыши в выходной буфер мыши (после процедуры WaitMouseData). Как в первом, мак и во втором случаях байт помещается в выходной буфер адаптера не ранее, чем через 15 микросекунд после наступления любого из этих событий.

Следует также отметить, что в приведенных программах также не корректно производится выполнение двухбайтовых команд адаптера (команды 60h - загрузить новый командный байт). При их реализации необходимо проверять признак освобождения входного буфера как перед записью кода команды по адресу 64h, так и перед записью нового командного байта по адресу 60h.

Эти три процедуры должны быть оформлены как подключаемый файл (include-файл) с именем list5_03.inc.

В листинге 1.21 показан пример программы (SetPS2MouseParameters) [6], изменяющей характеристики работы мыши PS/2 (разрешение и частоту дискретизации). Как упоминалось выше, пакет данных, описывающий состояние мыши, состоит из трех байт: байта статуса, байта разрешения и байта частоты дискретизации. Старые и новые значения характеристик отображаются на экран в виде таблицы (в двоичном коде). В программе используются подключаемые файлы list1_02.inc, list1_03.inc, list1_04.inc и list5_03.inc. Описание трех первых файлов приведено в разделах 1.2.7.1.2 - 1.2.7.1.4, а четвертый представлен в листинге 1.20.

Для запуска программы пригоден любой AT-совместимый персональный компьютер с подключенной мышью PS/2-типа.

ПРИМЕЧАНИЕ ———————————————————————————————

Если в момент запуска программы-примера на компьютере уже установлен драйвер мыши PS/2, то после завершения работы программы драйвер сам восстановит нужные ему значения параметров.

Листинг 1.21. Программа, изменяющая характеристики работы мыши PS/2.

```
IDEAL
P386
LOCALS
MODEL MEDIUM

; Подключить файл мнемонических обозначений
; кодов управляющих клавиш и цветовых кодов
include "list1_03.inc"
; Подключить файл макросов
include "list1_04.inc"

SEGMENT sseg para stack 'STACK'
DB 400h DUP(?)
ENDS

DATASEG
; Текстовые сообщения
Txt1 DB YELLOW,0,18
     DB "Управление параметрами работы мыши типа PS/2",0
     DB LIGHTGREEN,10,0,"Состояние",0
     DB LIGHTGREEN,10,16,"Статус",0
     DB LIGHTGREEN,10,32,"Разрешение",0
     DB LIGHTGREEN,10,48,"Дискретизация",0
     DB LIGHTCYAN,12,0,"Исходное:",0
     DB LIGHTCYAN,14,0,"Установленное:",0
     DB YELLOW,24,22
     DB "Нажмите любую клавишу на клавиатуре",0
Err1 DB 11,22,"Ошибка обмена данными с мышью PS/2!",0
ENDS

CODESEG
;*****************************
;* Основной модуль программы *
;*****************************
PROC SetPS2MouseParameters
        mov     AX,DGROUP
        mov     DS,AX
        mov     [CS:MainDataSeg],AX
; Установить текстовый режим и очистить экран
        mov     AX,3
        int     10h
; Скрыть курсор - убрать за нижнюю границу экрана
        mov     [ScreenString],25
        mov     [ScreenColumn],0
        call    SetCursorPosition
; Вывести текстовые сообщения на экран
        MShowColorText 8,Txt1

; ЗАПРЕТИТЬ ГЕНЕРАЦИЮ ПРЕРЫВАНИЯ IRQ12
        ; Прочитать внутренний регистр команд i8042
        call    Wait8042BufferEmpty
        mov     AL,020h
        out     64h,AL
        in      AL,60h
        ; Сохранить содержимое регистра команд в AH
        mov     AH,AL
        ; Записать внутренний регистр команд i8042
        call    Wait8042BufferEmpty
```

```
        mov     AL,060h
        out     64h,AL
        ; Запретить сигнал прерывания от мыши
        mov     AL,AH
        and     AL,11111101b
        out     60h,AL
; ПРОЧИТАТЬ ИСХОДНОЕ СОСТОЯНИЕ МЫШИ
; Установить белый цвет символов и черный фон
        mov     [TextColorAndBackground],0Fh
; Установить начальную позицию для вывода данных на экран
        mov     [ScreenString],12
        mov     [ScreenColumn],16
; Послать команду считывания состояния мыши
        call    Wait8042BufferEmpty
        mov     AL,0D4h ;послать мыши байт данных
        out     64h,AL
        call    Wait8042BufferEmpty
        mov     AL,0E9h ;команда считывания состояния
        out     60h,AL
; Получить от мыши код подтверждения приема команды
        call    WaitMouseData
        jz      @@DataInputError ;данные не поступили
        in      AL,60h
        ; Получено подтверждение приема команды?
        cmp     AL,0FAh
        jnz     @@DataInputError ;нет подтверждения приема
; Принять информацию от мыши
        ; Получить первый байт состояния
        call    WaitMouseData
        jz      @@DataInputError
        in      AL,60h
        call    ShowBinByte
        add     [ScreenColumn],8
        ; Получить второй байт состояния
        call    WaitMouseData
        jz      @@DataInputError
        in      AL,60h
        call    ShowBinByte
        add     [ScreenColumn],8
        ; Получить третий байт состояния
        call    WaitMouseData
        jz      @@DataInputError
        in      AL,60h
        call    ShowBinByte
        add     [ScreenColumn],8
; УСТАНОВИТЬ РАЗРЕШЕНИЕ
; Послать команду установки разрешения
        call    Wait8042BufferEmpty
        mov     AL,0D4h ;послать мыши байт данных
        out     64h,AL
        call    Wait8042BufferEmpty
        mov     AL,0E8h ;команда установки разрешения
        out     60h,AL
; Получить от мыши код подтверждения приема команды
        call    WaitMouseData
        jz      @@DataInputError ;данные не поступили
        in      AL,60h
        ; Получено подтверждение приема команды?
        cmp     AL,0FAh

        jnz     @@DataInputError ;нет подтверждения
; Послать код разрешения
        call    Wait8042BufferEmpty
        mov     AL,0D4h
        out     64h,AL
        call    Wait8042BufferEmpty
        mov     AL,1    ;код разрешения 1 точка/мм
        out     60h,AL
; Получить код подтверждения приема команды
        call    WaitMouseData
        jz      @@DataInputError ;данные не поступили
        in      AL,60h
        ; Получено подтверждение приема команды?
        cmp     AL,0FAh
        jnz     @@DataInputError ;нет подтверждения

; УСТАНОВИТЬ ЧАСТОТУ ДИСКРЕТИЗАЦИИ
; Послать команду установки частоты дискретизации
        call    Wait8042BufferEmpty
        mov     AL,0D4h ;послать мыши байт данных
        out     64h,AL
        call    Wait8042BufferEmpty
        mov     AL,0F3h ;установить частоту
        out     60h,AL
; Получить код подтверждения приема команды
        call    WaitMouseData
        jz      @@DataInputError ;данные не поступили
        in      AL,60h
        ; Получено подтверждение приема команды?
        cmp     AL,0FAh
        jnz     @@DataInputError ;нет подтверждения
; Послать код частоты дискретизации
        call    Wait8042BufferEmpty
        mov     AL,0D4h
        out     64h,AL
        call    Wait8042BufferEmpty
        mov     AL,0C8h ;выдавать 200 пакетов в секунду
        out     60h,AL
; Получить код подтверждения приема команды
        call    WaitMouseData
        jz      @@DataInputError ;данные не поступили
        in      AL,60h
        ; Получено подтверждение приема команды?
        cmp     AL,0FAh
        jnz     @@DataInputError ;нет подтверждения

; ПРОЧИТАТЬ НОВОЕ СОСТОЯНИЕ МЫШИ
        mov     [ScreenString],14
        mov     [ScreenColumn],16
; Послать команду считывания состояния мыши
        call    Wait8042BufferEmpty
        mov     AL,0D4h
        out     64h,AL
        call    Wait8042BufferEmpty
        mov     AL,0E9h
        out     60h,AL
; Получить код подтверждения приема команды
        call    WaitMouseData
        jz      @@DataInputError ;данные не поступили
        in      AL,60h
        ; Получено подтверждение приема команды?
        cmp     AL,0FAh
        jnz     @@DataInputError ;нет подтверждения
; Принять информацию от мыши
        ; Получить первый байт состояния
        call    WaitMouseData
        jz      @@DataInputError
        in      AL,60h
        call    ShowBinByte
        add     [ScreenColumn],8
        ; Получить второй байт состояния
        call    WaitMouseData
        jz      @@DataInputError
        in      AL,60h
        call    ShowBinByte
        add     [ScreenColumn],8
        ; Получить третий байт состояния
        call    WaitMouseData
        jz      @@DataInputError
        in      AL,60h
        call    ShowBinByte
        add     [ScreenColumn],8

; РАЗРЕШИТЬ ГЕНЕРАЦИЮ ПРЕРЫВАНИЯ IRQ12
        ; Прочитать внутренний регистр команд i8042
        call    Wait8042BufferEmpty
        mov     AL,020h
        out     64h,AL
        in      AL,60h
```

```
        ; Записать внутренний регистр команд i8042
        mov     AH,AL
        call    Wait8042BufferEmpty
        mov     AL,060h
        out     64h,AL
        mov     AL,AH
        or      AL,10b ;разрешить прерывание от мыши
        out     60h,AL

        jmp short @@End

; Ожидать нажатия любой клавиши на клавиатуре
@@End:  call    GetChar
; Переустановить текстовый режим и очистить экран
        mov     AX,3
        int     10h
; Выход в DOS
        mov     AH,4Ch
        int     21h

; Обработка ошибок
@@DataInputError:
        MFatalError Err1
ENDP SetPS2MouseParameters
ENDS

; Подключить процедуры вывода данных на экран
include "list1_02.inc"
; Подключить процедуры очистки выходного буфера
; контроллера клавиатуры и ожидания поступления
; данных от мыши
include "list5_03.inc"

END
```

В листинге 1.22 приведена программа PS2MouseStart, устанавливающая новый (пользовательский) обработчик прерывания от мыши PS/2 [6]. Прежде чем установить обработчик при помощи процедуры SetPS2MouseInterrupt, основная программа посылает мыши команду разрешения передачи данных, одновременно проверяя тем самым наличие мыши и ее исправность (если мышь не отвечает кодом подтверждения, значит, она не подключена).

Программа обработки прерывания PS2MouseInterrupt принимает пакеты данных от мыши и после приема каждого трехбайтного пакета определяет состояние кнопок и приращение значений координат курсора. Для вывода курсора на экран используется процедура ShowNewMouseCursorPosition, которая инвертирует байт атрибута символа в позиции курсора (запоминая предварительно старое значение атрибута и восстанавливая его при перемещении курсора). Основная программа контролирует (в цикле) состояние левой кнопки мыши и завершает свою работу при нажатии на нее, восстанавливая предварительно старый вектор прерывания при помощи процедуры Restore01dPS2MouseInterrupt.

В этой программе используются те же подключаемые файлы, что и в предыдущей программе.

Для запуска программы пригоден любой AT-совместимый персональный компьютер с подключенной мышью PS/2-типа.

Листинг 1.22. Установка нового обработчика прерывания мыши PS/2, отображающего указатель в текстовом режиме инверсией атрибута символа.

```
IDEAL
P386
LOCALS
MODEL MEDIUM

; Подключить файл мнемонических обозначений
; кодов управляющих клавиш и цветовых кодов
include "list1_03.inc"
; Подключить файл макросов
include "list1_04.inc"

SEGMENT sseg para stack 'STACK'
DB 400h DUP(?)
ENDS

ScreenLength equ 80
ScreenWidth  equ 25

DATASEG
; Область сохранения старого вектора прерывания мыши
OldPS2MouseInterruptOffset  DW ?
OldPS2MouseInterruptSegment DW ?
; Номер принимаемого от мыши байта
MouseByteNumber DB 0
; Трехбайтовая структура данных, передаваемая мышью
FirstByte       DB 0
SecondByte      DB 0
ThirdByte       DB 0
; Текущее состояние кнопок
ButtonsStatus   DB 0
; Текущие координаты курсора мыши
XCoordinate     DW 0
YCoordinate     DW 0
; Предыдущая позиция курсора мыши
OldXCoordinate  DW 0
OldYCoordinate  DW 0
; Старое значение фона символа
OldCharBackground DB 0Fh
; Текстовые сообщения
Txt1 DB LIGHTGREEN,0,28,"Тест для мыши PS/2-типа",0
     DB WHITE,12,14,"Отображение курсора "
     DB "мыши инверсией атрибута символа",0
     DB YELLOW,24,21
     DB "Для выхода нажмите левую клавишу мыши",0
Err1 DB 11,22,"Ошибка обмена данными с мышью PS/2!",0
ENDS

CODESEG
;*****************************
;* Основной модуль программы *
;*****************************
PROC PS2MouseStart
        mov     AX,DGROUP
        mov     DS,AX
        mov     [CS:MainDataSeg],AX
; Установить текстовый режим и очистить экран
        mov     AX,3
        int     10h
; Скрыть текстовый курсор - убрать за
; нижнюю границу экрана
        mov     [ScreenString],25
        mov     [ScreenColumn],0
        call    SetCursorPosition
; Вывести текстовые сообщения на экран
        MShowColorText 3,Txt1
```

```
; ЗАПРЕТИТЬ ГЕНЕРАЦИЮ ПРЕРЫВАНИЯ IRQ12
        ; Прочитать внутренний регистр команд i8042
        call    Wait8042BufferEmpty
        mov     AL,020h
        out     64h,AL
        in      AL,60h
        ; Сохранить содержимое регистра команд в AH
        mov     AH,AL
        ; Записать внутренний регистр команд i8042
        call    Wait8042BufferEmpty
        mov     AL,060h
        out     64h,AL
        ; Запретить сигнал прерывания от мыши
        mov     AL,AH
        and     AL,11111101b
        out     60h,AL
; РАЗРЕШИТЬ ПЕРЕДАЧУ ДАННЫХ ОТ МЫШИ
        call    Wait8042BufferEmpty
        mov     AL,0D4h ;послать мыши байт данных
        out     64h,AL
        call    Wait8042BufferEmpty
        mov     AL,0F4h ;разрешить передачу данных
        out     60h,AL
; Получить код подтверждения приема команды
        call    WaitMouseData
        jz      @@DataInputError ;данные не поступили
        in      AL,60h
        ; Получено подтверждение приема команды?
        cmp     AL,0FAh
        jnz     @@DataInputError ;нет подтверждения
; РАЗРЕШИТЬ ГЕНЕРАЦИЮ ПРЕРЫВАНИЯ IRQ12
; Прочитать внутренний регистр команд i8042
        call    Wait8042BufferEmpty
        mov     AL,020h
        out     64h,AL
        in      AL,60h
; Записать внутренний регистр команд i8042
        mov     AH,AL
        call    Wait8042BufferEmpty
        mov     AL,060h
        out     64h,AL
; Разрешить сигнал прерывания от мыши
        mov     AL,AH
        or      AL,10b
        out     60h,AL
; УСТАНОВИТЬ НОВЫЙ ОБРАБОТЧИК ПРЕРЫВАНИЯ
; Обнулить позицию курсора мыши
        mov     [XCoordinate],0
        mov     [YCoordinate],0
        mov     [OldXCoordinate],0
        mov     [OldYCoordinate],0
; Отобразить курсор мыши первый раз
        call    ShowNewMouseCursorPosition
; Установить новый обработчик прерывания
        call    SetPS2MouseInterrupt
; Цикл, пока не нажата левая кнопка
@@Next: test    [ButtonsStatus],1b
        jz      @@Next
        jmp short @@End

; ЗАВЕРШЕНИЕ РАБОТЫ ПРОГРАММЫ
; Восстановить прежний обработчик прерывания
@@End:  call    RestoreOldPS2MouseInterrupt
; Переустановить текстовый режим и очистить экран
        mov     AX,3
        int     10h
; Выход в DOS
        mov     AH,4Ch
        int     21h

; Обработка ошибок
@@DataInputError:
        MFatalError Err1

ENDP PS2MouseStart

;*************************************
;* ОБРАБОТЧИК ПРЕРЫВАНИЯ ОТ МЫШИ PS/2 *
;*************************************
proc PS2MouseInterrupt far
        sti     ;разрешить маскируемые прерывания
        pusha
        push    DS
        mov     AX,[CS:MainDataSeg]
        mov     DS,AX
        call    Wait8042BufferEmpty ;очистка буфера
        in      AL,60h              ;получить скан-код

; Выбирать порядковый номер принимаемого байта
        cmp     [MouseByteNumber],0
        je      @@SaveFirstByte
        cmp     [MouseByteNumber],1
        je      @@SaveSecondByte
        cmp     [MouseByteNumber],2
        je      @@SaveThirdByte
        jmp     @@Error
; Записать первый байт посылки
@@SaveFirstByte:
        test    AL,1000b    ;первый байт посылки?
        jz      @@Error     ;сбой синхронизации
        mov     [FirstByte],AL
        inc     [MouseByteNumber]
        jmp     @@EndMouseInterrupt
; Записать второй байт посылки
@@SaveSecondByte:
        mov     [SecondByte],AL
        inc     [MouseByteNumber]
        jmp     @@EndMouseInterrupt
; Записать третий байт посылки
@@SaveThirdByte:
        mov     [ThirdByte],AL
        mov     [MouseByteNumber],0
; (пакет данных от мыши принят полностью)

; Записать новое значение байта состояния кнопок
        mov     AL,[FirstByte]
        and     AL,0111b
        mov     [ButtonsStatus],AL
; Вычислить новую X-координату курсора
        ; Занести в AX перемещение по X
        mov     AH,0   ;дублируем знак во все разряды AH
        mov     AL,[FirstByte]
        test    AL,10000b
        jz      @@M0
        mov     AH,0FFh
        ; Занести в AL младший байт
@@M0:   mov     AL,[SecondByte]
        ; Вычислить новое значение координаты
        ; курсора по X

        add     AX,[XCoordinate]
        cmp     AX,0
        jge     @@M1
        mov     AX,0
        jmp short @@M2
@@M1:   cmp     AX,ScreenLength
        jl      @@M2
        mov     AX,ScreenLength-1
@@M2:   mov     [XCoordinate],AX

; Вычисляем новую Y-координату курсора
        ; Занести в AX перемещение по Y
        mov     AH,0   ;дублируем знак во все разряды AH
        mov     AL,[FirstByte]
        test    AL,100000b
        jz      @@M3
```

```
        mov     AH,0FFh
        ; Занести в AL младший байт
@@M3:   mov     AL,[ThirdByte]
        ; Вычислить новое значение координаты курсора
        ; по Y (Y-координата мыши PS/2 направлена
        ; противоположно экранной)
        neg     AX
        add     AX,[YCoordinate]
        cmp     AX,0
        jge     @@M4
        mov     AX,0
        jmp short @@M5
@@M4:   cmp     AX,ScreenWidth
        jl      @@M5
        mov     AX,ScreenWidth-1
@@M5:   mov     [YCoordinate],AX

; Показать курсор в новой позиции
        call    ShowNewMouseCursorPosition
        jmp short @@EndMouseInterrupt

; Обнаружен сбой в порядке передачи информации от мыши
@@Error:
        mov   [MouseByteNumber],0
; Нормальное завершение прерывания
@@EndMouseInterrupt:
        cli
        ; разрешение прерываний с меньшими приоритетами
        mov     AL,20h
        out     0A0h,AL ;команда EOI в i8059-2
        out     20h,AL  ;команда EOI в i8059-1
        pop     DS
        popa
        iret
endp PS2MouseInterrupt
;*****************************************
;* УСТАНОВИТЬ НОВЫЙ ВЕКТОР ПРЕРЫВАНИЯ МЫШИ *
;*****************************************
PROC SetPS2MouseInterrupt NEAR
        pusha
        push    ES
        mov     AX,0
        mov     ES,AX
; Запомнить прежний вектор обработчика
; прерывания мыши
        mov     AX,[ES:74h*4]
        mov     [OldPS2MouseInterruptOffset],AX
        mov     AX,[ES:74h*4+2]
        mov     [OldPS2MouseInterruptSegment],AX
; Установка нового вектора прерывания
        cli     ;запретить прерывания
        mov     AX,offset PS2MouseInterrupt
        mov     [ES:74h*4],AX
        mov     AX,CS
        mov     [ES:74h*4+2],AX
        sti     ;разрешить прерывания
; Размаскировать прерывание IRQ12
        in      AL,0A1h
        and     AL,11101111b
        jmp short $+2       ;задержка
        jmp short $+2       ;задержка
        out     0A1h,AL
        pop     ES
        popa
        ret
ENDP SetPS2MouseInterrupt
;*********************************************
;* ВОССТАНОВИТЬ СТАРЫЙ ВЕКТОР ПРЕРЫВАНИЯ МЫШИ *
;*********************************************
PROC RestoreOldPS2MouseInterrupt NEAR
        pusha
        push    ES
        mov     AX,0
        mov     ES,AX
; Восстановить прежний вектор обработчика прерывания
        cli
        mov     AX,[OldPS2MouseInterruptOffset]
        mov     [ES:74h*4],AX
        mov     AX,[OldPS2MouseInterruptSegment]
        mov     [ES:74h*4+2],AX
        sti
        pop     ES
        popa
        ret
ENDP RestoreOldPS2MouseInterrupt
;*****************************************
;* ОТОБРАЖЕНИЕ КУРСОРА МЫШИ ПУТЕМ ИНВЕРСИИ *
;*   АТРИБУТА СИМВОЛА В ПОЗИЦИИ КУРСОРА    *
;*****************************************
PROC ShowNewMouseCursorPosition NEAR
        pusha
        push    ES
; Настроить ES на текстовый буфер
        mov     AX,0B800h
        mov     ES,AX
; Вычислить адрес предыдущей позиции курсора мыши
        mov     AX,[OldYCoordinate]
        mov     DX,160
        mul     DX
        add     AX,[OldXCoordinate]
        add     AX,[OldXCoordinate]
        inc     AX
        ; Занести смещение в индексный регистр
        mov     DI,AX
        ; Восстановить исходный атрибут символа
        mov     AL,[OldCharBackground]
        mov     [ES:DI],AL
; Вычислить адрес новой позиции курсора мыши
        mov     AX,[YCoordinate]
        mov     DX,160
        mul     DX
        add     AX,[XCoordinate]
        add     AX,[XCoordinate]
        inc     AX
        ; Занести смещение в индексный регистр
        mov     DI,AX
        ; Сохранить значение атрибута символа
        mov     AL,[ES:DI]
        mov     [OldCharBackground],AL
        ; Инвертировать атрибут символа (кроме
        ; старшего разряда)
        xor     [byte ptr ES:DI],1111111b
        ; Запомнить координаты курсора
        mov     AX,[XCoordinate]
        mov     [OldXCoordinate],AX
        mov     AX,[YCoordinate]
        mov     [OldYCoordinate],AX
        pop     ES
        popa
ENDP ShowNewMouseCursorPosition
ENDS

; Подключить процедуры вывода данных на экран
include "list1_02.inc"
; Подключить процедуры очистки выходного буфера
; контроллера клавиатуры и ожидания поступления
; данных от мыши
include "list5_03.inc"

END
```

ПРИМЕЧАНИЕ

Для успешного запуска примеров из листингов 1.21 и 1.22 необходимо, чтобы к компьютеру была подключена мышь PS/2-типа. Кроме того, в BIOS SETUP должно быть разрешено использование мыши PS/2

и за мышью должно быть закреплено прерывание IRQ12.

1.3.4.4.2. Программные модули, выполняемые в среде отладчика AFD.

Ниже приводятся программные модули, реализующие прием информации от PS/2 мыши обработчиком прерывания INT 74h, по запросам, формируемым адаптером PS/2 мыши по линии IRQ 12 (прерывание от PS/2 мыши).

Первый программный модуль IITPS.txt, представленный в листинге 1.23, выполняет основные действия по подготовке подсистемы PS/2 мыши к передаче данных мышью в потоковом режиме, к приему и размещению этих данных в буфере памяти обработчиком прерывания INT 74h, а также реализует циклы ожидания заполнения буфера данными. После заполнения буфера памяти данными работа PS/2 мыши запрещается и сбрасывается разряд D5 регистра состояния адаптера (признак: - выходной буфер мыши заполнен) и управление возвращается отладчику, который отображает в своем окне памяти содержимое буфера, заполненного обработчиком прерывания, начиная с адреса DS:0000h (128 баит). Краткая информация по правилам работы с подсистемой аппаратных прерываний представлена в приложении.

В среде отладчика программный модуль IITPS.bin загружается с адреса DS:0100h. (В исходном состоянии в отладчике AFD в сегментных регистрах CS, DS, ES, SS, HS и FS содержатся одинаковые значения, но загрузка программных модулей командой L ведется в сегмент данных DS). Запуск на выполнение модуля IITPS.bin осуществляется командой G 100 отладчика.

Листинг 1.23. Программный модуль IITPS.bin.

```
   ;Настройка ведомого контроллера прерываний.
2C47:0100  E4A1      IN     AL,[A1] ;Снятие маскирования запроса IRQ12,
2C47:0102  24EF      AND    AL,EF   ;поступающего в разряд D4
2C47:0104  EB00      JMP    0106    ;регистра запросов
2C47:0106  EB00      JMP    0108    ;ведомого контроллера
2C47:0108  E6A1      OUT    [A1],AL ;прерываний.

   ;Формирование вектора прерывания INT74h по адресу 0000:01D0h.
2C47:010A  B80000    MOV    AX,0000 ;В сегментный регистр ES
2C47:010D  8EC0      MOV    ES,AX   ;заносится код 0000h.
2C47:010F  B80002    MOV    AX,0200 ;В младшие два байта вектора заносится
2C47:0112  26A3D001  MOV    ES:[01D0],AX ;смещение 0200h
2C47:0116  8CC8      MOV    AX,CS   ;В старшие два байта вектора заносится
2C47:0118  26A3D201  MOV    ES:[01D2],AX ;содержимое сегментного регистра CS.

   ;Передача в PS/2 мышь команды f6h "Set Default" - установить параметры
   ;мыши, используемые по умолчанию (масштаб - 1:1, разрешение - 4 отсчета на
   ;миллиметр, частота дискретизации - 100 выборок и потоковый режим).
2C47:011C  FA        CLI            ;Запрет обработки прерываний.
2C47:011D  E464      IN     AL,[64] ;Проверка и
2C47:011F  A801      TEST   AL,01   ;очистка выходного
2C47:0121  7404      JZ     0127    ;буфера
2C47:0123  E460      IN     AL,[60] ;адаптера
2C47:0125  EBF6      JMP    011D    ;клавиатуры и PS/2 мыши.
2C47:0127  E464      IN     AL,[64] ;Проверка: входной буфер
2C47:0129  A802      TEST   AL,02   ;адаптера пуст?
2C47:012B  75FA      JNZ    0127    ;если нет, то новый цикл проверки.
2C47:012D  B0D4      MOV    AL,D4   ;Код команды "Передать мыши байт данных"
2C47:012F  E664      OUT    [64],AL ;передается в адаптер.
2C47:0131  E464      IN     AL,[64] ;Проверка: входной буфер
2C47:0133  A802      TEST   AL,02   ;адаптера пуст?
2C47:0135  75FA      JNZ    0131    ;если нет, то новый цикл проверки.
2C47:0137  B0F6      MOV    AL,F6   ;Код команды "Set Default"
2C47:0139  E660      OUT    [60],AL ;передается в PS/2 мышь.
2C47:013B  E464      IN     AL,[64] ;Проверка: выходной буфер
2C47:013D  A820      TEST   AL,20   ;мыши полон? Если пуст,
2C47:013F  74FA      JZ     013B    ;то на новый цикл проверки.
2C47:0141  E464      IN     AL,[64] ;Проверка: выходной буфер
2C47:0143  A801      TEST   AL,01   ;адаптера полон? Если пуст,
2C47:0145  74FA      JZ     0141    ;то на новый цикл проверки.
2C47:0147  E460      IN     AL,[60] ;Чтение кода ответа мыши и помещение
2C47:0149  88C4      MOV    AH,AL   ;его в регистр AH.

   Передача в PS/2 мышь команды F4h "Enable" - разрешить передачу данных от
   мыши, работающей в потоковом режиме.
2C47:014B  B0D4      MOV    AL,D4   ;Код команды "Передать мыши байт данных"
```

```
2C47:014D  E664      OUT    [64],AL ;передается в адаптер.
2C47:014F  E464      IN     AL,[64] ;Проверка: входной буфер
2C47:0151  A802      TEST   AL,02   ;адаптера пуст?
2C47:0153  75FA      JNZ    014F    ;если нет, то новый цикл проверки.
2C47:0155  B0F4      MOV    AL,F4   ;Код команды "Enable"
2C47:0157  E660      OUT    [60],AL ;передается в PS/2 мышь.
2C47:0159  E464      IN     AL,[64] ;Проверка: выходной буфер
2C47:015B  A820      TEST   AL,20   ;мыши полон? Если пуст,
2C47:015D  74FA      JZ     0159    ;то на новый цикл проверки.
2C47:015F  E464      IN     AL,[64] ;Проверка: выходной буфер
2C47:0161  A801      TEST   AL,01   ;адаптера полон? Если пуст,
2C47:0163  74FA      JZ     015F    ;то на новый цикл проверки.
2C47:0165  E460      IN     AL,[60] ;Чтение кода ответа мыши и помещение
2C47:0167  88C3      MOV    BL,AL   ;его в регистр BL.
   ;Установка в командном байте адаптера разрядов: D5 - разрешение работы ;с
   мышью и D1 - разрешение прерывания от мыши.
2C47:0169  B020      MOV    AL,20   ;Код команды "Прочитать текущий командный
2C47:016B  E664      OUT    [64],AL ;байт" передается в адаптер.
2C47:016D  E464      IN     AL,[64] ;Проверка: выходной буфер
2C47:016F  A801      TEST   AL,01   ;адаптера полон? Если пуст,
2C47:0171  74FA      JZ     016D    ;то на новый цикл проверки.
2C47:0173  E460      IN     AL,[60] ;Чтение командного байта, установка
2C47:0175  24DF      AND    AL,DF   ;в нем флага "Разрешение работы с мышью"
2C47:0177  0C02      OR     AL,02   ;и флага "Разрешение прерывания от мыши".
2C47:0179  88C7      MOV    BH,AL   ;и запоминание его в регистре BH.
2C47:017B  B060      MOV    AL,60   ;Код команды "Записать новый командный
2C47:017D  E664      OUT    [64],AL ;байт" передается в адаптер.
2C47:017F  E464      IN     AL,[64] ;Проверка: входной буфер
2C47:0181  A802      TEST   AL,02   ;адаптера пуст?
2C47:0183  75FA      JNZ    017F    ;если нет, то новый цикл проверки.
2C47:0185  88F8      MOV    AL,BH   ;Значение нового командного байта
2C47:0187  E660      OUT    [60],AL ;передается в адаптер.
   ;Организация циклов ожидания заполнения обработчиком прерывания INT74h
   ;буфера памяти размером 128 (80h) байт.
2C47:0189  B90000    MOV    CX,0000 ;Обнуление счетчика байт в буфере.
2C47:018C  FB        STI            ;Разрешение обработки прерываний.
2C47:018D  F6C180    TEST   CL,80   ;Проверка: буфер заполнен? (128 байт)
2C47:0190  74FB      JZ     018D    ;если нет, то на новый цикл проверки.
2C47:0192  90        NOP
2C47:0193  90        NOP
   ;Передача в PS/2 мышь команды F5h "Disable" - остановить передачу данных
   ;от мыши, работающей в потоковом режиме.
2C47:0194  50        PUSH   AX      ;Содержимое AX запоминается в стеке.
2C47:0195  B0D4      MOV    AL,D4   ;Код команды "Передать мыши байт данных"
2C47:0197  E664      OUT    [64],AL ;передается в адаптер.
2C47:0199  E464      IN     AL,[64] ;Проверка: входной буфер
2C47:019B  A802      TEST   AL,02   ;адаптера пуст?
2C47:019D  75FA      JNZ    0199    ;если нет, то новый цикл проверки.
2C47:019F  B0F5      MOV    AL,F5   ;Код команды "Disable"
2C47:01A1  E660      OUT    [60],AL ;передается в мышь.
2C47:01A3  E464      IN     AL,[64] ;Проверка: выходной буфер полон?
2C47:01A5  A801      TEST   AL,01   ;если нет, то
2C47:01A7  74FA      JZ     01A3    ;новый цикл проверки.
2C47:01A9  E460      IN     AL,[60] ;Чтение кода подтверждения мыши.
   ;Сброс в регистре состояния адаптера разряда D5 - выходной буфер мыши
   ;полон командой адаптера 20h - прочитать текущий командный байт.
2C47:01AB  B020      MOV    AL,20   ;Код команды 20h
2C47:01AD  E664      OUT    [64],AL ;передается в адаптер
2C47:01AF  E464      IN     AL,[64] ;Проверка: выходной буфер полон?
2C47:01B1  A801      TEST   AL,01   ;если нет, то
2C47:01B3  74FA      JZ     01AF    ;новый цикл проверки.
2C47:01B5  E460      IN     AL,[60] ;Чтение командного байта.
   ;Завершение работы программы.
```

```
2C47:01B7  58        POP    AX      ;Восстановление из стека AX.
2C47:01B8  CC        INT3           ;Остановка и выход в AFD.
```

В программу можно вставить команды запоминания предыдущего вектора прерывания:

```
26A1D001  MOV    AX,ES:[01D0]
3EA30003  MOV    DS:[0300],AX
26A1D201  MOV    AX,ES:[01D2]
3EA30203  MOV    DS:[0302],AX
```

и восстановления его после завершения работы программы:

```
3EA10003  MOV    AX,DS:[0300]
26A3D001  MOV    ES:[01D0],AX
3EA10203  MOV    AX,DS:[0302]
26A3D201  MOV    ES:[01D2],AX
```

Первые четыре команды заносятся перед установкой нового вектора прерывания, а вторые четыре - в конце программы перед командой `POP AX`.

Второй программный модуль (INTPS.txt), представленный в листинге 1.24, выполняет функции обработчика прерываний. Он формирует буфер памяти, осуществляет его заполнение данными, обрабатывая прерывание INT 74h, и формирует информацию о степени заполнения буфера. В среде отладчика обработчик INTPS.bin загружается с адреса DS:0200h.

Листинг 1.24. Обработчик прерывания INT 74h (программный модуль INTPS.bin).

```
2C47:0200  57        PUSH   DI      ;Сохранение в стеке
2C47:0201  06        PUSH   ES      ;значений регистров
2C47:0202  50        PUSH   AX      ;процессора:
2C47:0203  53        PUSH   BX      ;DI, ES,AX,BX.
2C47:0204  FA        CLI            ;Запрещение обработки прерываний.
2C47:0205  BB0000    MOV    BX,0000 ;Обнуление счетчика байтов пакета.
2C47:0208  8CD8      MOV    AX,DS   ;Установка сегмента
2C47:020A  8EC0      MOV    ES,AX   ;буфера памяти.
2C47:020C  E464      IN     AL,[64] ;Проверка наличия байта в выходном
2C47:020E  A801      TEST   AL,01   ;буфере адаптера: если буфер
2C47:0210  74FA      JZ     020C    ;пуст, то на новый цикл проверки.
2C47:0212  E460      IN     AL,[60] ;Чтение байта, поступившего из мыши.
2C47:0214  89CF      MOV    DI,CX   ;Позиционирование на адрес в буфере памяти
2C47:0216  268805    MOV    ES:[DI],AL    ;Запись принят. байта в буф. памяти.
2C47:0219  41        INC    CX      ;Инкремент счетчика записанных в буф.байт.
2C47:021A  FEC3      INC    BL      ;Инкремент счетчика байт пакета.
2C47:021C  80FB03    CMP    BL,03   ;Если меньше 03, то на чтение очередного
2C47:021F  75EB      JNZ    020C    ;байта.
2C47:0221  B020      MOV    AL,20   ;Сброс D5 регистра состояния
2C47:0223  E664      OUT    [64],AL ;передачей адаптеру команды "Прочитать
2C47:0225  E464      IN     AL,[64] ;текущий командный байт" и.
2C47:0227  A801      TEST   AL,01   ;ожидание поступления командного байта
2C47:0229  74FA      JZ     0225    ;в выходной буфер адаптера.
2C47:022B  E460      IN     AL,[60] ;Чтение командного байта.
2C47:022D  B020      MOV    AL,20   ;Пересылка команды EOI (завершение обра-
2C47:022F  E6A0      OUT    [A0],AL ;ботки прерывания) в ведомый и ведущий
2C47:0231  E620      OUT    [20],AL ;контроллеры прерываний.
2C47:0233  5B        POP    BX      ;Восстановление из стека
2C47:0234  58        POP    AX      ;значений регистров
2C47:0235  07        POP    ES      ;процессора:
2C47:0236  5F        POP    DI      ;BX, AX, ES,DI.
2C47:0237  FB        STI            ;Разрешение обработки прерываний.
2C47:0238  CF        IRET           ;Возврат в прерванную программу.
```

В данном разделе использовались материалы из [4 - 6, 13].

Контрольные вопросы

1. Дайте общую характеристику манипулятору PS/2 Mouse.
2. Охарактеризуйте два основных варианта PS/2 Mouse.
3. Охарактеризуйте компоненты PS/2 Mouse, представленные на рис. 1.21.
4. Дайте общую характеристику оптическому манипулятору мышь.
5. Охарактеризуйте основные компоненты оптической мыши, представленных на рис. 1.22 и 1.23.

6. Охарактеризуйте компоненты электрической схемы 3D PS/2 мыши, представленной на рис. 1.24.
7. Охарактеризуйте контрольные точки электрической схемы (рис. 1.24).
8. Охарактеризуйте компоненты электрической схемы 3D PS/2 мыши, представленной на рис. 1.25.
9. Охарактеризуйте контрольные точки электрической схемы (рис. 1.25).
10. Охарактеризуйте компоненты электрической схемы 3D USB/PS/2 оптической мыши, представленной на рис. 1.26.
11. Охарактеризуйте контрольные точки электрической схемы (рис. 1.26).
12. Кратко охарактеризуйте интерфейс PS/2 мыши.
13. Охарактеризуйте программный модуль, представленный в листинге 1.16.
14. Каковы отличия взаимодействия на канальном уровне системы с PS/2 мышью и с клавиатурой?
15. Какие режимы работы поддерживает PS/2 мышь?
16. Какие операции выполняет PS/2 мышь после включения питания или получения команды Reset?
17. Как может осуществляться управление режимами работы PS/2 мыши?
18. Дайте описание стандартного формата передачи данных PS/2 мыши.
19. Чем отличается формат передачи 3D PS/2 мыши и 5-кнопочной 3D PS/2 мыши от стандартного формата PS/2 мыши?
20. Кратко охарактеризуйте команды, выполняемые PS/2 мышью.
21. Дайте описание основных действий программы, реализующей передачу байта в PS/2 мышь.
22. Охарактеризуйте особенности чтения данных, поступающих от PS/2 мыши.
23. Охарактеризуйте программу, представленную в листинге 1.17.
24. Объясните действие команды E7h.
25. Охарактеризуйте программу, представленную в листинге 1.18.
26. Охарактеризуйте программу, представленную в листинге 1.19.
27. Объясните осциллограмму, представленную на рис. 1.29.
28. Охарактеризуйте команду EBh.
29. Охарактеризуйте команды ECh и EDh.
30. Охарактеризуйте команды F0h - F2h.
31. Охарактеризуйте команду F3h.
32. Охарактеризуйте команды F4h и F5h.
33. Охарактеризуйте команду F6h.
34. Охарактеризуйте команду FFh.
35. Охарактеризуйте функцию C3h прерывания BIOS INT 15h.
36. Охарактеризуйте подфункции 00h и 01h функции C2h.
37. Охарактеризуйте подфункции 02h и 03h функции C2h.
38. Охарактеризуйте подфункции 04h и 05h функции C2h.
39. Охарактеризуйте подфункции 06h функции C2h.
40. Охарактеризуйте подфункции 07h функции C2h.
41. Для чего используются процедуры, представленные в листинге 1.20?
42. Охарактеризуйте программу, представленную в листинге 1.21.
43. Охарактеризуйте программу, представленную в листинге 1.22.
44. Охарактеризуйте программный модуль, представленный в листинге 1.23.
45. Охарактеризуйте программный модуль, представленный в листинге 1.24.

2. Манипуляторы-указатели — мышь.

Мышь и трекбол называют указательными устройствами (pointing devices), поскольку с их помощью пользователь может задать компьютеру местоположение курсора и подать одну из нескольких команд. Способ управления координатным курсором для системы безразличен и определяется его сферой применения и личными пристрастиями пользователя.

Манипулятор *мышь* (mouse) передает в систему информацию о своем перемещении по плоскости и нажатии кнопок (двух или трех, а в современных моделях и больше). Обычная конструкция имеет свободно вращающийся массивный обрезиненный шарик в днище корпуса, передающий вращение на два координатных диска с фотоэлектрическими датчиками. Датчики для каждой координаты представляют собой две открытые оптопары (светодиод-фотодиод), в оптический канал которых входит вращающийся диск с прорезями. Оптопары датчиков могут оформляться в виде монолитных конструкций или быть просто отдельными элементами, установленными на печатной плате.

В мыши имеется *микроконтроллер,* который обрабатывает сигналы с датчиков и посылает в ПК информацию о перемещениях и состоянии кнопок. Один из первых вариантов мыши — *Bus Mouse* (шинная мышь) — содержал только датчики и кнопки, а обработка их сигналов производилась на специализированной плате адаптера, устанавливаемого в слот шины ISA (откуда и название «шинная мышь»).

Манипулятор *трекбол* (Track Ball — дословно «следящий шар»), по сути, представляет собой перевернутую мышь, шарик которой вращают пальцами. Иногда он встраивается в клавиатуру (чаще устанавливается на портативных компьютерах). Преимущество шара в том, что он не требует для работы свободной плоской поверхности, а может закрепляться зажимом на краю стола. Однако вращать шар пальцами нравится не всем (хотя при этом и можно добиться большей точности позиционирования).

Оптическая мышь (optical mouse), не имеет механических частей, подверженных загрязнению и износу. Первые модели оптических мышей ориентировались по лучам, отраженным от специального коврика с сетчатым рисунком. Теоретически это надежнее, но загрязнения и царапины на коврике приводят к неожиданным «прыжкам», наклон оси мыши относительно оси коврика сильно искажает отображение траектории движения. Современные оптические мыши имеют встроенную видеокамеру с процессором, обрабатывающим полученное изображение. Эта мышь не требует специального коврика и может функционировать на любой поверхности — оптические «неровности», за которые она может «зацепиться взглядом», есть практически везде. Первые оптические мыши из-за слабости графического процессора обладали низким быстродействием — слишком быстрые перемещения «сбивали их с толку». Современные оптические мыши имеют хорошие параметры: разрешение — 800 dpi, скорость — до 1 м/с, ускорение — до 10g.

3D-мышь помимо двух обычных координат перемещения позволяет задавать и третью — с дополнительного колесика, вращаемого пальцем. Это колесико приводит в движение трещотки, нажимающие кнопки-датчики или взаимодействует с дополнительной двойной оптопарой.

По *интерфейсу с компьютером* различают несколько видов манипуляторов типа мышь, из которых в PC-совместимых компьютерах используются следующие:

- Serial Mouse — мышь с интерфейсом RS-232C, подключаемая к COM-порту ПК. Интерфейс однонаправленный: данные передаются только от мыши, параметрами самой мыши управлять невозможно.
- PS/2-Mouse — мышь с двунаправленным интерфейсом, подключаемая к специальному интерфейсному порту системной платы.
- USB Mouse — мышь с интерфейсом USB, низкоскоростное (Low Speed, LS) устройство USB, с которым устанавливается двунаправленная связь. Мышь с интерфейсом USB удобно подключать к клавиатуре или монитору со встроенным USB-хабом (меньше проводов идет к системному блоку).
- Bluetooth Mouse — мышь с двунаправленным радио интерфейсом.

Для ряда применений (в основном игр) важна частота опроса мыши. Для интерфейса USB она может достигать 125 опросов в секунду (но не больше!), интерфейс PS/2 допускает и большие частоты (до 200 опросов в секунду). Манипулятор Serial Mouse по возможной частоте посылок самые медленные (ниже 30 посылок в секунду).

С интерфейсами Serial Mouse и PS/2-Mouse иногда возникают недоразумения. Хотя оба они последовательные, эти интерфейсы имеют существенные принципиальные различия в уровнях сигналов, способе синхронизации, частоте и формате посылок:

- В интерфейсе PS/2 используется однополярный сигнал с уровнями ТТЛ, питание мыши — однополярное с напряжением +5 В относительно шины GND. Сигнал в интерфейсе RS-232C мыши Serial Mouse двуполярный с уровнями срабатывания +3 В и -3 В, и для него требуется двуполярное (относительно шины GND) питание мыши.
- В интерфейсе PS/2 применяются две раздельные сигнальные линии, одна для передачи данных, другая для передачи сигналов синхронизации. В Serial Mouse реализован асинхронный способ передачи данных всего по одной линии.

Даже если не рассматривать частоты и форматы посылок, ясно, что прямой совместимости между этими интерфейсами быть не может. Тем не менее выпускаются и продаются переходники (пассивные!),

позволяющие выбрать способ подключения мыши. Эти переходники предназначены только для универсальных мышей, у которых встроенный контроллер по напряжению питания способен распознать, к какому интерфейсу его подключили, и установить соответствующий тип своего выходного интерфейса. Универсальные мыши, поддерживающие интерфейсы и Serial Mouse, и PS/2, не особо распространены, поэтому часто приходится слышать о неудачных попытках применения таких переходников к обычным устройствам Serial Mouse или PS/2-Mouse. Сейчас выпускаются универсальные мыши PS/2-USB (интерфейсы тоже разные!) с соответствующими пассивными переходниками.

Дополнительную путаницу вносят мыши для компьютера Macintosh, которые имеют разъем, с виду напоминающий разъем PS/2. Однако при ближайшем рассмотрении и неудачной попытке включения его в PC становится ясно, что разъемы эти разные, да и интерфейс совершенно иной.

У современных мышей настраивается много параметров, доступных при установке «родного» драйвера. Основные параметры, отвечающие темпераменту пользователя, — время двойного щелчка и чувствительность.

В блокнотных ПК в качестве манипулятора может использоваться сенсорная панель (touch pad), чувствительная к прикосновению и перемещению пальца. Применяются и миниатюрные манипуляторы в виде кнопки, чувствительной к давлению в разных направлениях. К компьютеру может быть одновременно подключено несколько устройств-указателей (например, сенсорная панель и мышь или пара мышей). Операционные системы без специальной программной поддержки сообщения от всех указателей собирают в единый поток для управления единственным курсором на экране.

Чаще всего неисправности мыши связаны с внутренним переломом проводов около корпуса, что легко исправить, вырезав износившийся кусочек провода. Также часто мышь плохо работает из-за загрязнения шарика или валиков датчиков. Если резиновый шарик или валики датчиков загрязняются, мышь перестает распознавать движение. Для сохранения чистоты желательно пользоваться ковриком для мыши. Кроме того, шарик и валики периодически следует чистить — лучше всего протирать их тампоном, смоченным спиртом. Для чистки механики у мыши обычно имеется съемное «брюшко», снять которое можно его поворотом или сдвигом согласно указующей стрелке. Надежность распознавания движения мыши зависит и от расположения датчиков — взаимного и относительно колесика с прорезями. Некачественные оптопары чувствительны и к внешнему освещению: бывает, что разобранная мышь работает нормально, а с установленной крышкой — нет. Иногда достаточно слегка сблизить или, наоборот, раздвинуть светодиод и фотодиод, и мышь начинает работать нормально. «Дребезжащие» кнопки ремонту практически не подлежат — их приходится заменять.

Существуют варианты *беспроводных мышей* (cordless mouse): мышь с аккумуляторным питанием связана с интерфейсным блоком по инфракрасному каналу или по радиоканалу. Аналогично может подключаться и клавиатура. Интерфейсный блок подключается к компьютеру одним из стандартных «мышиных» интерфейсов: PS/2 или USB (возможно и через COM-порт). Беспроводная связь удобна, например, при проведении презентаций — докладчик не привязан к компьютеру (и видео проектору) проводами. Более элегантное решение — мышь (и клавиатура) с интерфейсом Bluetooth при наличии встроенного интерфейса Bluetooth в ПК; при этом лишних проводов и блоков нет.

Беспроводные клавиатуры с радио интерфейсом — «находка для шпиона»: весь текст, набираемый пользователем, а также имена и пароли могут быть легко перехвачены через радиоэфир. Поток данных от мыши особой ценности не представляет. Канал инфракрасной связи в плане конфиденциальности контролируется легче («шпион» должен быть виден), но менее удобен.

Некоторые беспроводные мыши имеют специфический дефект: при работе с ресурсоемкими приложениями они теряют точность наведения курсора (не удается с первого раза попасть в нужную точку). Это можно объяснить упрощенной организацией беспроводного канала связи: мышь рапортует об изменении своего состояния с определенным периодом. Эти рапорты получает интерфейсный блок и передает их в ПК по опросу, организованному интерфейсом подключения. Если беспроводной интерфейс однонаправленный (что, естественно, проще и дешевле), то в канале беспроводной связи невозможно организовать управление потоком (или квитирование). В этом случае, если из-за повышенной загрузки центрального процессора рапорт от мыши не будет своевременно передан драйверу, очередной рапорт (с указанием очередного изменения координат) потеряется. Чтобы таких неприятностей не было, нужен либо двунаправленный беспроводной канал связи, либо более сложный алгоритм работы контроллера в интерфейсном блоке. В последнем варианте контроллер должен подсчитывать текущие координаты мыши по приходящим рапортам и сообщать их приращения по опросу от компьютера. С передачей состояния кнопок таких проблем не возникает: время нажатия относительно периода опроса достаточно велико.

Принципы работы PS/2 мыши и принципы программного взаимодействия с ней рассмотрены в разделах 1.1 и 1.3. Манипулятор типа Serial Mouse рассматривается в ниже следующих разделах.

2.1. Манипуляторы мышь, подключаемые к COM-портам.

Serial Mouse — мышь с последовательным интерфейсом, подключаемая через 9-или 25-контактный разъем COM-порта (табл. 2.1). Эта мышь имеет встроенный микроконтроллер, который обрабатывает сигналы от координатных датчиков и кнопок. Каждое событие — перемещение мыши или нажатие-отпускание кнопки — кодируется двоичной посылкой по интерфейсу RS-232C. Для передачи информации

применяется асинхронная передача, а двуполярное питание, требуемое по протоколу RS-232, обеспечивается от управляющих линий интерфейса. Недостатком мыши Serial Mouse является то, что она занимает COM-порт и требует монопольного использования его штатной линии прерывания (IRQ4 для COM1 и IRQ3 для COM2). Конечно, то, что для работы мыши порту COM1 требуется именно прерывание IRQ4, является недостатком не самой мыши, а ее программного драйвера, но для пользователя, не увлекающегося написанием «мышиных» драйверов, важен сам факт этого ограничения. Две основные разновидности последовательных мышей — *MS Mouse* (Microsoft Mouse) и *PC Mouse* (Mouse Systems Mouse) — требуют разных драйверов, многие мыши имеют переключатель MS/PC. Эти два типа мышей при одинаковой скорости 1200 бит/с, одном стоп-бите и отсутствии контроля четности используют разные форматы посылок, приведенные в разделе 2.1.3.

- *MS Mouse:* 7 битов данных, трехбайтный пакет (в «классическом» варианте), положительным значениям соответствует перемещение по координате *X* вправо, а по координате *Y* вниз. Для трехкнопочных мышей добавляется четвертый байт, передаваемый только при изменении состояния средней кнопки. Для 3D-мыши четвертый байт имеет иное назначение.
- *PC Mouse:* 8 битов данных, пятибайтный пакет, положительным значениям соответствует перемещение по координате *X* вправо, а по координате У вверх.

Несовпадение форматов объясняет беспорядочные перемещения курсора на экране при несоответствии драйвера типу мыши.

Таблица 2.1. **Разъемы Serial Mouse**

Сигнал мыши	Сигнал COM-порта	Контакт DB9	Контакт DB25
Data	RD	2	3
GND	SG	5	7
+V(питание)	RTS, (DTR)	7 (4)	4 (20)
-V(питание)	TD	3	2

Системная поддержка последовательной мыши осуществляется только на уровне ОС (сервисы вызываются через Int 33h), драйвер мыши — загружаемый или встроенный в ОС. BIOS мышь не поддерживает, даже если и пользуется ею для навигации в CMOS Setup. Еще раз подчеркнем, что для работы мыши обязательно требуется линия аппаратного прерывания — IRQ4 или IRQ3 для последовательных мышей на портах COM1 или COM2 соответственно.

Конструкция Serial Mouse аналогична конструкции обычной PS/2 мыши и имеет свободно вращающийся массивный обрезиненный шарик в днище корпуса, передающий вращение на два координатных диска с фотоэлектрическими датчиками. Датчики для каждой координаты представляют собой две открытые оптопары (светодиод-фотодиод), в оптический канал которых входит вращающийся диск с прорезями. Оптопары датчиков могут оформляться в виде монолитных конструкций или быть просто отдельными элементами, установленными на печатной плате. В мыши имеется *микроконтроллер,* который обрабатывает сигналы с датчиков и посылает в ПК информацию о перемещениях и состоянии кнопок.

2.1.1. Схемотехника манипуляторов Serial Mouse.

Схемотехника Serial Mouse имеет много общего со схемотехникой обычной PS/2 мыши. На рис. 2.1 представлена общая электрическая схема универсальной Serial Mouse, которая может работать в режиме MS Mouse и PC Mouse. Переключение режимов работы осуществляется переключателем SW4, расположенным на нижней части корпуса манипулятора.

Компоненты электрической схемы.

- **SW1** - кнопочный датчик правой клавиши мыши.
- **SW2** - кнопочный датчик средней клавиши мыши.
- **SW3** - кнопочный датчик левой клавиши мыши.
- **X1** - кварцевый резонатор с частотой 32 768 Гц.
- **R2 и R3** - резисторы, через которые на контроллер мыши К0303028 подается напряжение питания положительной полярности.
- **C1** - конденсатор, фильтрующий высокочастотные помехи в цени питания отрицательной полярности.
- **J1** - разъем подключения интерфейсного кабеля к электронному модулю мыши.
- **U1** - микроконтроллер мыши типа К0303028
- **JR1** - резистор - перемычка величиной 1 Ом, включенный в цепи питания светодиодов EL1 и EL1.
- **ZD1** - ограничительный стабилитрон напряжения цепи питания отрицательной полярности.
- **R1** - резистор величиной 990 Ом, задающий величину тока в цепи светодиодов LE1 и LE2.
- **LE1** - светодиод оптопары датчика перемещения по горизонтали.
- **DUAL1** - сдвоенный фотодиод оптопары датчика перемещения по горизонтали.
- **DUAL2** - сдвоенный фотодиод оптопары датчика перемещения по вертикали.
- **LE1** - светодиод оптопары датчика перемещения по вертикали.
- **SW4** - переключатель режимов работы мыши (MS Mouse/PC Mouse).

- **J2** - разъем интерфейсного кабеля, подключаемый к COM-порту компьютера.

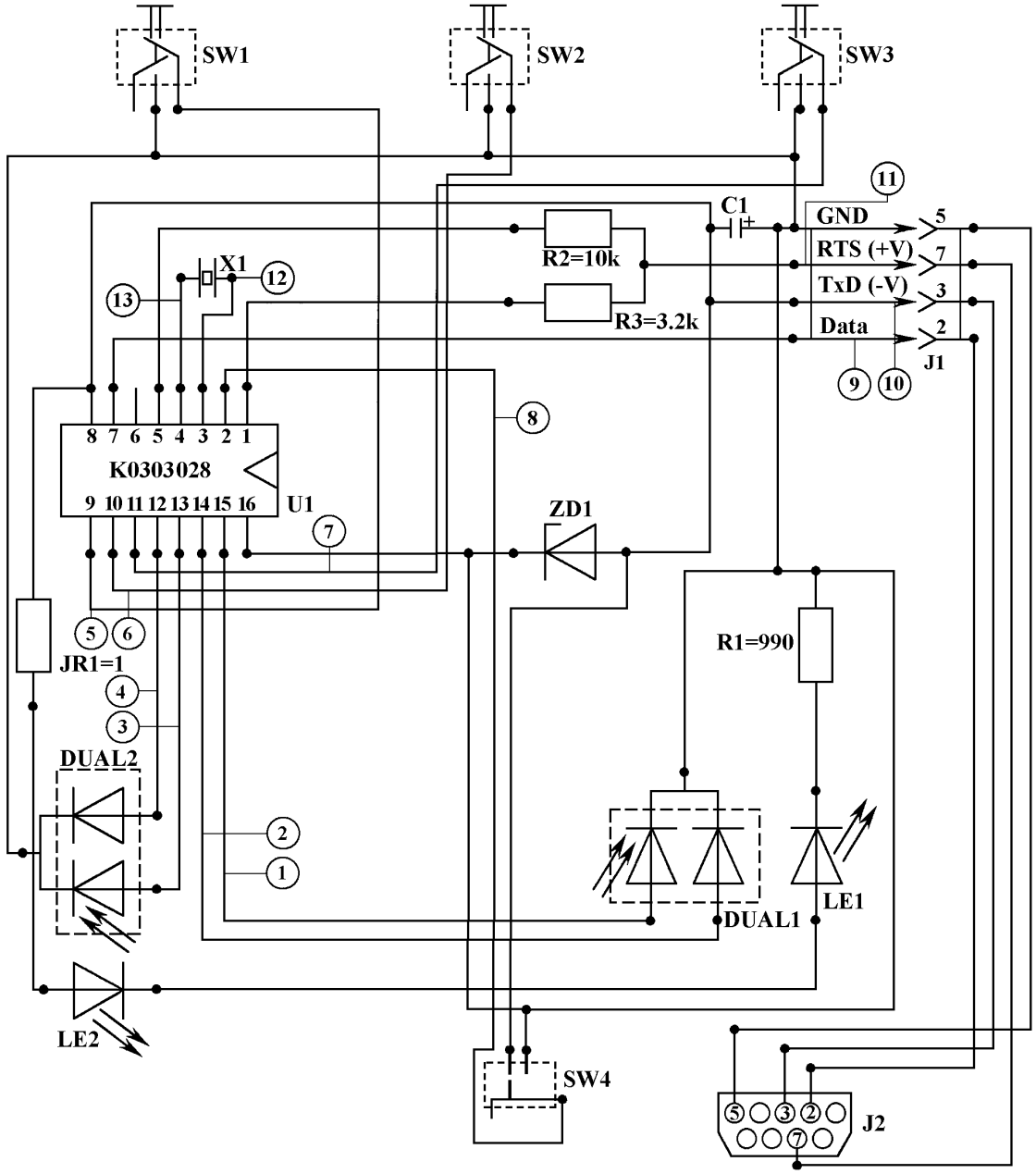

Рис.2.1. Общая электрическая схема универсальной COM мыши.

Контрольные точки электрической схемы.

Контрольные точки на схеме указаны цифрами, обведенными окружностями. Они обозначают цепи электрической схемы COM мыши, в которых можно контролировать характерные электрические сигналы.

1, 2 - сигналы, поступающие с фотодиодов оптопары датчика перемещения по горизонтали; сдвиг по фазе этих сигналов составляет примерно 90°, что позволяет определить направление перемещения мыши по горизонтали.

3, 4 - сигналы, поступающие с фотодиодов оптопары датчика перемещения по вертикали; сдвиг по фазе этих сигналов составляет примерно 90°, что позволяет определить направление перемещения мыши по вертикали.

5 - сигнал с кнопочного датчика **SW1** правой клавиши мыши.

6 - сигнал с кнопочного датчика **SW2** средней клавиши мыши.

7 - сигнал с кнопочного датчика **SW3** левой клавиши мыши.

8 - сигнал с переключателя **SW4** режима работы мыши.

9 - сигналы на линии данных интерфейса мыши.

10 - уровень напряжения питания мыши отрицательной полярности, оно поступает на мышь сразу же после инициализации контроллера последовательного порта процедурами начального теста POST BIOS.

11 - уровень напряжения питания мыши положительной полярности; процедуры начального теста POST BIOS устанавливают на этой линии отрицательный уровень напряжения (-12 В); положительный уровень

напряжения устанавливает драйвер мыши после его загрузки установкой "единиц" в двух младших разрядах регистра управления модемом (MCR) контроллера COM порта.

12, 13 - сигналы на выводах кварцевого резонатора **X1**.

2.1.2. Интерфейс Serial Mouse.

Как уже упоминалось Serial Mouse подключается к персональному компьютеру типа PC/AT через последовательный асинхронный интерфейс - COM-порт. Универсальный последовательный внешний интерфейс — *COM-порт* (Communications Port — коммуникационный порт) присутствует в PC начиная с первых моделей. Этот порт обеспечивает *асинхронный* обмен по стандарту RS-232C. (*Синхронный* обмен в PC поддерживают лишь специальные адаптеры, например SDLC или V.35). COM-порты реализуются на микросхемах универсальных асинхронных приемопередатчиков *(UART),* совместимых с семейством i8250/16450/16550. Они занимают в пространстве ввода-вывода по 8 смежных 8-битных регистров и могут располагаться по стандартным *базовым адресам* 3F8h (COM1), 2F8h (COM2), 3E8h (COM3), 2E8h (COM4). Порты могут вырабатывать *аппаратные прерывания* IRQ4 (обычно используются для COM1 и COM3) и IRQ3 (для COM2 и COM4). С внешней стороны порты имеют линии последовательных данных передачи и приема, а также набор сигналов управления и состояния, соответствующий стандарту RS-232C. COM-порты имеют внешние разъемы-вилки (male — «папа») *DB25P* или *DB9P,* выведенные на заднюю панель компьютера (см. п. 2.1). Характерной особенностью интерфейса является применение «не ТТЛ» сигналов — все внешние сигналы порта двуполярные. Гальваническая развязка отсутствует — схемная земля подключаемого устройства соединяется со схемной землей компьютера. Скорость передачи данных может достигать 115 200 бит/с.

Компьютер может иметь до четырех последовательных портов COM1-COM4 (для машин класса AT типично наличие двух портов) с поддержкой на уровне BIOS. Сервис BIOS Int 14h обеспечивает инициализацию порта, ввод и вывод символа (не используя прерываний) и опрос состояния. Через Int 14h скорость передачи программируется в диапазоне 110-9600 бит/с (меньше, чем реальные возможности порта). Для повышения производительности широко используется взаимодействие программ с портом на уровне регистров, для чего требуется совместимость аппаратных средств COM-порта с программной моделью i8250/16450/16550.

Название порта указывает на его основное назначение — подключение коммуникационного оборудования (например, модема) для связи с другими компьютерами, сетями и периферийными устройствами. К порту могут непосредственно подключаться и периферийные устройств с последовательным интерфейсом: принтеры, плоттеры, терминалы и другие. COM-порт широко используется для подключения мыши, а также организации непосредственной связи двух компьютеров. К COM-порту подключают и электронные ключи.

Практически все современные системные платы (еще начиная с PCI-плат для процессоров 486) имеют встроенные адаптеры двух COM-портов. Один из портов может использоваться и для беспроводной инфракрасной связи с периферийными устройствами (IrDA). Существуют карты ISA с парой COM-портов, где они чаще всего соседствуют с LPT-портом, а также с контроллерами дисковых интерфейсов (FDC+IDE). Если возникает потребность в большом количестве последовательных интерфейсов, то в ПК можно установить специальные адаптеры-мультиплексоры. Это весьма дорогие карты, они выпускаются обычно на 4, 8, 12 и даже 16 портов. Такое большое число разъемов на заднюю стенку ПК вывести проблематично, и у мультиплексоров обычно имеется внешний блок с разъемами (и электроникой), соединяемый с адаптером кабелем с многоконтактными разъемами. BIOS мультиплексоры не поддерживает.

«Классический» COM-порт позволяет осуществлять обмен данными только программно-управляемым способом, при этом для пересылки каждого байта процессору приходится выполнять несколько инструкций. Современные порты, выполненные на базе контроллеров типа 16550A, имеют FIFO-буферы данных и позволяют выполнять обмен по каналу DMA, однако в BIOS Setup большинства современных PC совместимых компьютеров не предусмотрено выделение COM-портам каналов DMA, как, например, это предусмотрено для режима ECP LPT-порта.

2.1.2.1. Интерфейс RS-232C

Интерфейс RS-232C предназначен для подключения аппаратуры, передающей или принимающей данные *(ООД* — оконечное оборудование данных, или *АПД* — аппаратура передачи данных; *DTE— Data Terminal Equipment),* к оконечной аппаратуре каналов данных *(АКД; DCE — Data Communication Equipment).* В роли АПД может выступать компьютер, принтер, плоттер и другое периферийное оборудование. В роли АКД обычно выступает модем. Конечной целью подключения является соединение двух устройств АПД. Полная схема соединения приведена на рис. 2.2; интерфейс позволяет исключить канал удаленной связи вместе с парой устройств АКД, соединив устройства непосредственно с помощью нуль-модемного кабеля (рис. 2.3).

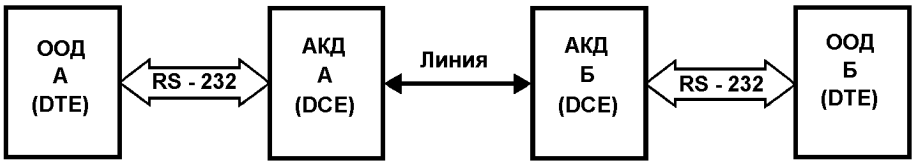

Рис. 2.2**. Полная схема соединения по RS-232C**

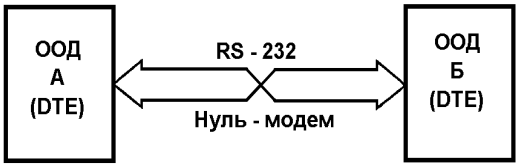

Рис. 2.3. Соединение по RS-232C нуль-модемным кабелем

Стандарт описывает управляющие сигналы интерфейса, пересылку данных, электрический интерфейс и типы разъемов. В стандарте предусмотрены асинхронный и синхронный режимы обмена, но COM-порты поддерживают только *асинхронный режим.* Функционально *RS-232C* эквивалентен стандарту МККТТ V.24/ V.28 и стыку С2, но они имеют различные названия сигналов.

Стандарт *RS-232C* описывает несимметричные передатчики и приемники — сигнал передается относительно общего провода — схемной земли (симметричные дифференциальные сигналы используются в других интерфейсах — например, *RS-422).* Интерфейс *не обеспечивает гальванической развязки* устройств. Логической единице соответствует напряжение на *входе приемника* в диапазоне от -12 до -3 В. Для линий управляющих сигналов это состояние называется *ON* («включено»), для линий последовательных данных — *MARK.* Логическому нулю соответствует диапазон от +3 до +12 В. Для линий управляющих сигналов состояние называется OFF(«выключено»), а для линий последовательных данных — *SPACE.* Диапазон от -3 до +3 В — зона нечувствительности, обусловливающая гистерезис приемника: состояние линии будет считаться измененным только после пересечения порога (рис. 2.4). Уровни сигналов на выходах передатчиков должны быть в диапазонах от -12 до -5 В и от +5 до +12 В для представления единицы и нуля соответственно. Разность потенциалов между схемными землями (SG) соединяемых устройств должна быть менее 2 В, при более высокой разности потенциалов возможно неверное восприятие сигналов.

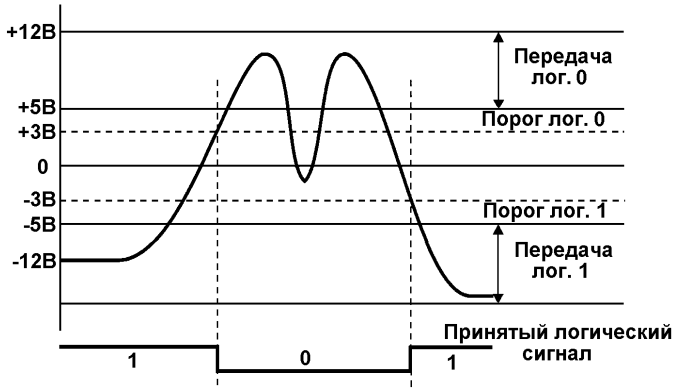

Рис. 2.4. Прием сигналов RS-232C

Интерфейс предполагает наличие *защитного заземления* для соединяемых устройств, если они оба питаются от сети переменного тока и имеют сетевые фильтры.

ВНИМАНИЕ -

Подключение и отключение интерфейсных кабелей устройств с автономным питанием должно производиться *при отключенном питании.* Иначе разность не выровненных потенциалов устройств в момент коммутации может оказаться приложенной к выходным или входным (что опаснее) цепям интерфейса и вывести из строя микросхемы.

Стандарт RS-232C регламентирует *типы применяемых разъемов.*

На аппаратуре АПД (в том числе на COM-портах) принято устанавливать *вилки DB-25P* - рис.2 5а или более компактный вариант — *DB-9P* (рис. 2.5б). Девятиштырьковые разъемы не имеют контактов для дополнительных сигналов, необходимых для синхронного режима (в большинстве 25-штырьковых разъемах эти контакты не используются). На аппаратуре *АКД* (модемах) устанавливают *розетки DB-25S* или *DB-9S.*

Это правило предполагает, что разъемы АКД могут подключаться к разъемам АПД непосредственно или через переходные «прямые» кабели с розеткой и вилкой, у которых контакты соединены «один в один». Переходные кабели могут являться и переходниками с 9 на 25-штырьковые разъемы (рис. 2.6).

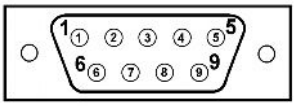

Рис. 2.5а. **9-контактный штырьковый разъем, вид со стороны задней стенки компьютера**

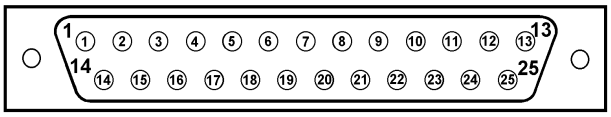

Рис. 2.5б. **25-контактный штырьковый разъем, вид со стороны задней стенки компьютера**

Если аппаратура АПД соединяется без модемов, то разъемы устройств (вилки) соединяются между собой *нуль-модемным кабелем* (Zero-modem, или Z-modem), имеющим на обоих концах розетки, контакты которых соединяются перекрестно по одной из схем, приведенных на рис. 2.7.

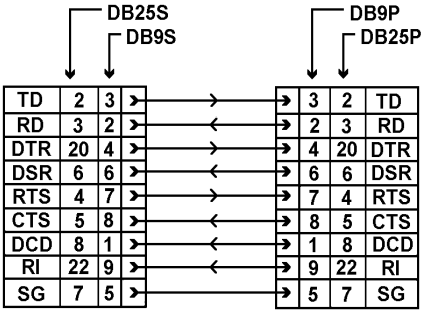

Рис. 2.6. Кабели подключения модемов

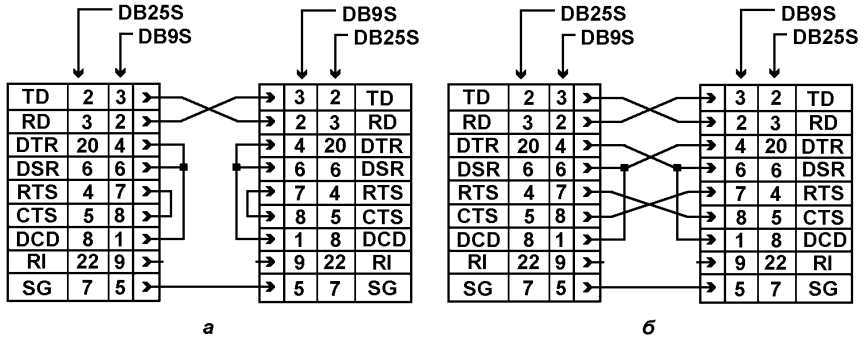

Рис. 2.7. Нуль-модемный кабель: а — минимальный, б — полный

Если на каком-либо устройстве АПД установлена розетка — это почти 100 % того, что к другому устройству оно должно подключаться прямым кабелем, аналогичным кабелю подключения модема. Розетка устанавливается обычно на тех устройствах, у которых удаленное подключение через модем не предусмотрено.

В табл. 2.1 приведено назначение контактов разъемов COM-портов (и любой другой аппаратуры передачи данных АПД). Контакты разъема DB-25S определены стандартом EIA/TIA-232-E, разъем DB-9S описан стандартом EIA/TIA-574. У модемов (АКД) название цепей и контактов такое же, но роли сигналов (вход-выход) меняются на противоположные.

Таблица 2.1**. Разъемы и сигналы интерфейса RS-232C**

Обозначение цепи			Контакт разъема		№ провода кабеля выносного разъема РС				Направление
COM-порт	RS-232	V.24 Стык 2	DB-25P	DB-9P	1[1]	2[2]	3[3]	4[4]	I/O
PG	AA	101	1	5	(10)	(10)	(10)	1	-
SG	AB	102	7	5	5	9	1	13	-
TD	BA	103	2	3	3	5	3	3	O
RD	BB	104	3	2	2	3	4	5	I
RTS	CA	105	4	7	7	4	8	7	O
CTS	CB	106	5	8	8	6	7	9	I
DSR	CC	107	6	6	6	2	9	11	I
DTR	CD	108/2	20	4	4	7	2	14	O
DCD	CF	109	8	1	1	1	5	15	I
RI	CE	125	22	9	9	8	6	18	I

[1] Ленточный кабель 8-битных мультиадаптеров.

[2] Ленточный кабель 16-битных мультиадаптеров и портов на системных платах.
[3] Вариант ленточного кабеля портов на системных платах.
[4] Широкий ленточный кабель к 25-контактному разъему.

Подмножество сигналов *RS-232C,* относящихся к асинхронному режиму, рассмотрим с точки зрения COM-порта PC. Для удобства будем пользоваться мнемоникой названий, принятой в описаниях COM-портов и большинства устройств (она отличается от безликих обозначений RS-232 и V.24). Напомним, что активному состоянию сигнала («включено») и *логической единице* передаваемых данных соответствует *отрицательный потенциал* (ниже -3 В) сигнала интерфейса, а состоянию «выключено» и *логическому нулю — положительный* (выше +3 В). Назначение сигналов интерфейса приведено в табл. 2.2. Нормальную последовательность управляющих сигналов для случая подключения модема к COM-порту иллюстрирует рис. 2.8.

Таблица 2.2. Назначение сигналов интерфейса RS-232C

Сигнал	Назначение
PG	*Protected Ground* — защитная земля, соединяется с корпусом устройства и экраном кабеля
SG	*Signal Ground* — сигнальная (схемная) земля, относительно которой действуют уровни сигналов
TD	*Transmit Data* — последовательные данные — выход передатчика
RD	*Receive Data* — последовательные данные — вход приемника
RTS	*Request To Send* — выход запроса передачи данных: состояние «включено» уведомляет модем о наличии у терминала данных для передачи. В полудуплексном режиме используется для управления направлением — состояние «включено» служит сигналом модему на переключение в режим передачи
CTS	*Clear To Send* — вход разрешения терминалу передавать данные. Состояние «выключено» запрещает передачу данных. Сигнал используется для аппаратного управления потоками данных
DSR	*Data Set Ready*— вход сигнала готовности от аппаратуры передачи данных (модем в рабочем режиме подключен к каналу и закончил действия по согласованию с аппаратурой на противоположном конце канала)
DTR	*Data Terminal Ready* — выход сигнала готовности терминала к обмену данными. Состояние «включено» поддерживает коммутируемый канал в состоянии соединения
DCD	*Data Carrier Detected* — вход сигнала обнаружения несущей удаленного модема
RI	*Ring Indicator* — вход индикатора вызова (звонка). В коммутируемом канале этим сигналом модем сигнализирует о принятии вызова

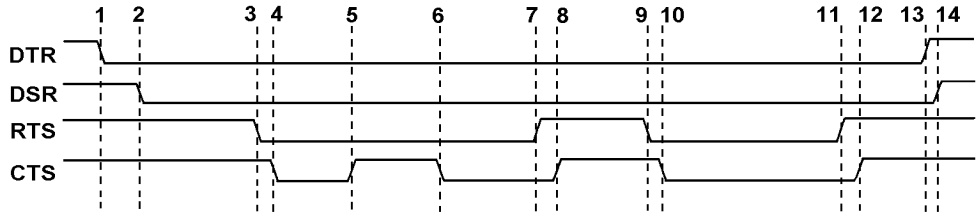

Рис. 2.8. Последовательность управляющих сигналов интерфейса

1. Установкой DTR компьютер указывает на желание использовать модем.
2. Установкой DSR модем сигнализирует о своей готовности и установлении соединения.
3. Сигналом RTS компьютер запрашивает разрешение на передачу и заявляет о своей готовности принимать данные от модема.
4. Сигналом CTS модем уведомляет о своей готовности к приему данных от компьютера и передаче их в линию.
5. Снятием CTS модем сигнализирует о невозможности дальнейшего приема (например, буфер заполнен) — компьютер должен приостановить передачу данных.
6. Сигналом CTS модем разрешает компьютеру продолжить передачу (в буфере появилось место).
7. Снятие RTS может означать как заполнение буфера компьютера (модем должен приостановить передачу данных в компьютер), так и отсутствие данных для передачи в модем. Обычно в этом случае модем прекращает пересылку данных в компьютер.
8. Модем подтверждает снятие RTS сбросом CTS.
9. Компьютер повторно устанавливает RTS для возобновления передачи.
10. Модем подтверждает готовность к этим действиям.
11. Компьютер указывает на завершение обмена.
12. Модем отвечает подтверждением.
13. Компьютер снимает DTR, что обычно является сигналом на разрыв соединения («повесить трубку»).
14. Модем сбросом DSR сигнализирует о разрыве соединения.

Из рассмотрения этой последовательности становятся понятными соединения DTR-DSR и RTS-CTS в

нуль-модемных кабелях.

2.1.2.2. Родственные интерфейсы и преобразователи уровней

В последовательном интерфейсе далеко не всегда используют двуполярные сигналы RS-232C — это неудобно, хотя бы из-за необходимости использования двуполярного питания приемопередатчиков. Сами микросхемы вышеописанных приемопередатчиков UART работают с сигналами логики ТТЛ или КМОП; такие же сигналы используются, например, и в сервисных портах винчестеров и других устройств. Многие устройства (в том числе карманные ПК и мобильные телефоны) имеют внешний последовательный интерфейс с уровнями низковольтной логики. Конечно, сигналы обычной логики не имеют столь высокой помехоустойчивости, как RS-232C, но не всегда это и требуется.

Для взаимного преобразования уровней интерфейса RS-232C и логики используются буферные микросхемы приемников (с гистерезисом) и передатчиков двуполярного сигнала. Преобразователи уровней располагаются между UART и разъемом RS232. Преобразователи передатчиков конвертируют уровень напряжения ТТЛ в уровень RS232, а преобразователи приемников - наоборот. Помимо этого они инвертируют выходные сигналы управления (RTS и DTR) и последовательные сигналы данных SIN (RD) и SOUT (TD). Входные сигналы состояния (CTS, DSR, DCD и RI) не инвертируются. Следует отметить, что сигналы управления и состояния инвертируются в самом контроллере, а последовательные сигналы выходных и входных данных - не инвертируются. Это означает, что:

- при записи 1 в разряды MCR.0 и MCR.1 регистра управления модемом (MCR) выходные линии RTS и DTR устанавливаются в состояние +12В;
- при установке на входных линиях CTS, DSR, RI и DCD напряжения -12В, в разрядах (0 - 3) регистра состояния модема (MSR) устанавливаются единицы;
- единица в сдвиговом регистре передатчика передается в линию TD уровнем -12В;
- уровень -12В на линии RD приводит к записи в сдвиговый регистр приемника единицы.

Особенности подключения преобразователей уровней между контроллером и разъемом RS 232C показаны на рис. 2.9а.

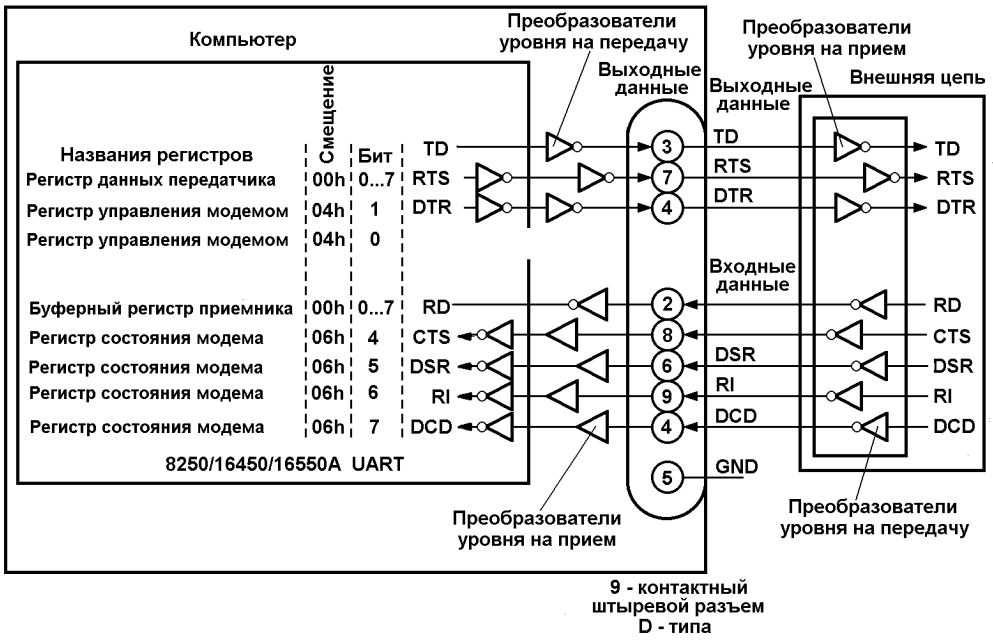

Рис. 2.9а. Логическая структура последовательного порта

При несоблюдении правил заземления и коммутации преобразователи уровней обычно становятся первыми жертвами «пиротехнических» эффектов. Раньше их нередко устанавливали в «кроватки», что облегчало их замену.

Часто буферные схемы входят прямо в состав интерфейсных БИС. Это удешевляет изделие, экономит место на плате, но в случае аварии оборачивается крупными финансовыми потерями. Вывести из строя интерфейсные микросхемы замыканием сигнальных цепей маловероятно: ток короткого замыкания передатчиков обычно не превышает 20 мА.

В специальных кабелях-адаптерах часто применяют преобразователи уровней фирмы Maxim и Sypex; они удобны тем, что содержат и приемники, и передатчики. Из широкого ассортимента этих преобразователей легко подобрать подходящий по количеству приемников и передатчиков, а также по питанию (однополярному, двуполярному, низковольтному).

Когда требуется большая помехоустойчивость (дальность и скорость передачи), применяют иные электрические варианты последовательных интерфейсов: *RS-422A (V.11, X.27), RS-423A* (*V. 10, X.26), RS-485.* На рис. 2.10 приведены схемы соединения приемников и передатчиков, а также показаны ограничения на

длину линии (L) и максимальную скорость передачи данных (V). Несимметричные линии интерфейсов *RS-232C* и *RS-423A* имеют самую низкую защищенность от синфазной помехи, хотя дифференциальный вход приемника *RS-423A* позволяет в какой-то мере исправить ситуацию. Лучшие параметры имеют интерфейсы *RS-422A* и *RS-485,* работающие на симметричных линиях связи. В них для передачи каждого сигнала используются дифференциальные приемопередатчики с отдельной (витой) парой проводов для каждой сигнальной цепи.

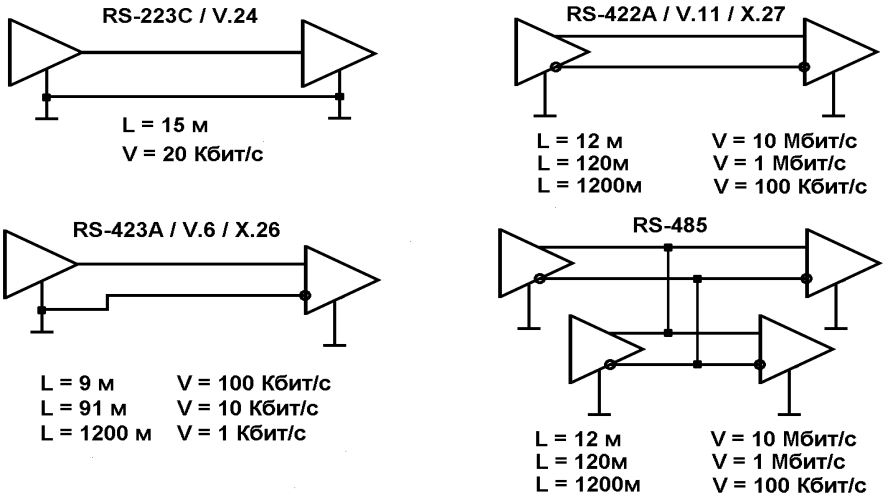

Рис. 2.10. Стандарты последовательных интерфейсов

Интерфейсы EIA-RS-422 (ITU-T V.11, X.27) и *EIA-RS-485* (ISO 8482) используют симметричную передачу сигнала и допускают как двухточечную, так и шинную топологию соединений. В них информативной является разность потенциалов между проводниками A и B. Если на входе приемника U_A-U_B >0,2 В (A положительнее B) — состояние «выключено» (space), U_A-U_B<-0,2 В (A отрицательнее B) — состояние «включено» (mark). Диапазон $|U_A-U_B|\le 0{,}2$ В является зоной нечувствительности (гистерезис), защищающей от воздействия помех. На выходах передатчика сигналы U_A и U_B обычно переключаются между уровнями 0 и +5 В (КМОП) или +1 и +4 В (ТТЛ), дифференциальное выходное напряжение должно лежать в диапазоне 1,5 - 5 В. Выходное сопротивление передатчиков 100 Ом. Интерфейсы электрически совместимы между собой, хотя и имеют некоторые различия в ограничениях. Принципиальное отличие передатчиков RS-485 — возможность переключения в третье состояние. Передатчики RS-422/485 совместимы с приемниками RS-423. Основные параметры интерфейсов приведены в табл. 2.3, топологию соединений иллюстрирует рис. 2.11.

Чтобы увеличить число узлов, можно повысить входное сопротивление приемников, но при этом снижается допустимая скорость или максимально возможная дальность передачи. Максимальная скорость передачи на коротких расстояниях (до 10м) ограничивается быстродействием передатчиков (достижима частота 25 МГц). На средних расстояниях ограничение определяется емкостью кабеля (1200 бит/с - 30 нФ, 9600 бит/с - 25 нФ, 115 кбит/с -250 пФ). Максимальная дальность (1200 м) ограничена сопротивлением петли постоянному току.

Таблица 2.3. Параметры интерфейсов RS-422 и RS-485

Параметр	RS-422	RS-485
Порог срабатывания, $\|U_A - U_B\|$, В	0,2	0,2
Допустимое напряжение синфазной помехи, В[1]	-6,8...+6,8	-6,8...+11,8
Допустимое напряжение на входах, В[1]	-7...+7	-7...+12
Входное сопротивление приемника, кОм	4	12
Минимальное сопротивление нагрузки передатчика, Ом	100	60
Максимальное число узлов	1 передатчик +10 приемников	32 (передатчиков, приемников или их комбинаций)
Максимальная длина, м	1200 (100 кбит/с) 12 (10Мбит/с)	1200 (100 кбит/с) 12(10Мбит/с)
Терминаторы, R=100 Ом	На дальнем конце от передатчика	На обоих концах
Ток короткого замыкания, мА	<150 на шину GND	<250 на шину с потенциалом -7...+12 В или между проводами A и B

[1] Напряжение измеряется относительно «схемной земли» узла.

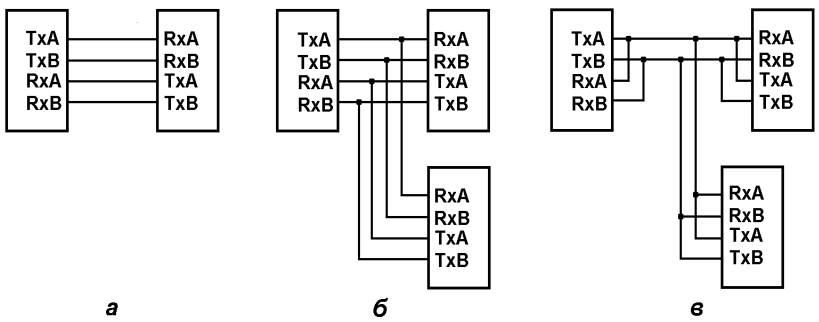

Рис. 2.11. Топология интерфейсов: а — RS-422, б — RS-485 четырехпроводный, в — RS-485 двухпроводный.

Интерфейс RS-485 может быть в двух версиях: двухпроводной и четырехпроводной. Четырехпроводная версия (рис. 2.11, *б)* выделяет задающий узел (master), передатчик которого работает на приемники всех остальных. Передатчик задающего узла всегда активен — переход в третье состояние ему не нужен. Передатчики остальных ведомых (slave) узлов должны иметь тристабильные выходы, они объединяются на общей шине с приемником ведущего узла. В двухпроводной версии (рис. 2.11, *в)* все узлы равноправны.

В вырожденном случае — при двухточечном соединении — интерфейсы RS-485 и RS-422 эквивалентны, и третье состояние не используется.

Для определенности состояния покоя шины RS-485, когда нет активных передатчиков, на линию устанавливают активные терминаторы, «растягивающие» потенциалы проводов. В покое провод B должен иметь более положительный потенциал, чем A.

При многоточечном соединении необходимо организовать метод доступа к среде передачи. Чаще всего используют полинг (polling) — опрос готовности к передаче, выполняемый ведущим устройством, или передачу права доступа в соответствии с определенным (установленным) регламентом. Иногда используют и методы случайного доступа (аналогично Ethernet).

Дифференциальный вход интерфейсов защищает от действия помех, но при этом должно осуществляться соединение «схемных земель» устройств между собой и с шиной заземления. Для соединения устройств между собой используют третий провод интерфейса (можно и экран). Для того чтобы по третьему проводу не протекал большой ток, выравнивающий «земляные потенциалы», в его цепь включают резисторы (рис. 2.12).

Интерфейс RS-422 часто используется для подключения периферийных устройств (например, принтеров). Интерфейс RS-485 популярен в качестве шин устройств промышленной автоматики.

Интерфейс «*токовая петля*» для представления сигнала использует не напряжение, а *ток* в двухпроводной линии, соединяющей приемник и передатчик.

Логической единице (состоянию «включено») соответствует протекание тока 20 мА, а логическому нулю — отсутствие тока. Такое представление сигналов для вышеописанного формата асинхронной посылки позволяет обнаружить обрыв линии — приемник заметит отсутствие стоп-бита (обрыв линии действует как постоянный логический нуль).

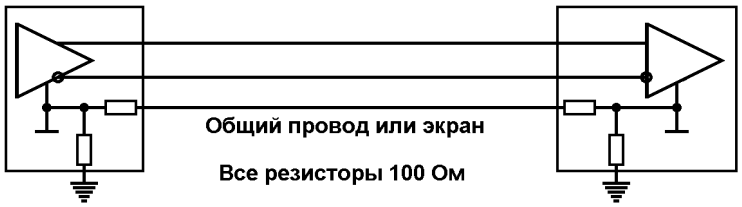

Рис. 2.12. Соединение «схемных земель» для интерфейсов RS-422 и RS-485

Токовая петля обычно предполагает *гальваническую развязку* входных цепей приемника от схемы устройства. При этом источником тока в петле является передатчик (этот вариант называют активным передатчиком). Возможно и питание от приемника (активный приемник), при этом выходной ключ передатчика может быть также гальванически развязан с остальной схемой передатчика. Существуют упрощенные варианты без гальванической развязки, но это уже вырожденный случай интерфейса. Заметим, что интерфейс MIDI (см. п. 8.5.3) с «классической» токовой петлей несовместим.

Токовая петля с гальванической развязкой позволяет передавать сигналы на расстояния до нескольких километров, но при невысоких скоростях (выше 19 200 бит/с не используют, а на километровых расстояниях допустима скорость до 9600 бит/с и ниже). Допустимое расстояние определяется сопротивлением пары проводов и уровнем помех. Поскольку интерфейс требует пары проводов для каждого сигнала, обычно используют только два сигнала последовательного интерфейса (4-проводная линия). В случае двунаправленного обмена применяются только сигналы передаваемых и принимаемых данных, а для управления потоком используется программный метод XON/XOFF. Если двунаправленный обмен не требуется, применяют одну линию данных, а для управления потоком обратная линия задействуется для

сигнала CTS (аппаратный протокол) или встречной линии данных (программный протокол). При надлежащем ПО одной токовой петлей можно обеспечить двунаправленную полудуплексную связь двух устройств. При этом каждый приемник «слышит» как сигналы передатчика на противоположной стороне канала, так и сигналы своего передатчика. Они расцениваются коммуникационными пакетами просто как эхо-сигнал. Для безошибочного приема передатчики должны работать поочередно.

Токовая петля позволяет использовать выделенные физические линии без модемов, но на малых скоростях. Иногда по токовой петле подключают терминалы с интерфейсом RS-232C, если не хватает штатной длины интерфейса или требуется гальваническая развязка. Преобразовать сигналы *RS-232C* в токовую петлю несложно — на рис. 2.13 приведена простейшая схема преобразователя применительно к подключению терминала. Для получения двуполярного сигнала, требуемого для входных сигналов COM-порта, применяется питание от интерфейса. Схема может быть усложнена для защиты оптронов от перегрузки и улучшения формы потенциальных сигналов. Допустимая скорость определяется и быстродействием применяемых оптронов (скорость 9600 бит/с достигается практически на любых оптронах).

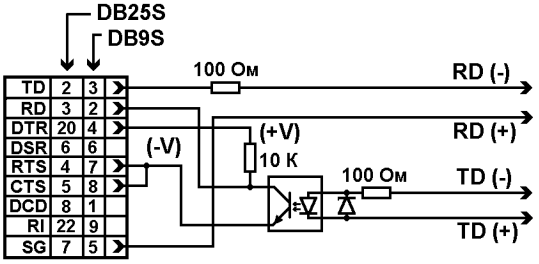

Рис. 2.13. Преобразование интерфейса RS-232C в «токовую петлю»

2.1.2.3. Асинхронный режим передачи

Асинхронный режим передачи является байт-ориентированным (символьно-ориентированным): минимальная пересылаемая единица информации — один байт (один символ). Формат посылки байта иллюстрирует рис. 2.14. Передача каждого байта начинается со *старт-бита,* сигнализирующего приемнику о начале посылки, за которым следуют *биты данных* и, возможно, *бит паритета* (четности). Значение бита определяет состояние линии передачи в течение битового интервала, длительность которого определяется скоростью передачи. Завершает посылку *стоп-бит,* гарантирующий паузу между посылками. Старт-бит следующего байта посылается в любой момент после стоп-бита, то есть между передачами возможны паузы произвольной длительности. Старт-бит, имеющий всегда строго определенное значение (логическая 1), обеспечивает простой механизм синхронизации приемника по сигналу от передатчика. Подразумевается, что приемник и передатчик работают на одной скорости обмена. Внутренний генератор синхронизации приемника использует счетчик-делитель опорной частоты, обнуляемый в момент приема начала старт-бита. Этот счетчик генерирует внутренние стробы, по которым приемник фиксирует последующие принимаемые биты. В идеале стробы располагаются в середине битовых интервалов, что позволяет принимать данные и при незначительном рассогласовании скоростей приемника и передатчика. Очевидно, что при передаче 8 бит данных, одного контрольного и одного стоп-бита предельно допустимое рассогласование скоростей, при котором данные будут распознаны верно, не может превышать 5 %. С учетом фазовых искажений и дискретности работы внутреннего счетчика синхронизации реально допустимо меньшее отклонение частот. Чем меньше коэффициент деления опорной частоты внутреннего генератора (чем выше частота передачи), тем больше погрешность привязки стробов к середине битового интервала, и требования к согласованности частот становятся более строгими. Чем выше частота передачи, тем больше влияние искажений фронтов на фазу принимаемого сигнала.

Взаимодействие этих факторов приводит к повышению требований к согласованности частот приемника и передатчика с ростом скорости обмена.

Рис. 2.14. Формат асинхронной передачи RS-232C

Формат асинхронной посылки позволяет выявлять возможные *ошибки передачи.*

- Если принят перепад, сигнализирующий о начале посылки, а по стробу старт-бита зафиксирован уровень логической единицы, старт-бит считается ложным и приемник снова переходит в состояние ожидания. Об этой ошибке приемник может не сообщать.

- Если во время, отведенное под стоп-бит, обнаружен уровень логического нуля, фиксируется ошибка стоп-бита.
- Если применяется контроль четности, то после посылки бит данных передается *контрольный бит.* Этот бит дополняет количество единичных бит данных до четного или нечетного в зависимости от принятого соглашения. Прием байта с неверным значением контрольного бита приводит к фиксации ошибки.
- Контроль формата позволяет обнаруживать обрыв линии: как правило, при обрыве приемник «видит» логический нуль, который сначала трактуется как старт-бит и нулевые биты данных, но потом срабатывает контроль стоп-бита.

Для асинхронного режима принят ряд *стандартных скоростей обмена*: 50, 75, 110, 150,300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600 и 115200 бит/с. Иногда вместо единицы измерения «бит/с» используют «бод» (baud). В бодах принято измерять частоту изменения состояния линии, что для COM-порта одно и тоже. Однако при недвоичном способе кодирования (широко применяемом в современных модемах) в канале связи скорости передачи бит (бит/с) и изменения сигнала (бод) могут отличаться в несколько раз.

Количество *бит данных* может составлять 5, 6, 7 или 8 (5- и 6-битные форматы распространены незначительно). Количество *стоп-бит* может быть 1, 1,5 или 2 («полтора бита» означает только длительность стопового интервала, измеряемая в битовых интервалах).

2.1.2.4. Управление потоком данных

Для управления потоком данных (Flow Control) могут использоваться два варианта протокола — аппаратный и программный. Иногда управление потоком путают с квитированием. *Квитирование* (handshaking) подразумевает посылку уведомления о получении элемента, в то время как *управление потоком* предполагает посылку уведомления о возможности или невозможности последующего приема данных. Зачастую управление потоком основано на механизме квитирования.

Аппаратный протокол управления потоком RTS/CTS (hardware flow control) -использует сигнал CTS, который позволяет остановить передачу данных, если приемник не готов к их приему (рис. 2.15). Передатчик «выпускает» очередной байт только при включенной линии CTS. Байт, который уже начал передаваться, задержать сигналом CTS невозможно (это гарантирует целостность посылки). Аппаратный протокол обеспечивает самую быструю реакцию передатчика на состояние приемника. Микросхемы асинхронных приемопередатчиков имеют не менее двух регистров в приемной части — сдвигающий, для приема очередной посылки, и хранящий, из которого считывается принятый байт. Это позволяет реализовать обмен по аппаратному протоколу без потери данных.

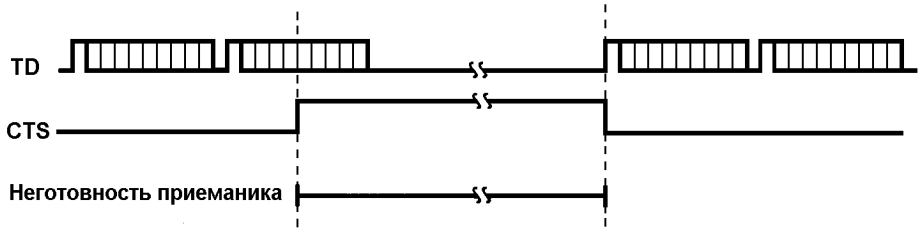

Рис. 2.15. Аппаратное управление потоком

Аппаратный протокол удобно использовать при подключении принтеров и плоттеров, если они его поддерживают. При непосредственном (без модемов) соединении двух компьютеров аппаратный протокол требует перекрестного соединения линий RTS — CTS. При непосредственном соединении по минимальному количеству линий у передающего терминала должно быть обеспечено состояние «включено» на линии CTS (соединением собственных линий RTS — CTS), в противном случае передатчик будет «молчать».

Применяемые в IBM PC приемопередатчики 8250/16450/16550 сигнал CTS аппаратно не обрабатывают, а только показывают его состояние в регистре MSR (см. п. 2.1.2.5.3). Реализация протокола RTS/CTS возлагается на драйвер BIOS Int 14h, и называть его «аппаратным» не совсем корректно. Если же программа, пользующаяся COM-портом, взаимодействует с UART на уровне регистров (а не через BIOS), то обработкой сигнала CTS для поддержки данного протокола она занимается сама. Ряд коммуникационных программ позволяет игнорировать сигнал CTS (если не используется модем), и для них не требуется соединение входа CTS с выходом даже своего сигнала RTS. Однако существуют и иные приемопередатчики (например, 8251), в которых сигнал CTS отрабатывается аппаратно. Для них, а также для «честных» программ, использование сигнала CTS на разъемах (а то и на кабелях) обязательно. Преимущество протокола RTS/CTS во времени реакции (по сравнению с программным методом XON/XOFF) остается лишь для буферированной (в режиме FIFO) передачи.

Программный протокол управления потоком XON/XOFF предполагает наличие двунаправленного канала передачи данных. Работает протокол следующим образом: если устройство, принимающее данные, обнаруживает причины, по которым оно не может их дальше принимать, оно по обратному последовательному каналу посылает байт-символ XOFF (13h). Противоположное устройство, приняв этот символ, приостанавливает передачу. Когда принимающее устройство снова становится готовым к приему данных, оно посылает символ XON (11h), приняв который противоположное устройство возобновляет

передачу. Время реакции передатчика на изменение состояния приемника по сравнению с аппаратным протоколом увеличивается, по крайней мере, на время передачи символа (XON или XOFF) плюс время реакции программы передатчика на прием символа (рис. 2.16). Из этого следует, что данные без потерь могут приниматься только приемником, имеющим дополнительный буфер принимаемых данных и сигнализирующим о неготовности заблаговременно (имея в буфере свободное место).

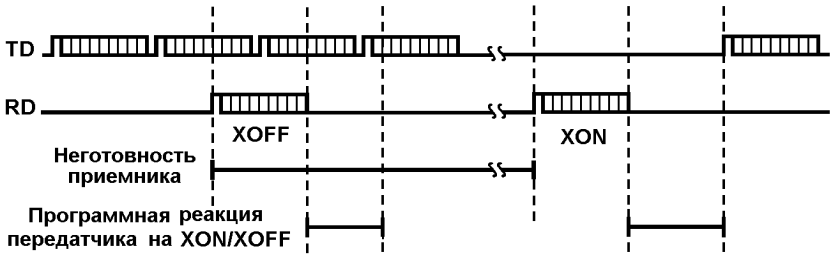

Рис. 2.16**. Программное управление потоком XON/XOFF**

Преимущество программного протокола заключается в отсутствии необходимости передачи управляющих сигналов интерфейса — минимальный кабель для двустороннего обмена может иметь только 3 провода (см. рис. 2.7*а).* Недостатком, помимо обязательного наличия буфера и большего времени реакции (снижающего общую производительность канала из-за ожидания сигнала XON), является сложность реализации полнодуплексного режима обмена. В этом случае из потока принимаемых данных должны выделяться (и обрабатываться) символы управления потоком, что ограничивает набор передаваемых символов.

Кроме этих двух распространенных стандартных протоколов, поддерживаемых и ПУ, и ОС, существуют и другие.

2.1.2.5. Микросхемы асинхронных приемопередатчиков

В COM-портах преобразование параллельного кода в последовательный для передачи и обратное преобразование при приеме данных выполняют специализированные микросхемы UART (Universal Asynchronous Receiver-Transmitter — универсальный асинхронный приемопередатчик - УАПП). Эти же микросхемы формируют и обрабатывают управляющие сигналы интерфейса. COM-порты IBM PC/XT /AT базируются на микросхемах, совместимых на уровне регистров с UART i8250 — 8250/16450/16550A. Это семейство представляет собой усовершенствование начальной модели, направленное на повышение быстродействия, снижение потребляемой мощности и загрузки процессора при интенсивном обмене. Отметим, что:

- 8250 имеет ошибки (появление ложных прерываний), учтенные в XT BIOS;
- 8250A — ошибки исправлены, но в результате потеряна совместимость с BIOS; эта микросхема работает в некоторых моделях AT, но непригодна для скорости 9600 бит/с;
- 8250B — исправлены ошибки 8250 и 8250A, восстановлена ошибка в прерываниях — возвращена совместимость с XT BIOS; работает в AT под DOS (кроме скорости 9600 бит/с).

Микросхемы 8250x имеют невысокое быстродействие по обращениям со стороны системной шины. Они не допускают обращения к своим регистрам в смежных шинных циклах процессора; для корректной работы с ними требуется введение программных задержек между обращениями CPU.

В компьютерах класса AT применяют микросхемы UART перечисленных ниже модификаций.

- 16450 — быстродействующая версия 8250 для AT. Ошибок 8250 и полной совместимости с XT BIOS не имеет.
- 16550 — развитие 16450. Может использовать канал DMA для обмена данными. Имеет FIFO-буфер, но некорректность его работы не позволяет им воспользоваться.
- 16550A — имеет работающие 16-байтные FIFO-буферы приема и передачи и возможность использования DMA. Именно этот тип UART должен применяться в ПК при интенсивных обменах на скоростях 9600 бит/с и выше. Совместимость с этой микросхемой обеспечивает большинство микросхем контроллеров портов ввода-вывода, входящих в современные чипсеты.

UART (УАПП) — это сложное переплетение логических и синхронизирующих схем, которые автоматизируют процесс выполнения последовательной связи. Точная последовательность операций, выполняемых УАПП в каждой конкретной ситуации, контролируется внешними параметрами. Однако в общих чертах работу УАПП при приеме и передаче можно описать следующим образом.

При передаче символа УАПП должен выполнить следующие операции.

- Принять символ (как данные в параллельной форме) от системного интерфейса ввода-вывода.
- Преобразовать символ в последовательность отдельных битов (параллельно-последовательное преобразование).
- Сформировать кадр путем добавления к информационным разрядам стартового и стопового битов, а также бита четности.

- Передать все биты сформированного кадра на интерфейс с требуемой скоростью (в бодах).
- Сообщить о готовности к передаче следующего символа.

Работая в режиме приемника, УАПП должен выполнить обратную последовательность операций.

- Принять данные в последовательной форме от внешнего интерфейса.
- Проверить правильность структуры кадра: стартовый бит, информационные разряды, бит четности, столовый бит; если нет — выдать сигнал ошибки кадрирования.
- Осуществить проверку четности (независимо от ее типа); если четность нарушена, выдать сигнал ошибки четности.
- Преобразовать информационные разряды в символ. Передать символ (как данные в параллельной форме) в системный интерфейс ввода-вывода.
- Сообщить во внешний интерфейс, что символ принят.

2.1.2.5.1. Микросхемы УАПП 8250 и 16450

Исходная системная плата IBM PC не содержала никаких специальных схем для осуществления последовательной связи. Эту функцию обеспечивал расположенный на отдельной плате адаптер, называемый *асинхронным коммуникационным адаптером* (фирмы IBM). Адаптер был построен на микросхеме УАПП INS8250 фирмы National Semiconductor.

За время, прошедшее с начала выпуска, микросхема INS8250 несколько раз модифицировалась: INS8250-A, INS8250-B, PC8250C. Все модификации микросхемы 8250 идентичны по большинству своих функциональных характеристик. Однако некоторые ранние модификации, хотя они и функционируют надлежащим образом в рамках паспортных технических условий, дают сбои при программировании на работу со скоростями передачи, превышающими 38400 бод. Такие микросхемы обычно используются в IBM/PC устаревших моделей и на устаревших асинхронных платах расширения.

Микросхема PC16450C — полный функциональный аналог микросхем NS16450 и INS8250A. По сравнению с последними она имела улучшенные временные характеристики и изготавливалась с использованием КМОП-технологии.

С точки зрения программиста УАПП 8250 и 16450 совершенно идентичны.

2.1.2.5.1.1. Характеристики и функциональные возможности

По замыслу микросхема 8250 предназначена для обеспечения интерфейса между системным интерфейсом IBM/PC и модемом или другими внешними устройствами. Микросхема 8250 «избавляет» микропроцессор от большей части «забот», связанных с обеспечением последовательной связи. УАПП, например, автоматически добавляет, проверяет и удаляет кадрирующие биты. Аналогично автоматически генерируются во время передачи и проверяются при приеме биты четности. Регистрируются любые ошибки четности, и сообщения о них передаются процессору. Микросхема 8250 вырабатывает набор отдельных указателей для состояний операций передачи и приема, а также указателей, сообщающих о состоянии линии и устройства сопряжения (модема). Эти указатели могут быть считаны процессором или запрограммированы на генерацию прерываний. Наличие встроенных регистров хранения данных и сдвиговых регистров для операций передачи и приема исключает необходимость точной синхронизации работы процессора с потоком последовательных данных.

Микросхема содержит программируемый генератор-контроллер скорости передачи в бодах, обеспечивающий деление частоты внешнего опорного сигнала (не более 8 МГц) на коэффициент, который может изменяться от 1 до 65535. Результирующий сигнал на выходе этого генератора представляет собой 16-фазную последовательность синхронизирующих импульсов, управляющих работой внутренних логических схем передатчика. Тот *же* самый 16-фазный синхронизирующий сигнал в принципе можно использовать в качестве опорного сигнала независимых часов приемника, для того чтобы гарантировать правильность выборки принимаемых данных.

Для упрощения аппаратных средств и обеспечения гибкого программного интерфейса микросхема 8250 содержит логические схемы обработки сигнала управления модемом, работа которых жестко связана с ее системой прерываний. Микросхема контролирует сигналы управления модемом и может быть запрограммирована на выдачу сообщений о каких-либо изменениях путем прерываний. Встроенные средства самотестирования включают циклические тесты для диагностики каналов связи и моделирования разрывов линии, ошибок четности, кадрирования и переполнения.

2.1.2.5.1.2. Блок-схема

Внутреннюю структуру микросхемы 8250 можно представить в виде набора схем синхронизации, управляющих и переключающих логических схем, а также 11 программно-доступных регистров. В совокупности эти структурные единицы образуют интерфейс между внутренними логическими схемами персонального компьютера и внешним миром систем последовательной связи. Эта идея физического интерфейса как раз и реализована в микросхеме 8250. Структурные составляющие данной микросхемы можно разбить на две функционально различные группы, связанные между собой общей шиной данных, по которой осуществляется обмен информацией. Это функциональное разделение иллюстрируется на рис. 2.17 в виде упрощенной блок-схемы, представляющей внутреннюю структуру УАПП 8250. Блок-схема показывает взаимосвязи между отдельными структурными составляющими и некоторые сигналы, которые

обычно передаются в том и другом направлениях.

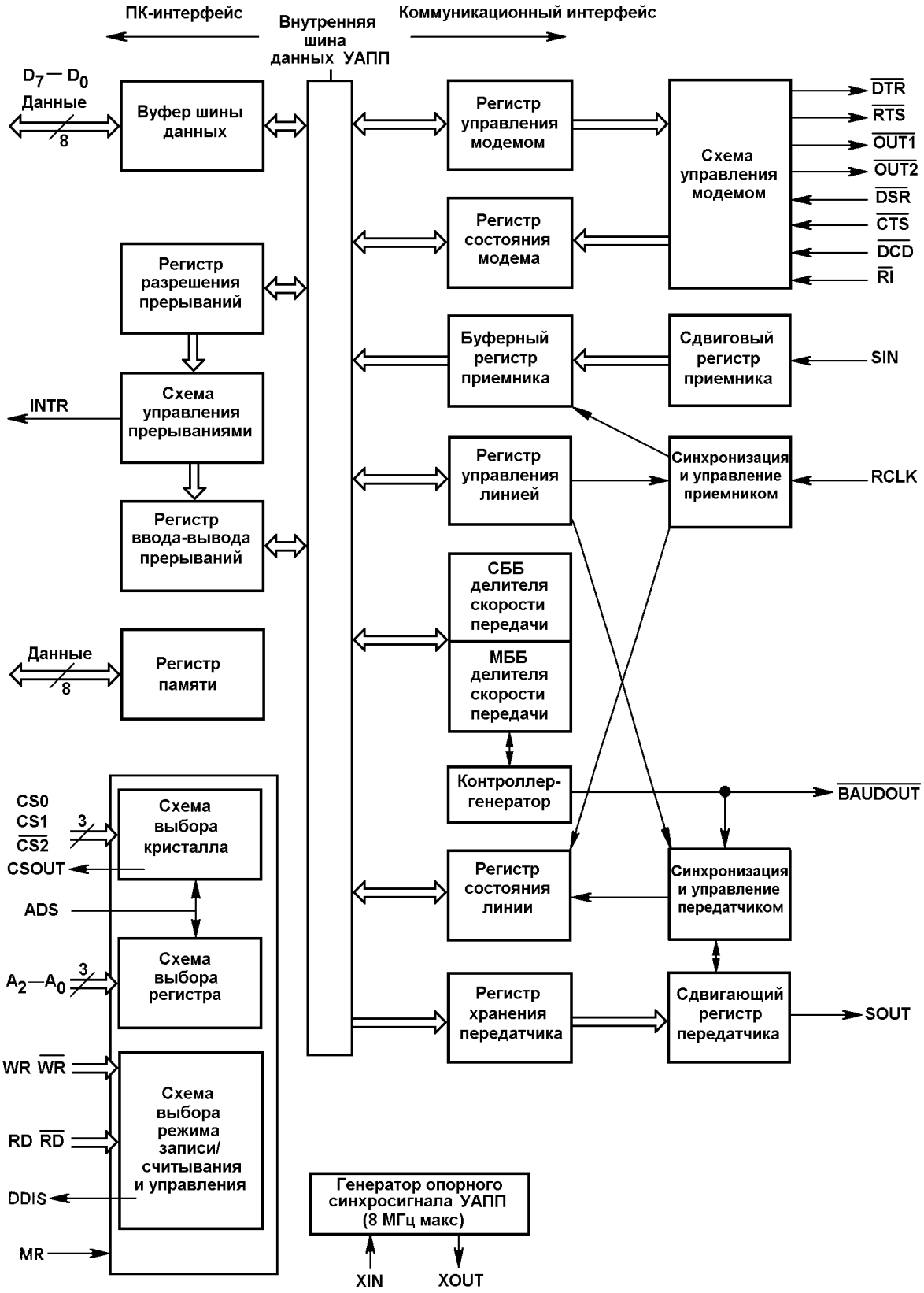

Рис. 2.17. Блок-схема УАПП 8250/16450

2.1.2.5.2. УАПП 16550

Микросхема 8250 значительно упрощала программирование систем последовательной связи. Работая со скоростями до 9600 бод, она выполняла свои функции превосходно, полностью соответствуя по своим характеристикам невысокому (в прошлом) быстродействию IBM PC. Но, несмотря на столь успешное применение в прошлом и почти универсальную популярность, с развитием техники связи микросхема 8250

стала «выглядеть» весьма посредственно. Чтобы исправить ситуацию, фирма National разработала и выпустила микросхему PC16550C/NS16550AF — улучшенную модификацию исходного УАПП типа 8250.

По умолчанию микросхема 16550 эмулирует микросхему 8250 и в большинстве случаев может быть установлена на месте последней. Работая в совместимом режиме, эта микросхема является полным функциональным аналогом УАПП 8250 и 16450. Ниже подробно описываются новые возможности и режимы работы микросхемы 16550.

В отличие от микросхемы 8250, микросхема 16550 имеет второй режим работы, предназначенный для сокращения вмешательства процессора в процедуру последовательной передачи данных. В этом режиме внутренние буферные регистры приемника и передатчика расширяются от 1 до 16 байтов и управляются с использованием логики «первым пришел — первым вышел», или FIFO-логики (First In, First Out). FIFO-буфер приемника используется также для хранения трех битов информации об ошибках для каждого символа. Ошибки четности, кадрирования и BREAK-сигналы буферизуются вместе с символом, к которому они относятся.

2.1.2.5.2.1. Характеристики и функциональные возможности

Микросхема 16550 совместима снизу вверх с микросхемами 8250 и 16450. Поэтому она обладает более широким набором характеристик и функциональных возможностей по сравнению с последними. Микросхема 16550 выполняет следующие функции.

- Обеспечивает простой интерфейс между системным интерфейсом и модемом или другим внешним устройством.
- Автоматически добавляет, удаляет и проверяет кадрирующие блоки
- Генерирует и проверяет биты четности под управлением специальной программы.
- Выделяет указатели состояния операций передачи и приема, а также состояния линии передачи данных и устройства сопряжения.
- Содержит встроенные сдвиговые регистры и регистры хранения для операций передачи и приема данных, что исключает необходимость точной синхронизации работы процессора с потоком последовательных данных.
- Содержит программируемый генератор-контроллер скорости передачи, работающий с внешним опорным сигналом частотой до 24 МГц.
- Содержит логические схемы обработки сигналов управления модемом, связанные с системой прерываний.
- Содержит встроенные средства самотестирования.
- Может работать под управлением программного обеспечения, разработанного для микросхем 8250 и 16450.
- Внутренние буферы позволяют хранить до 16 символов (и связанную с ними информацию об ошибках) при операциях передачи и приема данных.

Помимо этого 16550 имеет возможности поддержки обмена в режиме прямого доступа к памяти (DMA).

2.1.2.5.2.2. Блок-схема

В своей основе УАПП 16550 — это улучшенная модификация УАПП 8250. Поэтому внутренняя конфигурация микросхемы 16550 очень близка к конфигурации микросхемы 8250 и может быть представлена как совокупность схем синхронизации, логических схем управления и коммутации сигналов, буферов и 12 адресуемых регистров. Упрощенная блок-схема УАПП 16550 изображена на рис. 2.18. На этой блок-схеме указаны взаимосвязи между отдельными структурными единицами и сигналы на входах и выходах микросхемы.

Микросхема 16550 монтировалась в том же корпусе, что и микросхема 8250. Два вывода, однако, были переопределены для реализации процедуры DMA - прямого доступа к памяти (ПДП). В микросхеме 8250 вывод CSOUT обеспечивает обратную связь с системой, которая выбрала данную микросхему и готова для передачи данных. В микросхеме 16550 этот вывод предназначен для выдачи сигнала TXRDY. Когда в буфере передатчика нет никаких символов, этот сигнал имеет низкий уровень. TXRDY-сигнал остается в этом состоянии до тех пор, пока не заполнится буфер передатчика. Влияние такого переопределения данного вывода минимально, так как в стандартной реализации последовательного порта IBM PC с использованием микросхемы 8250 этот вывод никак не использовался.

Микросхема 16550 вырабатывает также один дополнительный сигнал для указания состояния буфера приемника. Этот RXRDY-сигнал имеет низкий уровень, если в буфере приемника находится хотя бы один символ. Низкий уровень RXRDY-сигнала сохраняется до тех пор, пока буфер приемника не будет пуст. Влияние этой замены также минимально, поскольку соответствующий вывод микросхемы 8250 в IBM PC никуда не подключается.

2.1.2.5.3. Регистры УАПП 8250, 16450 и 16550.

Микросхемы UART 8250, 16450 и 16550 с программной точки зрения представляют собой набор регистров, доступ к которым определяется адресом (смещением адреса регистра относительно базового

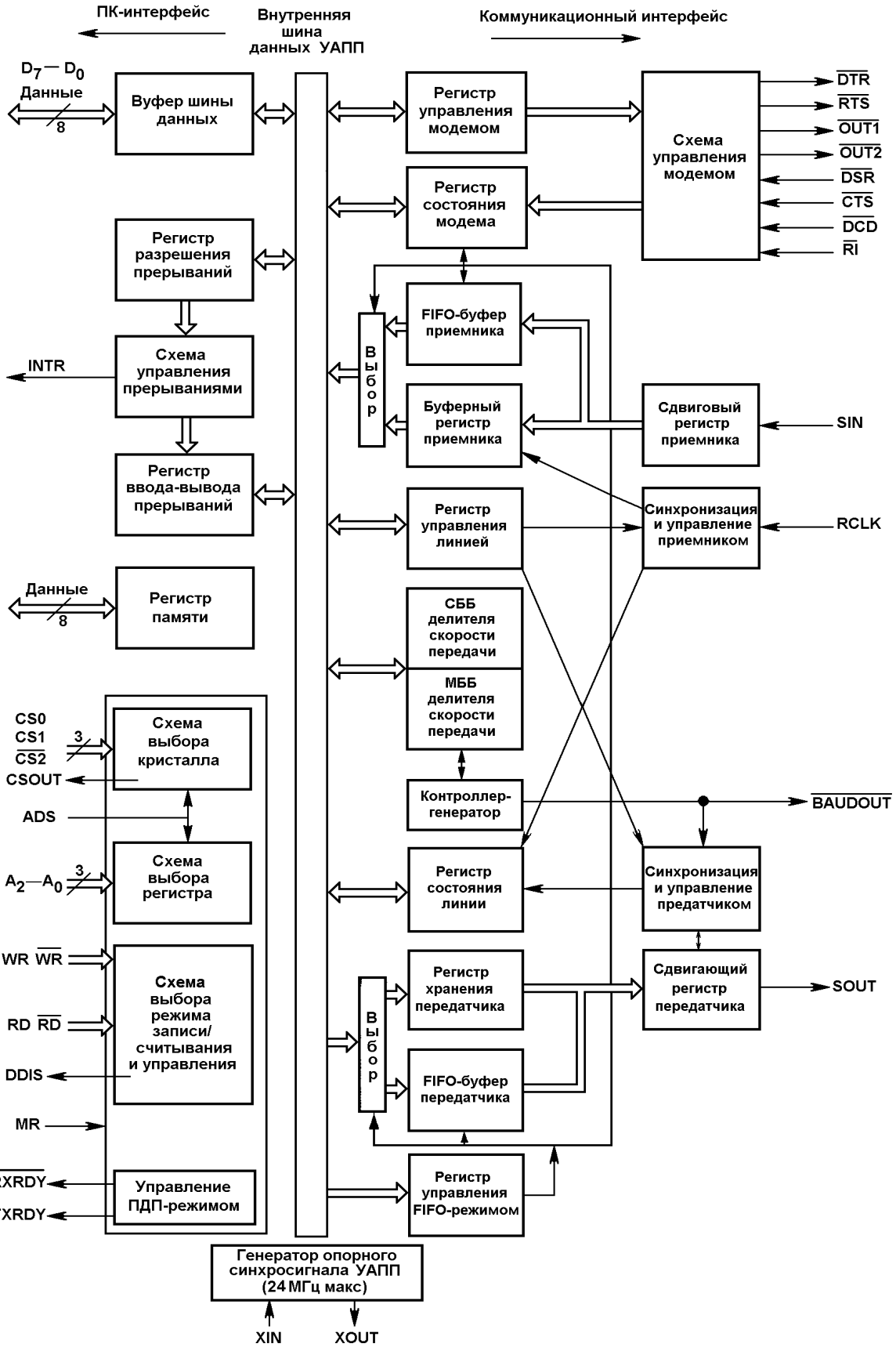

Рис. 2.18. Блок-схема УАПП 16550

адреса порта) и значением бита DLAB (бита 7 регистра LCR). В адресном пространстве микросхемы занимают 8 смежных адресов. Список регистров УАПП (UART) 16550 и способы доступа к ним приведены в табл. 2.4. Микросхемы 8250 и 16450 отличаются отсутствием регистра FCR и всех возможностей FIFO и DMA. Определения большинства полей-разрядов регистров микросхемы 16550 идентичны с определениями соответствующих разрядов регистров микросхем 8250 и 16450 как в режиме эмуляции УАПП 8250, так и в

FIFO-режиме.

Таблица 2.4. Регистры UART 16550A

Адрес (Смещение)	DLAB	Регистр Имя	Название	Чтение/запись R/W
0h	0	THR	Transmit Holding Register	WO
0h	0	RBR	Receiver Buffer Register	RO
0h	1	DLL	Divisor Latch LSB	R/W
1h	1	DLM	Divisor Latch MSB	R/W
1h	0	IER	Interrupt Enable Register	R/W
2h	X	IIR	Interrupt Identification Register	RO
2h	X	FCR	FIFO Control Register	WO
3h	X	LCR	Line Control Register	R/W
4h	X	MCR	Modem Control Register	R/W
5h	X	LSR	Line Status Register	R/W[1]
6h	X	MSR	Modem Status Register	R/W'
7h	X	SCR	Scratch Pad Register	R/W

[1] Некоторые биты допускают только чтение. Запись в регистр может привести к сбою протокола.

X - значение разряда DLAB безразлично.

RO - только чтение.

WO - только запись.

R/W - и запись и чтение.

Так, форматы перечисленных ниже регистров УАПП 16550 полностью совпадают с форматами регистров микросхем 8250 и 16450.

- Регистр хранения передатчика (РХП) (THR)
- Буферный регистр приемника (БРП) (RBR)
- Регистр управления линией (РУЛ) (LCR)
- Младший байт буфера делителя скорости передачи (МББ) (DLL)
- Старший байт буфера делителя скорости передачи (СББ) (DLM)
- Регистр сверхоперативной памяти (РСП) (CSR)
- Регистр управления модемом (РУМ) (MCR)
- Регистр состояния модема (РСМ) (MSR)

В форматы регистров:

- регистр состояния линии, РСЛ (LSR),
- регистр разрешения прерываний, РРП (IER),
- регистр идентификации прерываний, РИП (IIR)

внесены изменения в части переопределения функций некоторых разрядов при работе УАПП в режиме FIFO.

Для обеспечения управления режимами работы УАПП 16550 в его архитектуру был добавлен один новый регистр - регистр управления FIFO-режимом (FCR).

Ниже приводится описание регистров UART 8250, 16450 и 16550 и их форматов.

THR *— промежуточный регистр данных передатчика* (только для записи). Данные, записанные в регистр, будут пересланы в выходной сдвигающий регистр TSR (когда он будет свободен), из которого поступят на выход TD. Бит 0 передается первым. При длине посылки менее 8 бит старшие биты игнорируются.

RBR *— буферный регистр принимаемых данных* (только для чтения). Данные, принятые входным сдвигающим регистром RSR, помещаются в регистр RBR, откуда они могут быть считаны процессором. Если к моменту окончания приема очередного символа предыдущий символ не был считан из регистра RBR, фиксируется ошибка переполнения. При длине посылки менее 8 бит старшие биты в регистре имеют нулевое значение.

DLL *— регистр младшего байта буфера делителя частоты.*

DLM *— регистр старшего байта буфера делителя частоты.* Содержимое делителя (DLL и DLM) в десятичном виде определяется по формуле D=115200/V, где V — скорость передачи, бит/с. В рамках архитектуры PC AT использовалась входная частота синхронизации 1,8432 МГц, поступавшая в контроллер, делилась на заданный коэффициент D, после чего получалась 16-кратная скорость передачи данных (V×16). В регистры делителя всегда заносится шестнадцатеричный эквивалент коэффициента D. В таблице 2.5 приведены рекомендуемые скорости обмена COM порта и соответствующие значения десятичной и шестнадцатеричной константы деления D.

Фирма IBM выпускала высокоскоростные контроллеры последовательного порта, которые использовали входную частоту синхронизации 11,0592 МГц, и были способны передавать данные со скоростями до 345 600 бод.

Следует отметить, что для контроллера 16550A и его аналогов в рамках чипсетов существует недокументированная возможность дополнительного программного увеличения скорости обмена по сравнению с представленными в табл. 2.5 стандартными значениями. Эта возможность реализуется записью в регистр состояния линии контроллера (LSR) определенных управляющих байтов (например 80h). Используя дополнительное изменение кратности деления частоты можно расширить диапазон скоростей обмена, устанавливаемых функцией 00h драйвера COM порта INT 14h, который обычно ограничен значением 9600 бод.

Таблица 2.5. Соответствие скорости обмена и констант деления.

Требуемая скорость приема/передачи	Значение делителя, соответствующее требуемой скорости	
	Десятичное	Шестнадцатеричное
50	2304	0900h
75	1536	0600h
150	768	0300h
300	384	0180h
600	192	00C0h
1200	96	0060h
2400	48	0030h
4800	24	0018h
7200	16	0010h
9600	12	000Ch
19200	6	0006h
38400	3	0003h
57600	2	0002h
115200	1	0001h

IER — регистр разрешения прерываний. Единичное значение бита разрешает прерывание от соответствующего источника.

Назначение бит регистра IER:

- биты [7:4]=0 — не используются;
- бит 3 — Mod_IE — по изменению состояния модема (любой из линий CTS, DSR, RI, DCD);
- бит 2 — RxL_IE — по обрыву/ошибке линии;
- бит 1 — TxD_IE — по завершению передачи;
- бит 0 — RxD_IE — по приему символа (в режиме FIFO — прерывание по временной задержке).

В FIFO-режиме установка бита 0 в 1 разрешает прерывание по причине временной задержки в работе FIFO-буфера приемника. Если это прерывание разрешено, то оно будет происходить при одновременном выполнении следующих условий.

- Хотя бы один символ находится в FIFO-буфере приемника.
- Текущий символ не принят за время, превышающее длительность четырех непрерывно следующих друг за другом символов.
- Информация из FIFO-буфера приемника не считывалась процессором в течение времени, превышающего длительность четырех непрерывно следующих друг за другом символов.

Длительность символа измеряется числом синхронизирующих импульсов контроллера-генератора скорости передачи, так что пороговый уровень задержки не зависит от скорости передачи. Если условия этого прерывания не реализуются, то таймер задержки сбрасывается в исходное состояние каждый раз, когда символ принимается или считывается из FIFO-буфера приемника.

IIR — *регистр идентификации прерываний и признака режима FIFO* (только для чтения). Для упрощения программного анализа UART выстраивает внутренние запросы прерывания по четырехуровневой системе приоритетов. Порядок приоритетов (по убыванию): состояние линии, прием символа, освобождение регистра передатчика, состояние модема. При возникновении условий прерывания UART указывает на источник с высшим приоритетом до тех пор, пока он не будет сброшен соответствующей операцией. Только после этого будет выставлен запрос с указанием следующего источника. Ниже объясняется назначение бит регистра *IIR*.

- Биты [7:6] — признак режима FIFO:
  - 11 — режим FIFO 16550A;
  - 10 — режим FIFO 16550;
  - 00 — обычный.
- Биты [5:4] — не используются.
- Бит 3 — прерывание по временной задержке приема в режиме FIFO (в буфере есть символы для считывания).

- Биты [2:1] — причина прерывания с наивысшим приоритетом (в обычном, не FIFO-режиме):
  - 11 — ошибка/обрыв линии, сброс выполняется чтением регистра состояния линии;
  - 10 — принят символ, сброс выполняется чтением данных;
  - 01 — передан символ (регистр THR пуст), сброс выполняется записью данных;
  - 00 — изменение состояния модема; сброс выполняется чтением регистра состояния модема.
- Бит 0 — признак необслуженного запроса прерывания (1 — нет запроса, 0 — есть запрос).

В режиме FIFO причину прерывания идентифицируют биты [3:1].

- 011 — ошибка/обрыв линии. Сброс выполняется чтением регистра состояния линии.
- 010 — принят символ. Сброс выполняется чтением регистра данных приемника.
- 110 — индикатор временной задержки (за 4-кратный интервал времени символа не передано и не принято ни одного символа, хотя в буфере имеется, по крайней мере, один). Сброс выполняется чтением регистра данных приемника.
- 001 — регистр THR пуст. Сброс выполняется записью данных.
- 000 — изменение состояния модема (CTS, DSR, RI или DCD). Сброс выполняется чтением регистра *MSR*.

FCR *— регистр управления FIFO* (только для записи). Ниже описано назначение бит регистра FCR:

- Биты [7:6] — ITL (Interrupt Trigger Level) — уровень заполнения FIFO-буфера, при котором вырабатывается прерывание:
  - 00 — 1 байт (по умолчанию);
  - 01 — 4 байта;
  - 10 — 8 байт;
  - 11 — 14 байт.
- Биты [5:4] зарезервированы.
- Бит 3 — разрешение операций DMA.
- Бит 2 — RESETTF (Reset Transmitter FIFO) — сброс счетчика FIFO-передатчика (записью единицы; сдвигающий регистр не сбрасывается).
- Бит 1 — RESETRF (Reset Receiver FIFO) — сброс счетчика FIFO-приемника (записью единицы; сдвигающий регистр не сбрасывается).
- Бит 0 — TRFIFOE (Transmit And Receive FIFO Enable) — разрешение (единицей) режима FIFO для передатчика и приемника. При смене режима FIFO-буферы автоматически очищаются.

LCR *— регистр управления линией* (настройки параметров канала). Ниже описано назначение бит регистра *LCR*.

- Бит 7 — DLAB (Divisor Latch Access Bit) — управление доступом к делителю частоты.
- Бит 6 — BRCON (Break Control) — формирование обрыва линии (посылка нулей) при BRCON=1.
- Бит 5 — STICPAR (Sticky Parity) — принудительное формирование бита паритета:
  - 0 — контрольный бит генерируется в соответствии с паритетом выводимого символа;
  - 1 — постоянное значение контрольного бита: при EVENPAR=1 — нулевое, при EVENPAR=0 — единичное.
  - Бит 4 — EVENPAR (Even Parity Select) — выбор типа контроля: 0 — нечетность, 1 — четность.
  - Бит 3 — PAREN (Parity Enable) — разрешение контрольного бита:
  - 1 — контрольный бит (паритет или постоянный) разрешен;
  - 0 — контрольный бит запрещен.
- Бит 2 — STOPB (Stop Bits) — количество стоп-бит:
  - 0 — 1 стоп-бит;
  - 1 — 2 стоп-бита (для 5-битного кода стоп-бит будет иметь длину 1,5 бит).
- Биты [1:0] — SERIALDB (Serial Data Bits) — количество бит данных:
  - 00 — 5 бит;
  - 01 — 6 бит;
  - 10 — 7 бит;
  - 11 — 8 бит.

MCR *— регистр управления модемом.* Ниже описано назначение бит регистра MCR.

- Биты [7:5]=0 — зарезервированы.
- Бит 4 — LME (Loopback Mode Enable) — разрешение режима диагностики:

- 0 — нормальный режим;
- 1 — режим диагностики (см. ниже).

- Бит 3 — IE (Interrupt Enable) — разрешение прерываний с помощью внешнего выхода OUT2; в режиме диагностики поступает на вход MSR. 7:
  - 0 — прерывания запрещены;
  - 1 — прерывания разрешены.
- Бит 2 — OUT1C (OUT1 Bit Control) — управление выходным сигналом 1 (в PC не используется); в режиме диагностики поступает на вход MSR. 6.
- Бит 1 — RTSC (Request To Send Control) — управление выходом RTS; в режиме диагностики поступает на вход MSR. 4:
  - 0 — активен (-V);
  - 1 — пассивен (+V).
- Бит 0 — DTRC (Data Terminal Ready Control) — управление выходом DTR; в режиме диагностики поступает на вход MSR. 5:
  - 0 — активен (-V);
  - 1 — пассивен (+V).

LSR *—регистр состояния линии* (точнее, состояния приемопередатчика). Ниже описано назначение разрядов регистра LSR.

- Бит 7 - FIFOE (FIFO Error Status) — ошибка принятых данных в режиме FIFO (буфер содержит хотя бы один символ, принятый с ошибкой формата, паритета или обрывом). В не FIFO-режиме всегда 0.
- Бит 6 — TEMPT (Transmitter Empty Status) — регистр передатчика пуст (нет данных для передачи ни в сдвиговом регистре, ни в буферных регистрах THR или FIFO).
- Бит 5 — THRE (Transmitter Holding Register Empty) — регистр передатчика готов принять байт для передачи. В режиме FIFO указывает на отсутствие символов в FIFO-буфере передачи. Может являться источником прерывания.
- Бит 4 — BD (Break Detected) — индикатор обрыва линии (вход приемника находится в состоянии 0 не менее чем время посылки символа).
- Бит 3 — FE (Framing Error) — ошибка кадра (неверный стоп-бит).
- Бит 2 — PE (Parity Error) — ошибка контрольного бита (паритета или фиксированного).
- Бит 1 — OE (Overrun Error) — переполнение (потеря символа). Если прием очередного символа начинается до того, как предыдущий выгружен из сдвигающего регистра в буферный регистр или в регистр FIFO, прежний символ в сдвигающем регистре теряется.
- Бит 0 — DR (Receiver Data Ready) — принятые данные готовы (в DHR или FIFO-буфере). Сброс — чтением приемника.

Индикаторы ошибок — биты [4:1] — сбрасываются после чтения регистра LSR. В режиме FIFO признаки ошибок хранятся в FIFO-буфере вместе с каждым символом. В регистре они устанавливаются (и вызывают прерывание) в тот момент, когда символ, принятый с ошибкой, находится на вершине FIFO (первый в очереди на считывание). В случае обрыва линии в FIFO заносится только один «обрывной» символ, и UART ждет восстановления и последующего старт-бита.

Примечание: Регистр LSR может быть использован для реализации недокументированной возможности некоторых реализаций контроллеров COM портов - увеличение скорости передачи в 16 раз по сравнению с устанавливаемой в счетчике делителя контроллера. Эта возможность реализуется записью в регистр LSR константы 80h. Скорость восстанавливается записью в LSR константы 00h.

MSR — *регистр состояния модема.* Ниже описано назначение бит регистра MSR:

- Бит 7 — DCD (Data Carrier Detect) — состояние линии DCD:
  - 0 — активна (-V);
  - 1 — пассивна (+V).
- Бит 6 — RI (Ring Indicator) — состояние линии RI:
  - 0 — активна (-V);
  - 1 — пассивна (+V).
- Бит 5 — DSR (Data Set Ready) — состояние линии DSR:
  - 0 — активна (-V);
  - 1 — пассивна (+V).
- Бит 4 — CTS (Clear To Send) — состояние линии CTS:
  - 0 — активна (-V);
  - 1 — пассивна (+V).
- Бит 3 — DDCD (Delta Data Carrier Detect) — изменение состояния DCD.

- Бит 2 — TERI (Trailing Edge Of Ring Indicator) — спад огибающей RI (окончание звонка).
- Бит 1 — DDSR (Delta Data Set Ready) — изменение состояния DSR.
- Бит 0 — DCTS (Delta Clear To Send) — изменение состояния CTS.

Признаки изменения (биты [3:0]) сбрасываются по чтению регистра.

SCR *— рабочий регистр* (8 бит), на работу UART не влияет, предназначен для временного хранения данных (в 8250 отсутствует).

В *диагностическом режиме* (при LME=1 в регистре MCR) внутри UART организуется внутренняя «заглушка»:

- выход передатчика переводится в состояние логической единицы;
- вход приемника отключается;
- выход сдвигающего регистра передатчика логически соединяется со входом приемника;
- входы DSR, CTS, RI и DCD отключаются от входных линий и внутренне управляются битами DTRC, RTSC, OUT1C, IE;
- выходы управления модемом переводятся в пассивное состояние (логический ноль).

Переданные данные в последовательном виде немедленно принимаются, что позволяет проверять внутренний канал данных порта (включая сдвигающие регистры) и отработку прерываний, а также определять скорость работы UART.

2.1.2.6. Системная поддержка COM портов

COM-порты на уровне BIOS поддерживаются следующими средствами:

- процедурами POST на стадии инициализации и тестирования;
- сервисом BIOS Int 14h (драйвер COM-порта).

Системная BIOS не поддерживает аппаратные прерывания RS232, хотя имеются два вектора прерываний: IRQ4 (INT 0Ch) для RS232-1 и IRQ3 (INT 0Bh) для RS232-2.

В процессе начального тестирования POST BIOS проверяет наличие последовательных портов (регистров UART 8250 или совместимых) по стандартным адресам и помещает базовые адреса обнаруженных портов в ячейки BIOS Data Area (ячейки памяти рабочей области данных BIOS) 0:0400, 0402, 0404, 0406. Эти ячейки хранят адреса портов с логическими именами COM1 - COM4 (в сервисе Int 14h - логические номера 0 - 3). Нулевое значение адреса является признаком отсутствия порта с данным номером. При программировании RS232 необходимо пользоваться не жестко привязанными адресами, а адресами, извлеченными из области данных BIOS. В ячейки памяти 0:047C, 047D, 047E, 047F заносятся константы, задающие тайм-ауты для соответствующих портов. Значения этих тайм-аутов инициализируются на 1 при выполнении POST и могут быть установлены пользователями до значения 255 единиц. Каждая единица соответствует приблизительно 900 мс. Обнаруженные порты *инициализируются* на скорость обмена 2400 бит/с, 7 бит данных с контролем на четность (even), 1 стоп-бит. Управляющие сигналы интерфейса DTR и RTS и линия передачи данных TD переводятся в исходное состояние («включено» — отрицательное напряжение -12 В на соответствующих контактах интерфейсного разъема).

Сервис BIOS Int 14h обеспечивает выполнение следующих функций:

- *инициализация* (установка скорости обмена и формата посылок; запрет источников прерываний) — на сигналы DTR и RTS влияния не оказывает (после аппаратного сброса они пассивны);
- *вывод символа* — активируются сигналы DTR и RTS, и после освобождения регистра THR в него помещается выводимый символ;
- *ввод символа* — активируется только сигнал DTR (RTS переходит в пассивное состояние), и ожидается готовность принятых данных;
- *опрос состояния* модема и линии (чтение регистров MSR и LSR).

Ожидание готовности ввода и вывода ограничивается по тайм-ауту. Готовность проверяется опросом состояния.

В таблице 2.6 приведены системные ресурсы, выделяемые под стандартные COM-порты в ПК типа IBM PC/AT.

Таблица 2.6. Ячейки памяти рабочей области данных BIOS, используемые для COM-портов.

Ячейка памяти	Байт	Функция
0000:002C	4	Вектор INT 0Bh
0000:0030	4	Вектор INT 0Ch
0000:0050	4	Вектор INT 14h
0040:0000	2	Адрес базового порта RS232-1
0040:0002	2	Адрес базового порта RS232-2
0040:0004	2	Адрес базового порта RS232-3
0040:0006	2	Адрес базового порта RS232-4

0040:0071	1	Break bit
0040:007C	1	Тайм-аут для RS232-1
0040:007D	1	Тайм-аут для RS232-2
0040:007E	1	Тайм-аут для RS232-3
0040:007F	1	Тайм-аут для RS232-4

2.1.2.6.1. Драйвер COM-порта Int 14h.

Int 14h вызывается для выполнения всех функций, относящихся к вводу-выводу через RS232. BIOS указывает на точку входа в ПЗУ драйвера RS232.

ДЕЙСТВИЕ: после входа управление передается одной из четырех программ в соответствии с кодом функции в регистре AH. При использовании запрещенного кода функции происходит простой возврат управления. Сохраняются все регистры, кроме тех, которые возвращают значение.

НАЗНАЧЕНИЕ: прикладная программа может использовать Int 14h для прямого выполнения функций ввода-вывода через RS232 COM-порты).

По команде Int 14h выполняются следующие функции:

- 00h - нициализировать RS232;
- 01h - передать символ;
- 02h - принять символ;
- 03h - определить состояние.

При выполнениии функций прерывания остаются разрешенными. Маска для прерываний интерфейса и порт управления для прерываний RS232 не изменяются.

Перед выполнением команды Int14h необходимо занести определенную информацию в регистры процессора в соответствии с требованиями вызываемой функции. Результаты выполнения функции, если это предусмотрено функцией возвращаются программе также через регистры процессора.

Функция 00h: Инициализировать RS232

В функции указывается длина слова, число стоповых разрядов, тип паритета и скорость передачи для заданного порта.

Хотя BIOS поддерживает четыре COM-порта, при инициализации после включения питания BIOS ищет только порты 1 и 2. Для использования портов 3 и 4 пользователь должен установить базовые адреса интерфейсного порта в ячейки памяти 0040:0004h и 0040:0006h соответственно.

ВХОДНЫЕ ПАРАМЕТРЫ:
AH = 00h,
AL - конфигурация порта,
DX - номер RS232 (0 - 3).
ВЫХОДНЫХ ПАРАМЕТРОВ не предусмотрено.
В таблице 2.7 приведено назначение разрядов байта конфигурации (регистр AL).

Таблица 2.7. Байт конфигурации.

Разряды AL							
7	6	5	4	3	2	1	0
Скорость передачи: 000 - 110 бод 001 - 150 бод 010 - 300 бод 011 - 600 бод 100 - 1200 бод 101 - 2400 бод 110 - 4800 бод 111 - 9600 бод			Паритет: 00 - отсутствует 01 - нечетный 10 - отсутствует 11 - четный		Стоповые биты: 0 - 1 бит 1 - 2 бита	длина слова: 00 - 5 бит 01 - 6 бит 10 - 7 бит 11 - 8 бит	

Функция 01h Передать символ

INT 14h, AH = 01h выводит символ на заданный RS232.
ВХОДНЫЕ ПАРАМЕТРЫ:
AH = 01h,
AL - символ, который необходимо передать (00h.-.FFh),
DX - номер RS232 (0 - 3).
ВЫХОДНЫЕ ПАРАМЕТРЫ:
AH - состояние линии (см. определение для INT 14h, AH = 03h).

Функция 02h Принять символ

INT 14h, AH = 02h принимает символ с заданного RS232.
ВХОДНЫЕ ПАРАМЕТРЫ:
AH = 02h,

DX - номер RS232 (0 - 3).

ВЫХОДНЫЕ ПАРАМЕТРЫ:

AH - состояние линии (определение см. INT 14h, AH = 03h),

AL - символ, который принят (00h - FFh).

Функция 03h Определить состояние интерфейса

INT 14h, AH = 03h считывает регистры модема и состояния линии заданного RS232 и помещает эти значения в выходные регистры.

ВХОДНЫЕ ПАРАМЕТРЫ:

AH = 03h,

DX - номер RS232 (0 - 3).

ВЫХОДНЫЕ ПАРАМЕТРЫ:

AH - состояние линии,

AL - состояние модема.

В таблицах 2.8 и 2.9 приведено описание разрядов слов состояния линии и модема соответственно.

Таблица 2.8. Слово состояния линии.

Разряды AH							
7	6	5	4	3	2	1	0
1 - тайм-аут	1 - передатчик пуст	1 - готов к передаче	1 – пауза	1 - ошибка стопразряда	1 - ошибка паритета	1 - пере полнение	1 - готовы данные

Таблица 2.9. Слово состояния модема.

Разряды AL							
7	6	5	4	3	2	1	0
1 – детектор принимаемого сигнала (цепь 109)	1 - индикатор вызова (цепь 125)	1 - аппаратура передачи данных готова (цепь 107)	1 – готов к передаче (цепь 106)	0	0	0	0

2.1.2.7. Конфигурирование COM-портов

Компьютер может иметь до четырех последовательных портов COM1-COM4; для машин класса AT типично наличие двух портов. Управление последовательным портом разделяется на два этапа — предварительное конфигурирование (Setup) аппаратных средств порта и текущее (оперативное) переключение режимов работы прикладным или системным ПО. Конфигурирование COM-порта зависит от его исполнения. Порт на плате расширения конфигурируется джамперами на самой плате. Порт на системной плате конфигурируется через BIOS Setup.

Конфигурированию подлежат перечисленные ниже параметры:

- *Базовый адрес,* который для портов COM1-COM4 обычно имеет значение 3F8h, 2F8h, 3E8h и 2E8h. При инициализации BIOS проверяет наличие портов по адресам именно в этом порядке и присваивает обнаруженным портам •логические имена COM1, COM2, COM3 и COM4. Для портов COM3 и COM4 возможны альтернативные адреса 3E0h, 338h и 2E0h, 238h соответственно. Для PS/2 стандартными для портов COM3-COM8 являются адреса 3220h, 3228h, 4220h, 4228h, 5220h и 5228h соответственно.
- Используемая *линия запроса прерывания:* для COM 1 и COM3 обычно используется IRQ4 или IRQ11, для COM2 и COM4 — IRQ3 или IRQ10. В принципе номер прерывания можно назначать в произвольных сочетаниях с базовым адресом (номером порта), но некоторые программы и драйверы (например, драйверы последовательной мыши) настроены на стандартные сочетания. Каждому порту, нуждающемуся в аппаратном прерывании, назначают отдельную линию, не совпадающую с линиями запроса прерываний других устройств. Прерывания необходимы для портов, к которым подключаются устройства ввода, UPS (источники бесперебойного питания) или модемы. При подключении принтера или плоттера прерываниями пользуются только многозадачные ОС (не всегда), и этот дефицитный ресурс PC можно сэкономить. Также прерывания обычно не задействуют при связи двух компьютеров нуль-модемным кабелем. Возможность разделяемого использования одной линии запроса несколькими портами (или ее разделения с другими устройствами) зависит от реализации аппаратного подключения и ПО. При использовании портов, установленных на шину ISA, разделяемые прерывания обычно не работают.
- *Канал DMA* (для микросхем UART 16450/16550, расположенных на системной плате) — разрешение использования и номер канала DMA. Режим DMA при работе с COM портами используют редко.

2.1.2.8. Использование COM портов

COM-порт широко применяется для подключения различных периферийных и коммуникационных устройств, связи с различным технологическим оборудованием, объектами управления и наблюдения, программаторами, внутрисхемными эмуляторами и прочими устройствами, используя протокол RS-232C. COM-порты чаще всего применяют для подключения *манипуляторов* (мышь, трекбол). В этом случае порт используется в режиме последовательного ввода. Мышь с последовательным интерфейсом — *Serial Mouse* —

может подключаться к любому исправному порту. Для согласования разъемов порта и мыши возможно применение переходника DB-9S-DB-25P или DB-25S-DB-9P. Для мыши требуется прерывание, для порта COM1 — IRQ4, для COM2 — IRQ3. То, что для использования мыши порт COM1 должен использовать прерывание IRQ4, является особенностью ее драйвера, но для пользователя важен сам факт ограничения. Каждое событие — перемещение мыши или нажатие-отпускание кнопки — кодируется двоичной посылкой по интерфейсу RS-232C. Применяется асинхронная передача; двуполярное питание обеспечивается от управляющих линий интерфейса.

Для подключения *внешних модемов* используется полный (9-проводный) кабель *АПД-АКД,* схема которого приведена на рис. 2.4. Этот же кабель используется для согласования разъемов (по количеству контактов); возможно применение переходников 9 - 25, предназначенных для мыши. Для работы коммуникационного ПО обычно требуются прерывания, но здесь есть свобода выбора номера (адреса) порта и линии прерывания. Если предполагается работа на скоростях 9600 бит/с и выше, то COM-порт должен быть реализован на микросхеме UART 16550A или совместимой. Возможности работы с использованием FIFO-буферов и обмена по каналам DMA зависят от коммуникационного ПО.

Для связи *двух компьютеров,* удаленных друг от друга на небольшое расстояние, используют и непосредственное соединение их COM-портов нуль-модемным кабелем (см. рис. 2.5). Программы типа Norton Commander или Interink MS-DOS позволяют обмениваться файлами со скоростью до 115,2 Кбит/с без применения аппаратных прерываний. Это же соединение может использоваться и сетевым пакетом Lantastic, предоставляющим более развитый сервис, и средствами ОС Windows.

COM-порт позволяет выполнять подключение *электронных ключей* (Security Devices), предназначенных для защиты от нелицензированного использования ПО. Эти устройства могут быть как «прозрачными» (обеспечивающими возможность подключения периферии через тот же порт), так и полностью занимающими порт.

COM-порт при наличие соответствующей программной поддержки позволяет превратить PC в *терминал,* эмулируя систему команд распространенных специализированных терминалов (VT-52, VT-100 и т. д.). Простейший терминал получается, если замкнуть друг на друга функции BIOS обслуживания COM-порта (Int 14h), телетайпного вывода (Int 10h) и клавиатурного ввода (Int 16h). Однако такой терминал будет работать лишь на малых скоростях обмена (если, конечно, его делать не на Pentium), поскольку функции BIOS хоть и универсальны, но не слишком быстры.

COM-порт может использоваться и как двунаправленный интерфейс, у которого имеется три программно-управляемых выходных линии (TD, DTR, RTS) и четыре программно-читаемых входных линии (CTS, DSR, DCD, RI) с двуполярными сигналами. Их можно использовать, например, для программной реализации синхронных последовательных интерфейсов и других целей. Во времена AT-286 была известна схема однобитного широтно-импульсного преобразователя, позволяющего записывать звуковой сигнал на диск PC, используя входную линию COM-порта. Воспроизведение этой записи через обычный динамик PC позволяло передать речь.

2.1.2.9. COM-порт и PnP

Современные ПУ, подключаемые к COM-порту, могут поддерживать спецификацию PnP. Основная задача ОС заключается в идентификации подключенного устройства, для чего разработан несложный протокол, реализуемый на любых COM-портах чисто программным способом. Этот протокол иллюстрирует рис. 2.19.

1. Порт инициализируется с состоянием линий DTR=ON, RTS=OFF, TD=Mark — состояние покоя (Idle).
2. Некоторое время (0,2 с) ожидается появление сигнала DSR, которое указало бы на наличие устройства, подключенного к порту. В простейшем случае устройство имеет на разъеме перемычку DTR-DSR, обеспечивающую указанный ответ. Если устройство обнаружено, выполняются манипуляции управляющими сигналами DTR и RTS для получения информации от устройства. Если ответ не получен, ОС, поддерживающая динамическое реконфигурирование, периодически опрашивает состояние порта для обнаружения новых устройств.
3. Порт программируется на режим 1200 бит/с, 7 бит данных, без паритета, 1 стоп-бит, и на 0,2 с снимается сигнал DTR. После этого устанавливается DTR= 1, а еще через 0,2 с устанавливается и RTS=1.

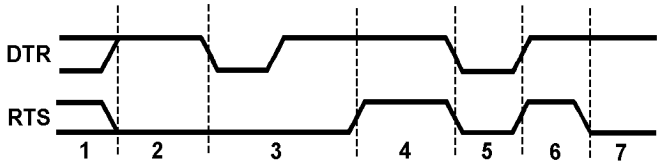

Рис. 2.19. Запрос идентификатора устройства PnP

4. В течение 0,2 с ожидается приход первого символа от устройства. По приходу символа начинается прием идентификатора (см. ниже). Если за это время символ не пришел, выполняется вторая попытка опроса (см. п. 5), несколько отличающаяся от первой.

5. На 0,2 с снимаются оба сигнала (DTR-0 и RTS-0), после чего они оба устанавливаются (DTR=1 и RTS=1).
6. В течение 0,2 с ожидается приход первого символа от устройства, по приходу символа начинается прием идентификатора (см. ниже). Если за это время символ не пришел, то в зависимости от состояния сигнала DSR переходят к проверке отключения *Verify Disconnect* (при DSR=0) или в дежурное состояние *Connect Idle* (при DSR=1).
7. В дежурном состоянии *Connect Idle* устанавливается DTR=1, RTS=0, порт программируется на режим 300 бит/с, 7 бит данных, без паритета, 1 стоп-бит. Если в этом состоянии обнаружится DSR=0, ОС следует уведомить об отключении устройства.

Посимвольный прием идентификатора устройства имеет ограничения по тайм-ауту в 0,2 с на символ, а также общее ограничение в 2,2 с, позволяющее принять строку длиной до 256 символов. Строка идентификатора PnP должна иметь маркеры начала (28h или 08h) и конца (29h или 09h), между которыми располагается тело идентификатора в стандартизованном формате. Перед маркером начала может находиться до 16 символов, не относящихся к идентификатору PnP. Если за первые 0,2 с ожидания символа (шаг 4 или 6) маркер начала не пришел, или же сработал тайм-аут, а маркер конца не получен, или же какой-либо символ принят с ошибкой, происходит переход в состояние *Connect Idle.* Если получена корректная строка идентификатора, она передается ОС.

Для *проверки отключения (Verify Disconnect)* устанавливается DTR=1, RTS-0 и через 5 с проверяется состояние сигнала DSR. При DSR=1 происходит переход в состояние *Connect Idle* (см. п. 7), при DSR=0 происходит переход в состояние *Disconnect Idle,* в котором система может периодически опрашивать сигнал DSR для обнаружения подключения устройства.

Описанный механизм разрабатывался компанией Microsoft с учетом совместимости с устройствами, не относящимися к устройствам PnP, — он обеспечивает невозможность их вывода из строя и устойчивость системы к сообщениям, не являющимся PnP-идентификаторами. Например, обычная мышь Microsoft Mouse (не PnP) по включении питания ответит ASCII-символом «M» (трехкнопочная — строкой «M3»).

2.1.2.10. Неисправности и тестирование COM портов

Неполадки с COM портами случаются (выявляются) при установке новых портов или неудачном подключении внешних устройств.

2.1.2.10.1. Проверка конфигурирования

Тестирование последовательных портов (как и параллельных) начинают с проверки их опознавания системой. Список *адресов* установленных портов обычно появляется в таблице, выводимой BIOS перед загрузкой ОС. Список можно посмотреть с помощью тестовых программ или прямо в BIOS Data AREA с помощью отладчика.

Если BIOS обнаруживает меньше портов, чем установлено физически, значит, двум портам присвоен один адрес или установлен нестандартный адрес какого-либо порта. Проблемы могут возникать с адресами портов COM3 и COM4: не все версии BIOS будут искать порты по альтернативным адресам 3E0h, 338h, 2E0h и 238h; иногда не производится поиск по адресам 3E8h и 2E8h. Нумерация найденных портов, отображаемая в заставке, может вводить в заблуждение: если установлены два порта с адресами 3F8h и 3E8h, в заставке они могут называться COM1 и COM2, и по этим именам на них можно ссылаться. Однако те же порты в заставке могут называться COM1 и COM3 (поскольку 3E8h является штатным адресом для COM3), но попытка сослаться на порт COM3 будет неудачной, поскольку в данном случае адрес 3E8h будет находиться в ячейке 0:402h BIOS Data Area, соответствующей порту COM2, а в ячейке порта COM3 (0:404h) будет нуль — признак отсутствия такового порта. «Объяснить» системе, где какой порт, можно вручную с помощью любого отладчика, занеся правильные значения базовых адресов в ячейки BIOS Data Area (это придется делать каждый раз после перезагрузки ОС перед использованием «потерянного» порта). Существуют тестовые утилиты, позволяющие находить порты (например, Port Finder). Если двум портам назначен один и тот же адрес, тестовая программа обнаружит ошибки порта только с помощью *внешней заглушки (External LoopBack).* Программное тестирование порта без заглушки не покажет ошибок, поскольку при этом включается диагностический режим (см. описание UART) и конфликтующие (по отдельности исправные) порты будут работать параллельно, обеспечивая совпадение считываемой информации. В «реальной жизни» нормальная работа конфликтующих портов невозможна. Разбираться с конфликтом адресов удобно, последовательно устанавливая порты и наблюдая за адресами, появляющимися в списке.

Если физически установлен только один порт и его не обнаруживает BIOS, причины те же, что с LPT-портом: либо он отключен при конфигурировании, либо вышел из строя. Неисправность может устраниться при вынимании/вставке платы адаптера в слот системной шины.

При работе с COM-портом задействуются соответствующие *аппаратные прерывания* — их используют при подключении модема, мыши и других устройств ввода. Неработоспособность этих устройств может быть вызвана некорректной настройкой запроса прерывания. Здесь возможны как конфликты с другими устройствами, так и несоответствие номера прерывания адресу порта.

2.1.2.10.2. Функциональное тестирование

В первом приближении COM-порт можно проверить диагностической программой (Checkit) без использования заглушек. Этот режим тестирования проверяет микросхему UART (внутренний диагностический режим) и вырабатывание прерываний, но не входные и выходные буферные микросхемы, которые являются более частыми источниками неприятностей. Если тест не проходит, причину следует искать или в конфликте адресов/прерываний, или в самой микросхеме UART.

Для более достоверного тестирования рекомендуется использовать *внешнюю заглушку,* подключаемую к разъему COM-порта (рис. 2.20). В отличие от LPT-порта у COM-порта количество входных сигналов превышает количество выходных, что позволяет выполнить полную проверку всех цепей. Заглушка соединяет выход приемника с входом передатчика. Обязательная для всех схем заглушек перемычка RTS-CTS позволяет работать передатчику — без нее символы не смогут передаваться с помощью драйвера COM-порта INT 14h. Выходной сигнал DTR обычно используют для проверки входных линий DSR, DCD и RI.

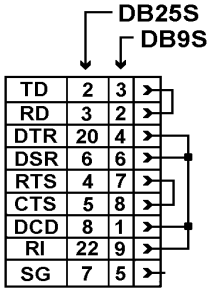

Рис. 2.20. Заглушка для проверки COM-портов (LoopBack для Checkit и Norton Diagnostics)

Если тест с внешней заглушкой не проходит, причину следует искать во внешних буферах, их питании или в ленточных кабелях, служащих для подключения внешних разъемов. Здесь может помочь осциллограф или вольтметр. Последовательность проверки может быть следующей:

1. Проверить наличие двуполярного питания выходных схем передатчиков (этот шаг логически первый, но поскольку он технически самый сложный, его можно отложить на крайний случай, когда появится желание заменить буферные микросхемы).
2. Проверить напряжение на выходах TD, RTS и DTR: после аппаратного сброса на этих выходах должен быть отрицательный потенциал около -12 В (по крайней мере, ниже -5 В). Если этих потенциалов нет, возможна ошибка подключения разъема к плате через ленточный кабель. Распространенные варианты:
   - ленточный кабель не подключен;
   - ленточный кабель подключен неправильно (разъем перевернут или вставлен со смещением);
   - раскладка ленточного кабеля не соответствует разъему платы.

   Первые два варианта проверяются внимательным осмотром, третий же может потребовать некоторых усилий. В табл. 2.1 приведены три возможные варианта раскладки 10-проводного ленточного кабеля для разъема COM-порта. Возможно существование и других вариантов. Эти ленточные кабели поставлялись в комплекте материнских плат.

   Если дело в ошибочной раскладке, то эти три выходных сигнала удастся обнаружить на других контактах разъемов (на входных контактах потенциал совсем небольшой). Если эти сигналы обнаружить не удалось, очевидно, вышли из строя буферные формирователи и их необходимо заменить.

Замена микросхем приемников и передатчиков существенно облегчается, если они установлены в «кроватки». Перед заменой следует с помощью осциллографа или вольтметра удостовериться в неисправности конкретной микросхемы.

Если буферные элементы включены в состав интерфейсной БИС (что теперь весьма распространено), то такой порт ремонту не подлежит (по крайней мере, в обычных условиях). Неисправный COM-порт, установленный на системной плате, можно попытаться отключить в BIOS SETUP, но порт мог сгореть и вместе со схемой своего отключения — тогда он останется «живым мертвецом» в карте портов ввода-вывода и прерываний. Иногда он полностью выводит из строя системную плату.

Источниками ошибок могут являться *разъемы* и *кабели.* В разъемах встречаются плохие контакты, а кабели, кроме возможных обрывов, могут иметь плохие частотные характеристики. Частотные свойства кабелей обычно сказываются при большой длине (десятки метров) на высоких скоростях обмена (56 или 115 Кбит/с). При необходимости использования длинных кабелей на высоких скоростях сигнальные провода данных должны быть перевиты с отдельными проводами «схемной земли».

В ряде отечественных PC-совместимых (почти) компьютеров (например, Искра 1031) для последовательного интерфейса применялась микросхема КР580ВВ51 — аналог i8251. Однако эта микросхема является универсальным синхронно-асинхронным приемопередатчиком (УСАПП или USART — Universal Asynchronous Receiver-Transmitter). Совместимости с PC на уровне регистров COM-порта такие компьютеры не имеют. Хорошо, если у соответствующих компьютеров имеется «честный» драйвер BIOS Int 14h, а не заглушка, возвращающая состояние модема «всегда готов» и ничего не делающая.

2.1.2.10.3. Питание от интерфейса, или причины неработоспособности мыши

При подключении к COM-порту устройств с небольшим энергопотреблением возникает соблазн получить питание от выходных линий интерфейса. Если линии управления DTR и RTS не используются по прямому назначению, их можно задействовать как питающие с напряжением около 12В. Ток короткого замыкания на «схемную землю» ограничен буферной микросхемой передатчика на уровне 20 мА. При инициализации порта эти линии переходят в состояние «включено», то есть вырабатывают *отрицательное* напряжение. Линия TD в покое находится в состоянии логической единицы, так что на выходе вырабатывается также *отрицательное* напряжение. Потенциалами линий можно управлять через регистры COM-порта (выход TD вырабатывает положительное напряжение, если установить бит BRCON регистра LCR). Питание можно получать и с сигнальных линий через выпрямительные диоды с использованием накопительных конденсаторов. При этом, конечно, следует учитывать, сколько времени выходной сигнал находится в нужном состоянии (чтобы накопленной энергии хватало).

Двуполярным питанием от линий интерфейса (+V от DTR и RTS, -V от TD) пользуются все манипуляторы, подключаемые к COM-портам. Зная это, в случае неработоспособности мыши с данным портом следует проверить напряжения на соответствующих контактах разъема. Бывает, что с конкретным портом не работает только конкретная мышь (модель или экземпляр), хотя другие мыши с этим портом и эти же мыши с другими портами работают нормально. Здесь дело может быть в уровнях напряжений. Стандарт требует от порта выходного напряжения не менее 5 В (абсолютного значения), и если данный порт обеспечивает только этот минимум, некоторым мышам не хватит мощности для питания светодиодов (главных потребителей энергии).

Порт получает двуполярное питание через системную плату от блока питания компьютера. Отсутствие на выходе блока питания напряжения +12 В обычно обнаружится по неработоспособности дисков. Отсутствие напряжения -12 В «заметят» только устройства, подключенные к COM-портам. Блок питания теоретически контролирует наличие этих напряжений на своем выходе (сообщая о неполадках сигналом Power Good, вызывающим аппаратный сброс). Встречаются упрощенные схемы блоков питания, у которых контролируются не все напряжения. Кроме того, возможны плохие контакты в разъеме подключения питания к системной плате.

2.1.3.Форматы передачи данных манипуляторами типа 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 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 таблице 2.10. 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 2.10. 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 введен для двухкнопочной мыши. Средняя кнопка трехкнопочной 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 произошел сбой в процессе передачи и следует проигнорировать принятый пакет.

Скорость приема-передачи данных принята равной 1200 бод, длина передаваемого слова - 7 бит, контроль по четности 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 таблице 2.11.

Таблица 2.11. 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[1]	0	M	0	0	0	0	0

[1] Передача четвертого байта производится только в случае изменения состояния средней кнопки мыши.

Конкуренция между изготовителями компьютерного оборудования приводит к тому, что на рынке появляется все больше «навороченных» устройств, снабженных рядом дополнительных функций. Дополнительные возможности требуют, естественно, передачи дополнительной информации от устройства к компьютеру. Например, формат данных трехкоординатной (3D) мыши, показанный в таблице 2.12, включает координату Z (Z0-Z3, целое число со знаком) и состояние специальной кнопки R_0.

Существует группа универсальных устройств, которые поддерживают два типа протоколов - PC Mouse и MS Mouse. Переключение между протоколами осуществляется аппаратно (переключателем на корпусе мыши) или программно (путем подачи специального сигнала со стороны драйвера мыши).

Таблица 2.12. Формат 3D Serial 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
4	0	0	R0	Z3	Z2	Z1	Z

Особенностью устройств данного типа является 8-битная длина слова данных во всех режимах работы (при этом в режиме MS Mouse шестой разряд в каждом слове данных просто дублируется в седьмой, как показано в таблице 2.13).Соответственно при использовании подобных устройств для последовательного порта необходимо установить следующие параметры работы: скорость приема-передачи данных 1200 бод, длина передаваемого слова - 8 бит, контроль по четности не используется, число стоповых бит равно 1. Однако даже если задана длина передаваемого слова 7 бит, в режиме MS Mouse седьмой разряд игнорируется, и нарушения процесса передачи не происходит. Таким образом, устройства данного типа могут успешно взаимодействовать с любым драйвером, поддерживающим стандартный протокол Microsoft.

Таблица 2.13. Нестандартный 8-битный формат типа MS Mouse для мыши с двумя альтернативными протоколами передачи данных

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

Предложенный фирмой IBM формат Mouse System (PC Mouse), который показан в таблице 2.14, в настоящее время почти не применяется, однако до сих пор поддерживается некоторыми универсальными устройствами.

Таблица 2.14. 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"

Обозначения в таблице расшифровываются следующим образом:

- X0-X7 - перемещение по оси X;
- Y0-Y7 - перемещение по оси Y;
- L - состояние левой кнопки (0 - нажата, 1 - отпущена);
- M - состояние средней кнопки (0 - нажата, 1 - отпущена);
- R - состояние правой кнопки (0 - нажата, 1 - отпущена).

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). Столь хитрый 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, то есть противоположно оси Y дисплея.

Для работы с устройством типа PC Mouse необходимо установить следующие параметры последовательного порта: скорость приема-передачи 1200 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.

2.1.4. Работа с манипуляторами мышь

Как это ни странно, ни BIOS, ни DOS не содержат непосредственных средств для работы с Serial Mouse. Для того, чтобы задействовать это устройство необходимо загружать драйвер мыши или специальную резидентную программу, выполняющую функции драйвера мыши. Универсальный драйвер мыши был предложен фирмой Microsoft. Он обеспечивает унифицированный интерфейс для работы с манипуляторами «мышь» или «трекбол» любого типа. Драйвер позволяет выполнять свыше 40 различных функций. Все фирмы-изготовители манипуляторов делают драйверы для своих устройств совместимыми с драйвером Microsoft, но каждая фирма вносит в них усовершенствования, создавая программистам множество ненужных проблем. Кроме того, стандартные функции мыши в MS-DOS рассчитаны только на текстовые и устаревшие 16-цветные графические режимы: попытка обращения к ним при использовании видеорежимов SVGA, Hi Color и True Color приводит в лучшем случае к «глюкам» на экране монитора, в худшем — к зависанию системы.

Для подключения драйвера мыши файл CONFIG.SYS должен содержать строку следующего вида:
devise=c:\mouse\mouse.sys

Если используется резидентная программа, она обычно вызывается в файле AUTEXEC.BAT:
c:\mouse\mouse.com

Драйвер мыши выполняет следующие функции:

- отслеживает перемещения курсора и нажатия на клавиши мыши;
- выводит на экран курсор, повторяющий движение мыши в графическом или текстовом режимах;
- предоставляет программам интерфейс для работы с мышью, основанный на вызове прерывания INT 33h.

В отличие от устройств других типов, аппаратный интерфейс мыши стандартизирован гораздо лучше, чем программный, и работать с мышью желательно через аппаратуру. Использовать прерывания DOS необходимо только для устройств, не имеющих простого общепринятого интерфейса, например, подключенных через инфракрасный порт или шину USB.

2.1.4.1. Функции DOS, предназначенные для работы с мышью

Драйвер мыши, независимо от того, реализован он через устанавливаемый драйвер или резидентную программу, устанавливает в операционной системе MS-DOS обработчик прерывания INT 33h. Этот обработчик выполняет все функции, связанные с обслуживанием мыши:

- сброс мыши и установку драйвера в исходное состояние;
- включение и выключение курсора мыши;
- установку курсора в определенное место экрана;
- определение текущих координат курсора и текущего состояния клавиш;
- определение координат курсора и состояния клавиш в момент нажатия на клавишу и в момент отпускания клавиши;
- определение области на экране, в пределах которой может перемещаться курсор;
- определение области на экране, в пределах которой курсор не будет виден;
- определение формы графического и текстового курсоров;
- определение величины перемещения мыши в сотых датах дюйма;
- подключение к драйверу процедуры, определенной в программе, получающей управление при нажатии на заданную клавишу или при перемещении мыши;
- запоминание и восстановление состояния драйвера;
- управление эмуляцией светового пера;
- управление скоростью движения курсора;
- указание пли определение используемой страницы видеопамяти;
- управление драйвером мыши.

Ниже приведено достаточно подробное описание функций прерывания INT 33h, используемых при

работе с мышью. Следует отметить, что для работы с мышью в графических режимах выше 16-цветного программист должен сам написать подпрограммы, обеспечивающие отображение курсора мыши и перемещение его по экрану в конкретном видеорежиме.

2.1.4.1.1. Прерывание 33h, функция 0000h:

Проверить наличие драйвера мыши и произвести сброс (Инициализация мыши)

Эта функция с кодом 00h выполняет аппаратный сброс мыши и программную установку драйвера мыши в начальное состояние. Заметим, что с помощью функции 00h можно выполнить установку драйвера в исходное состояние, не выполняя аппаратного сброса мыши.

Регистры на входе: AX = 0000h
Регистры на выходе: AX = состояние мыши;
BX = количество клавиш у мыши

Если на выходе из прерывания регистр AX содержит значение 0000h, значит, драйвер мыши не установлен. Если же содержимое этого регистра равно FFFFh, драйвер имеется и вы можете использовать другие функции для работы с мышью.

В регистре BX возвращается количество клавиш мыши:

Содержимое регистра	*Количество клавиш*
0	Больше или меньше чем две
2	Две клавиши
3	Мышь системы Mouse Systems, имеет 3 клавиши
FFFFh	Стандартная мышь Microsoft с двумя клавишами

При установке драйвера мыши в исходное состояние для программ, работающих в текстовом режиме, выполняются следующие действия:

- курсор перемещается в центр экрана и гасится;
- разрешается перемещение курсора по всей поверхности экрана, причем на экране отсутствуют зоны, в которых курсор является невидимым;
- устанавливается режим отображения курсора - инвертирование атрибута символа, на который указывает курсор;
- для изображения курсора выбирается нулевая страница видеопамяти;
- разрешается эмуляция светового пера (хотя это вам едва ли пригодится);
- устанавливается начальная скорость перемещения курсора.

Из всего перечисленного выше наиболее важным является то, что после сброса курсор на экран не отображается. При работе в текстовом режиме нужно вызвать функцию 000lh, чтобы курсор появился на экране. В современных графических режимах курсор средствами драйвера не отображается или отображается неправильно, поэтому с точки зрения драйвера он должен оставаться скрытым, то есть функцию 000lh вызывать нельзя.

2.1.4.1.2. Прерывание 33h, функция 0001h:

Включить курсор мыши

Функция 01h позволяет включать курсор мыши:

Регистры на AX = 000lh входе:
Регистры на выходе: Регистры не используются

Для управления видимостью курсора драйвер мыши использует внутренний счетчик. Этот счетчик можно увеличивать, вызывая функцию 01h прерывания INT 33h, или уменьшать при помощи функции 02h этого же прерывания.

После инициализации драйвера функцией 00h счетчик устанавливается равным -1. После первого вызова функции 01h счетчик становится равным нулю. При этом курсор мыши становится видимым, его можно перемещать по экрану.

Если счетчик равен нулю, то следующие вызовы функции 0lh игнорируются драйвером. Для того чтобы погасить курсор, используйте функцию 02h, которая при вызове уменьшает каждый раз содержимое счетчика на единицу.

Функция 01h сбрасывает область, в которой курсор не отображается (если такая область была ранее установлена функцией 10h).

Применять данную функцию можно только в текстовых режимах.

2.1.4.1.3. Прерывание 33h, функция 0002h:

Выключить курсор мыши

Функция 02h уменьшает на единицу счетчик видимости курсора. Если содержимое счетчика становится равным -1, изображение курсора пропадает с экрана.

Регистры на входе: AX = 0002h
Регистры на выходе: Регистры не используются

Если ваша программа применяет для вывода на экран метод прямой записи в видеопамять, перед обновлением содержимого экрана необходимо погасить курсор, а после завершения обновления - высветить его опять.

Это связано с тем, что драйвер мыши запоминает старое значение атрибута символа, на который указывал курсор до обновления содержимого видеопамяти. Вы изменили атрибут, записав новое значение непосредственно в экранную память. Теперь, если установить курсор мыши на другой символ, изображение старого символа будет испорчено - появится прямоугольник (как бы еще одно изображение курсора мыши).

Поскольку в современных графических режимах функция 0001h не работает, то не применяется и обратная ей функция 0002h.

2.1.4.1.3.1. Программа MSCURSOR

В листинге 2.1, приведем исходный текст программы MSCURSOR демонстрирующей применение описанных выше функций [9]. Программа инициализирует мышь, делает видимым курсор мыши и прячет курсор после того, как пользователь нажмет любую клавишу.

Листинг 2.1. Файл mscursor\mscursor.c

```
// ===================================
// Включение и выключение курсора мыши
// (C) Фролов А. В., 1997
//
// E-mail: frolov@glas.apc.org
// WWW:    http://www.glasnet.ru/~frolov
// или http://www.dials.ccas.ru/frolov
// ===================================
#include <dos.h>
#include <stdio.h>
#include <conio.h>

int main(void)
{
  int nButtons;
  union REGS rg;

  printf("MSCURSOR, (c) A. Frolov, 1997\n");

  // Инициализируем мышь
  rg.x.ax = 0;
  int86(0x33, &rg, &rg);

  if(rg.x.bx == 0)
  {
    printf("Mouse not found");
    return -1;
  }

  // Сохраняем количество клавиш
  nButtons = rg.x.bx;

  printf("Mouse type: ");

  switch (nButtons)

  {
    case 2:
    {
      printf("2-button mouse\n");
      break;
    }
    case 3:
    {
      printf("2-button Mouse Systems\n");
      break;
    }
    case 0:
    {
      printf("Unknown type %d\n", nButtons);
      break;
    default:
    {
      printf("Unknown type %d\n", nButtons);
      break;
    }
  }

  // Включаем курсор и ожидаем, пока пользователь
  // нажмет на клавишу
  rg.x.ax = 1;
  int86(0x33, &rg, &rg);
  printf("Mouse cursor on. Press any key\n");
  getch();

  // Выключаем курсор
  rg.x.ax = 2;
  int86(0x33, &rg, &rg);

  printf("Mouse cursor off. Press any key\n");
  getch();
  return 0;
```

2.1.4.1.4. Прерывание 33h, функция 0003h:

Получить информацию о положении курсора и состоянии кнопок мыши

Функция 03h возвращает текущие координаты курсора мыши и состояние клавиш.

Регистры на входе: AX = 0003h
Регистры на выходе: BX = состояние клавиш мыши;
CX = координата X курсора;
DX = координата Y курсора

В зависимости от того, была ли нажата какая-либо клавиша мыши в момент вызова функции, в регистре BX могут быть установлены следующие флаги:

Установленный бит регистра BX	Клавиша, которая была нажата
0	Левая
1	Правая
2	Средняя

Для графических режимов координаты располагаются в различных диапазонах, в зависимости от текущего режима видеоадаптера:

Размер экрана в пикселах	Номер режима	Диапазон координат по оси X	Диапазон координат по оси Y
320×200	4, 5	0 - 638	0 - 199
640×200	6	0 - 639	0 - 199
320×200	0Dh	0 - 638	0 - 199
640×200	0Eh	0 - 639	0 - 199
640×350	0Fh	0 - 639	0 - 349

Программы, работающие в текстовом режиме, должны разделить полученные координаты на 8 (как координату X, так и координату Y).

Серьезный недостаток данной функции заключается в том, что невозможно определить, была ли нажата (отпущена) кнопка мыши в текущей позиции курсора, или это произошло ранее в другом месте экрана. Поэтому функция 0003h применяется только в графических видеорежимах — в подпрограмме, отвечающей за перемещение курсора по экрану. Для получения более точной информации о нажатиях и отпусканиях кнопок мыши используются функции 0005h и 0006h.

2.1.4.1.4.1. Программа MSGCURS

В листинге 2.2, приведем исходный текст программы MSGCURS, которая запрашивает номер режима видеоадаптера, устанавливает его и динамически отображает координаты курсора, а также состояние клавиш мыши [9]. После завершения работы программа восстанавливает первоначальный режим видеоадаптера.

Листинг 2.2. Файл msgcurs\msgcurs.c

```
// ==================================
// Включение и выключение курсора мыши
// в разных видеорежимах
// (C) Фролов А. В., 1997
//
// E-mail: frolov@glas.apc.org
// WWW:    http://www.glasnet.ru/~frolov
// или http://www.dials.ccas.ru/frolov
// ==================================
#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
union REGS rg;
int main(void)
{
  int i;
  unsigned old_videomode, new_videomode;
  char buf[20], *bufptr;
  int nButtons;
  // Определяем текущий видеорежим
  rg.x.ax = 0x0f00;
  int86(0x10, &rg, &rg);
  old_videomode = rg.h.al;
  // Устанавливаем новый видеорежим
  buf[0] = 10;
  printf("Enter new video mode: ");
  bufptr = cgets(buf);
  // Преобразуем введенное число к формату int
  new_videomode = atoi(bufptr);
  rg.h.ah = 0;
  rg.h.al = new_videomode;
  int86(0x10, &rg, &rg);
  // Инициализируем мышь
  rg.x.ax = 0;
  int86(0x33, &rg, &rg);
  if(rg.x.bx == 0)
  {
    printf("Mouse not found\n");
    return -1;
  }
  // Сохраняем количество клавиш
  nButtons = rg.x.bx;
  printf("Mouse type: %d\n", nButtons);
  // Включаем курсор
  rg.x.ax = 1;
  int86(0x33, &rg, &rg);
  printf("Mouse cursor on. Press any key\n");
  getch();
  while(!kbhit())
  {
    rg.x.ax = 3;
    int86(0x33, &rg, &rg);
    printf("%2d x:%5d y:%5d",
           rg.x.bx, rg.x.cx, rg.x.dx);
         for(i=0; i<18; i++) printf("\b");
  }
  getch();
  // Выключаем курсор
  rg.x.ax = 2;
  int86(0x33, &rg, &rg);
  // Восстанавливаем режим видеоадаптера
  rg.h.ah = 0;
  rg.h.al = old_videomode;
  int86(0x10, &rg, &rg);
  return 0;
}
```

Заметим, что использование функции 03h не самый лучший способ работы с мышью. Программа должна постоянно следить за координатами курсора или за состоянием клавиш. Это может привести к непроизводительным затратам процессорного времени на опрос состояния.

Позже будут рассмотрены другие способы определения состояния мыши.

2.1.4.1.5. Прерывание 33h, функция 0004h:

Установить новое положение курсора

Обычно курсор мыши устанавливает не программа, а пользователь. Однако с помощью функции 04h программа тоже может установить курсор в заданную позицию.

Регистры на входе: AX = 0004h;
CX = новая координата X курсора;
DX = новая координата Y курсора
Регистры на выходе: Регистры не используются

Положение координаты задается в пикселах относительно левого верхнего угла экрана, причем предполагается, что ось Y направлена сверху вниз. Для текстового режима устанавливаемые номера строки и столбца должны быть умножены на 8.

В том случае, когда программа пытается установить курсор в область, где курсор невидим (эта область задается функцией 10h), она сможет это сделать. Курсор при этом исчезнет с экрана, что не всегда желательно.

Если область для перемещения курсора была ограничена при помощи функций 07h или 08h, то курсор будет вести себя следующим образом. При попытке установить курсор за границу области ограничения он будет установлен в точку, которая находится внутри границы на минимальном расстоянии от точки, заданной при вызове функции.

2.1.4.1.6. Прерывание 33h, функция 0005h:

Определить положение курсора при нажатии клавиши.

Регистры на входе: AX = 0005h;
BX= клавиша, при нажатии которой запоминается состояние мыши:
0 - левая, 1 - правая, 2 - средняя
Регистры на выходе: AX = состояние клавиш мыши:
BX = количество нажатий на заданную клавишу. Это значение обнуляется после вызова функции;
CX = координата курсора X;
DX = координата курсора Y

В зависимости от того, была ли нажата какая-либо клавиша мыши в момент вызова функции, в регистре AX могут быть установлены следующие флаги:

Установленный бит регистра AX	Клавиша, которая была нажата
0	Левая
1	Правая
2	Средняя

В отличие от функции 03h эта функция возвращает программе не текущее состояние мыши, а запомненное в момент последнего нажатия на клавишу, заранее определенную при вызове функции. Она также возвращает количество нажатий на заданную клавишу, которое вы можете использовать для обнаружения двойных щелчков.

2.1.4.1.7. Прерывание 33h, функция 0006h:

Определить положение курсора при отпускании клавиши

Функция 06h возвращает программе состояние мыши, запомненное в момент отпускания клавиши, которая была заранее определена при вызове функции. Она также возвращает количество отпусканий заданной клавиши.

Регистры на входе: AX = 0006h;
BX = клавиша, при отпускании которой запоминается состояние мыши:
0 - левая; 1 - правая; 2 - средняя
Регистры на выходе: AX = состояние клавиш мыши;
BX = количество нажатий на заданную клавишу. Это значение обнуляется после вызова функции;
CX = координата курсора X;
DX = координата курсора Y

Формат регистра AX, определяющий состояние клавиш мыши, такой же, как и для функции 05h.

2.1.4.1.8. Прерывание 33h, функция 0007h:

Задать диапазон движения курсора по горизонтали

Функция 07h позволяет ограничить диапазон перемещений курсора мыши по горизонтали.

Регистры на входе: AX = 0007h;
CX = минимальная координата X;
DX = максимальная координата X
Регистры на выходе: Регистры не используются

Функция 0007h используется совместно с функцией для задания на экране рабочего окна, за пределы которого не должен выходить курсор мыши. В простейшем случае функция применяется, чтобы предотвратить уход части изображения графического курсора за границу экрана.

2.1.4.1.9. Прерывание 33h, функция 0008h:

Задать диапазон движения курсора по вертикали

Функция 08h позволяет ограничить диапазон перемещений курсора мыши по вертикали.

Регистры на входе: AX = 0008h;
CX = минимальная координата Y;
DX = максимальная координата Y

Регистры на выходе: Регистры не используются

Как и функция 0007h, данная функция чаще всего применяется, чтобы предотвратить уход части изображения курсора за границу экрана.

2.1.4.1.10. Прерывание 33h, функция 0009h:

Задать форму курсора в графическом режиме

С помощью функции 09h программа, работающая в графическом режиме, может изменить форму курсора мыши и положение точки внутри изображения курсора, координаты которой используются в качестве координат курсора остальными функциями.

Регистры на входе: AX = 0009h;
BX = номер позиции точки-указателя графического курсора (от -16 до 16);
CX = номер строки точки-указателя (от -16 до 16);
ES:DX = указатель на растровое изображение курсора

Регистры на выходе: Регистры не используются

Регистры ES:DX указывают на область памяти размером 64 байта. Эта область состоит из двух массивов длиной по 32 байта. Первый массив представляет собой логическую маску размером 16×16 бит, которая накладывается на участок видеопамяти с использованием логической операции "И". Второй массив - тоже маска размером 16×16 бит, но она накладывается с использованием логической операции "Исключающее ИЛИ", инвертируя отдельные точки изображения.

Номера позиции и строки точки-указателя, устанавливаемые по умолчанию, равны нулю (BX = CX = 0). Это соответствует левой верхней точке в изображении курсора. Значения BX = CX =15 соответствуют нижней правой точке.

2.1.4.1.10.1. Программа MSGFORM

Программа MSGFORM, изменяет форму курсора в графическом режиме (листинг 2.3) [9].

Листинг 2.3. Файл msgform\msgfonn.c

```
// ========================================
// Изменение формы курсора в графическом режиме
// (C) Фролов А. В., 1997
//
// E-mail: frolov@glas.apc.org
// WWW:    http://www.glasnet.ru/~frolov
// или http://www.dials.ccas.ru/frolov
// ========================================
#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

union REGS reg;
void ms_gform(int xt, int yt, char _far *form);
unsigned char form[64] =
{
  // Массив маски по "И"
  255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
  255, 255, 255, 255, 255, 255,
  255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
  255, 255, 255, 255, 255, 255,
  // Массив маски по "Исключающее ИЛИ"
  127, 254, 127, 254, 127, 254, 127, 254, 127, 254,
  127, 254, 127, 254, 0,   0,
  0,   0,   127, 254, 127, 254, 127, 254, 127, 254, 127, 254,
  127, 254, 127, 254
};

int main(void)
{
  unsigned old_videomode, new_videomode;
  char buf[20], *bufptr;

// Определяем текущий видеорежим
reg.x.ax = 0x0f00;
int86(0x10, &reg, &reg);
old_videomode = reg.h.al;

// Устанавливаем новый видеорежим
buf[0] = 10;
printf("Enter new video mode: ");
bufptr = cgets(buf);
new_videomode = atoi(bufptr);

reg.h.ah = 0;
reg.h.al = new_videomode;
int86(0x10, &reg, &reg);

// Инициализируем мышь
reg.x.ax = 0;
int86(0x33, &reg, &reg);

if(reg.x.bx == 0)
{
  printf("Mouse not found\n");
  return -1;
}
// Задаем новую форму для курсора мыши
ms_gform(0,0, &form[0]);

// Включаем курсор
 reg.x.ax = 1;
 int86(0x33, &reg, &reg);
 getch();

 reg.h.ah = 0;
 reg.h.al = old_videomode;
```

```
  int86(0x10, &reg, &reg);
  return 0;
}
void ms_gform(int xt, int yt, char _far *form)
{
  struct SREGS segregs;
  reg.x.ax = 9;
  reg.x.bx = xt;
  reg.x.cx = yt;
  reg.x.dx = FP_OFF(form);
  segregs.es = FP_SEG(form);
  int86x(0x33,&reg,&reg,&segregs);
}
```

2.1.4.1.11. Прерывание 33h, функция 000Ah:

Задать форму курсора в текстовом режиме

С помощью функции 0Ah программа может изменять форму курсора мыши в текстовом режиме.

Регистры на входе: AX = 000Ah;
BX = тип курсора:
0 - определяемый программно, 1 - определяемый аппаратно;
CX = маска экрана (для BX = 0) или начальная строка курсора (для BX = 1);
DX = маска курсора (для BX = 0) или конечная строка курсора (для BX = 1)

Регистры на выходе: Регистры не используются

В зависимости от содержимого регистра BX драйвер мыши использует курсор, определяемый аппаратными средствами, либо курсор, определяемый программно. По умолчанию используется "программный курсор", который отображается в виде символа с инвертированным значением атрибута. Курсор, сформированный аппаратными средствами, выглядит аналогично обычному текстовому курсору, его форма - прямоугольник. Размер этого прямоугольника можно задавать при помощи регистров CX и DX.

Для курсора, определяемого программно, вначале выполняется операция логического "И" над содержимым видеопамяти в том месте, куда указывает курсор, и маской экрана. Затем выполняется операция "Исключающее ИЛИ" с маской курсора.

Младший байт масок соответствует коду ASCII символа, старший - это байт атрибута символа.

Значения, используемые по умолчанию, - BX = 7700h, CX = FFFFh.

Если вам надо изменить цвет курсора, не меняя его форму, задайте CX = 00FFh, BX = xx00h, где xx определяет цвет.

2.1.4.1.11.1. Программа MSTFORM

Приведем исходный текст программы MSTFORM (листинг 2.4), создающую курсор в виде вертикальной стрелки, направленной вверх, на синем фоне [9].

Листинг 2.4. Файл mstform\mstfonn.c

```
// =======================================
// Изменение формы курсора в текстовом режиме
// (C) Фролов А. В., 1997
//
// E-mail: frolov@glas.apc.org
// WWW:    http://www.glasnet.ru/~frolov
// или http://www.dials.ccas.ru/frolov
// =======================================
#include <dos.h>
#include <stdio.h>
#include <conio.h>
union REGS reg;
void ms_tform(int type, int mask1, int mask2);
int main(void)
{
  // Инициализируем мышь
  reg.x.ax = 0;
  int86(0x33, &reg, &reg);
  if(reg.x.bx == 0)
  {
    printf("Mouse not found\n");
    return -1;
  }
  // Задаем новую форму для курсора мыши
  ms_tform(0, 0, 0x1418);
  // Включаем курсор
  reg.x.ax = 1;
  int86(0x33, &reg, &reg);
  getch();
  return 0;
}
void ms_tform(int type, int mask1, int mask2)
{
  reg.x.ax = 0xA;
  reg.x.bx = type;
  reg.x.cx = mask1;
  reg.x.dx = mask2;
  int86(0x33,&reg,&reg);
}
```

2.1.4.1.12. Прерывание 33h, функция 000Bh:

Определить содержимое счетчиков перемещения

Функция 0Bh позволяет определить относительное перемещение мыши с момента последнего вызова этой функции. Результат возвращается в указанных выше регистрах. Для измерения перемещения используется единица mickey -"мики". Один мик соответствует 0,005 дюйма (то есть 1/200 дюйма).

Регистры на входе: AX = 000Bh
Регистры на выходе: CX = перемещение по горизонтали с момента последнего вызова функции;
DX = перемещение по вертикали с момента последнего вызова функции

Отрицательные значения перемещения означают движение влево и вверх, положительные - вправо и вниз.

Для преобразования миков в пикселы, соответствующие точкам экрана, можно использовать функцию 1Bh, которая будет описана позже.

2.1.4.1.13. Прерывание 33h, функция 000Ch:

Установить драйвер событий

Функция 0Ch позволяет программе создать свой собственный драйвер (обработчик) событий, связанных с перемещением мыши, а также с нажатием или отпусканием клавиш мыши.

Регистры на входе: AX = 000Ch;
CX = маска вызова;
ES:DX = адрес подключаемого драйвера событий
Регистры на выходе: Регистры не используются

Биты маски вызова, передаваемой функции в регистре CX, определяют условие вызова драйвера событий:

Бит маски вызова	*Когда выполняется* вызов
0	Перемещение мыши
1	Нажатие левой клавиши
2	Отпускание левой клавиши
3	Нажатие правой клавиши
4	Отпускание правой клавиши
5	Нажатие средней клавиши
6	Отпускание средней клавиши
7 - 15	Зарезервированы

Если записать в регистр CX значение 7Fh, драйвер будет вызываться при возникновении любого события, а если 00h - драйвер будет отключен.

Адрес процедуры драйвера передается при вызове функции в регистровой паре ES:DX. Драйвер должен быть оформлен в виде процедуры типа FAR, завершающейся командой дальнего возврата RETF.

Когда драйвер получает управление, в регистрах процессора содержатся следующие значения:

Реги	*Описание*
AX	Маска вызова, такая же, как и при вызове функции 0Ch
BX	Состояние клавиш мыши: бит 0 - левая клавиша; бит 1 - правая клавиша; бит 2 - средняя клавиша
CX	Координата X курсора мыши
DX	Координата Y курсора мыши
SI	Относительное перемещение мыши по горизонтали в миках
DI	Относительное перемещение мыши по вертикали в миках
DS	Сегмент данных драйвера мыши

Так как регистр DS при вызове драйвера событий содержит сегмент данных драйвера мыши, ваш драйвер событий должен позаботиться о правильной установке этого регистра. Однако драйвер событий не обязан сохранять и восстанавливать содержимое регистра DS и других регистров процессора.

Отметим, что, если вам необходимо отключить драйвер, выполните повторный вызов функции 0Ch, записав в регистр CX нулевое значение. Если ваша программа, устанавливающая собственный драйвер событий, завершает свою работу и передает управление MS-DOS, предварительно она обязательно должна отключить драйвер событий.

Составление программы драйвера событий имеет некоторые особенности. Драйвер событий вызывается не из программы пользователя, а из драйвера мыши. При этом сегментный регистр DS будет указывать на сегмент данных драйвера мыши, а не на сегмент данных вашей программы.

Драйвер событий может также организовать очередь событий, записывая в эту очередь состояние мыши на момент появления события и время появления события. Прикладная программа будет затем извлекать

события из очереди и анализировать их.

2.1.4.1.13.1. Программа MSDRIVER

Программа MSDRTVER иллюстрирует способ работы с драйвером событий. Исходный текст драйвера событий, составленный на языке ассемблера, представлен в листинге 2.5 [9].

Листинг 2.5. Файл msdriver\handler.asm

```
;**
;.Name       ms_handl
;.Title      Драйвер событий
;
;.Descr      Драйвер событий вызывается драйвером мыши,
;           когда происходит какое-нибудь событие из числа
;           заданных при установке драйвера событий.
;           Функция не должна вызываться из программы
;           пользователя, ее вызывает только драйвер мыши.


;
;.Proto      void far ms_handl(void);
;
;.Params     Не используются
;**
        DOSSEG
DGROUP  GROUP   _DATA
_DATA   SEGMENT WORD PUBLIC 'DATA'
_DATA   ENDS
_TEXT   SEGMENT WORD PUBLIC 'CODE'
        ASSUME  cs:_TEXT, ds:DGROUP, ss:DGROUP
; Флаг вызова драйвера событий
extrn   _ms_flag:word
; Внешние переменные для записи содержимого регистров
extrn   _ms_bx:word
extrn   _ms_cx:word
extrn   _ms_dx:word
extrn   _ms_si:word
extrn   _ms_di:word
extrn   _ms_ds:word
public  _ms_handl
_ms_handl   proc far
        mov     _ms_ds, ds
; Так как на входе в драйвер событий регистр
; DS указывает на сегмент данных драйвера мыши,
; устанавливаем его на сегмент данных программы;
        push    ax
        mov     ax, DGROUP
        mov     ds, ax
        pop     ax
        mov     _ms_bx, bx
        mov     _ms_cx, cx
        mov     _ms_dx, dx
        mov     _ms_si, si
        mov     _ms_di, di
; Устанавливаем флаг вызова драйвера в 1, сигнализируя
; программе о том, что произошло событие.
        mov     _ms_flag, 1
        ret
_ms_handl   endp
_TEXT   ENDS
        END
```

При вызове этот драйвер устанавливает глобальную переменную _ms_flag в единицу, затем переписывает содержимое всех нужных регистров в соответствующие глобальные переменные.

Программа, установив драйвер событий и сбросив флаг _ms_flag, может выполнять какие-либо действия (например, вывод на экран движущегося изображения), постоянно проверяя флаг _ms_flag. Как только произойдет какое-либо событие (нажатие или отпускание клавиши мыши, перемещение мыши), драйвер событий установит флаг в единицу. Программа при этом может узнать состояние мыши, прочитав содержимое глобальных переменных _ms_bx, _ms_dx и так далее.

Исходный текст программы MSDRIVER представлен в листинге 2.6 [9]

Листинг 2.6. Файл msdriver\msdriver.c

```
// ================================
// Работа с драйвером событий
// (C) Фролов А. В., 1997
//
// E-mail: frolov@glas.apc.org
// WWW:    http://www.glasnet.ru/~frolov
// или http://www.dials.ccas.ru/frolov
// ================================
#include <dos.h>
#include <stdio.h>
#include <conio.h>

union REGS reg;
struct SREGS segregs;

extern void far MS_HANDL(void);
void ms_seth(int mask, void (far *hand)());
// Флаг драйвера событий. При вызове драйвер
// событий запишет в эту переменную значение 1
unsigned MS_FLAG;

// Область для содержимого регистров на входе
// в драйвер событий.
unsigned MS_BX;
unsigned MS_CX;
unsigned MS_DX;
unsigned MS_SI;
unsigned MS_DI;
unsigned MS_DS;
int main (void)
{
  MS_FLAG=0;

  // Инициализируем мышь
  reg.x.ax = 0;
  int86(0x33, &reg, &reg);

  if(reg.x.bx == 0)
  {
    printf("Mouse not found\n");
    return -1;
```

```
}
// Подключаем драйвер событий,
// устанавливаем маску таким образом, чтобы
// драйвер вызывался при нажатии на левую
// или правую клавишу мыши

ms_seth(2 | 8, MS_HANDL);
// Включаем курсор
reg.x.ax = 1;
int86(0x33, &reg, &reg);
// Ожидаем вызова драйвера событий.
for(;;)
{
  if(MS_FLAG)
  {
    printf("\nRegisters on driver entry:"
      "\nms_bx: %0X"
      "\nms_cx: %0X"
      "\nms_dx: %0X"
      "\nms_si: %0X"
      "\nms_di: %0X"
      "\nms_ds: %0X",
      MS_BX,MS_CX,MS_DX,MS_SI,MS_DI,MS_DS);
      printf("\nPress any key...");
      getch();
      return 0;
    }
  }
}
void ms_seth(int mask, void (far *hand)())
{
  reg.x.ax = 0x14;
  reg.x.cx = mask;
  reg.x.dx = FP_OFF(hand);
  segregs.es = FP_SEG(hand);
  int86x(0x33,&reg,&reg,&segregs);
}
```

2.1.4.1.14. Прерывание 33h, функция 000Dh:

Включить эмуляцию светового пера

Световое перо теперь можно увидеть разве лишь в музее. Однако вы можете заменить световое перо на мышь, если ваша программа использует световое перо (например, она написана на языке Бейсик и вызывает функцию PEN). Для включения эмуляции светового пера вы можете воспользоваться функцией 0Dh:

Регистры на входе: AX = 000Dh
Регистры на выходе: Регистры не используются

После включения режима эмуляции драйвер запоминает координаты курсора в момент, когда пользователь нажимает на клавиши мыши. Эти координаты могут быть впоследствии считаны функцией PEN или функцией 04h прерывания INT 10h, предназначенной для работы со световым пером.

2.1.4.1.15. Прерывание 33h, функция 000Eh:

Выключить эмуляцию светового пера

Функция 0Eh выключает режим эмуляции светового пера.

Регистры на входе: AX = 000Eh
Регистры на выходе: Регистры не используются

2.1.4.1.16. Прерывание 33h, функция 000Fh:

Задать скорость перемещения курсора мыши

Функция 0Fh определяет чувствительность мыши к перемещению по поверхности стола, то есть устанавливает соответствие между величиной перемещения мыши по столу и величиной перемещения курсора мыши по экрану.

Регистры на входе: AX = 000Fh;
CX = количество миков на 8 точек по горизонтали;
DX = количество миков на 8 точек по вертикали
Регистры на выходе: Регистры не используются

При инициализации драйвера мыши используются следующие значения: CX = 8, DX = 16.

2.1.4.1.17. Прерывание 33h, функция 00010h:

Установить область исключения для курсора

Функция позволяет задать на экране прямоугольную область, в которой автоматически выключается изображение курсора мыши - область исключения. Эта область отменяется функциями 01h (включить курсор мыши) и 00h (инициализация).

Регистры на входе: AX = 00l0h;
CX, DX = координаты (X, Y) верхнего левого угла области исключения;
SI, DI = координаты (X, Y) нижнего правого угла области исключения
Регистры на Регистры не используются выходе:

Оператор может поместить курсор мыши в область исключения, при этом изображение курсора пропадет.

Основное назначение этой функции - предоставить программе возможность изменять содержимое области экрана, не выключая изображение курсора. Недостаток наличия области исключения – пользователь может потерять курсор мыши, если он случайно окажется в такой области.

2.1.4.1.18. Прерывание 33h, функция 00012h:

Задать увеличенный графический курсор

Функция 12h позволяет задать увеличенный по размеру курсор мыши, но она определена только для мыши системы PC MOUSE.

Регистры на входе: AX = 0012h;
BH = ширина курсора в словах;
CH = количество строк в изображении курсора;
BL = номер позиции точки-указателя графического курсора (от -16 до 16);
CL = номер строки точки-указателя (от -16 до 16);
ES:DX = указатель на растровое изображение курсора

Регистры на выходе: Регистры не используются

2.1.4.1.19. Прерывание 33h, функция 00013h:

Определить порог удвоения скорости

Если вы перемещаете мышь со скоростью, превышающей порог удвоения, заданный функцией 13h, аппаратура мыши удваивает величину перемещения. Таким образом, используя медленное перемещение мыши, вы можете точно устанавливать курсор на требуемый элемент изображения. Если вам необходимо переместить курсор на значительное расстояние по экрану, вы можете увеличить скорость перемещения мыши.

Формат вызова функции 13h приведен ниже.

Регистры на входе: AX = 0013h

Регистры на выходе: DX = значение порога удвоения, мики в секунду

При инициализации устанавливается значение порога, равное 64 микам в секунду (1/3 дюйма в секунду). Если вам надо установить это значение, вы можете при вызове функции 13h задать DX = 0.

2.1.4.1.20. Прерывание 33h, функция 00014h:

Заменить драйвер событий

Функция 14h аналогична функции 0Ch, однако ее основное назначение - временная замена драйвера событий. Например, подпрограмма в начале своей работы может установить свой драйвер событий, а перед завершением - активизировать драйвер, использовавшийся ранее.

Регистры на входе: AX = 0014h;
CX = маска вызова;
ES:DX = адрес подключаемого драйвера событий

Регистры на выходе: CX = маска предыдущего драйвера событий;
ES:DX = адрес предыдущего (заменяемого) драйвера событий

2.1.4.1.21. Прерывание 33h, функция 00015h:

Определить размер буфера состояния драйвера

Если вам требуется временно сохранить состояние драйвера мыши, а затем восстановить его, вы можете воспользоваться специально предназначенными для этого функциями 16h и 17h. Для этих функций требуется буфер, в котором будет храниться состояние драйвера. Размер буфера можно определить с помощью функции 15h.

Регистры на входе: AX = 0015h

Регистры на выходе: BX = размер буфера, требующийся для хранения состояния драйвера мыши

Когда может потребоваться запоминание и восстановление состояния драйвера?

Например, при использовании мыши резидентными (TSR) программами желательно сохранить состояние драйвера перед началом работы TSR-программы и восстановить его перед завершением работы TSR-программы.

2.1.4.1.22. Прерывание 33h, функция 00016h:

Сохранить состояние драйвера

Функция 16h позволяет записать состояние драйвера в буфер, размер которого должен быть определен с помощью функции 15h.

Регистры на входе: AX = 0016h
ES:DX = адрес буфера для записи состояния драйвера

Регистры на выходе: Регистры не используются

2.1.4.1.23. Прерывание 33h, функция 00017h:

Восстановить состояние драйвера

Функция 17h позволяет восстановить состояние драйвера из буфера, в который оно было записано при помощи функции 16h.

Регистры на входе: AX = 0017h
ES:DX = адрес буфера, содержащего состояние драйвера

Регистры на выходе: Регистры не используются

2.1.4.1.24. Прерывание 33h, функция 00018h:

Установить альтернативный драйвер событий

Функция 18h предназначена для установки альтернативного драйвера событий.

Регистры на входе: AX = 0018h;
CX = маска вызова;
ES:DX = адрес подключаемого драйвера событий
Регистры на выходе: AX = результат установки:
0018h - драйвер успешно установлен;
FFFFh - ошибка при установке драйвера

По сравнению с функцией 0Ch эта функция обеспечивает дополнительные возможности:

- проверку состояния клавиш <Shift>, <Control>, <Alt> во время нажатия на клавиши мыши;
- возможность одновременной установки до трех драйверов событий, каждый из которых использует свою маску событий, задаваемых в регистре CX.

При попытке установить два драйвера с одной и той же маской событий функция возвращает в регистре AX код ошибки FFFFh. В этом случае вы можете использовать функцию 19h для получения адреса предыдущего установленного драйвера событий, отключить его и повторить попытку подключения своего драйвера.

Функция 18h может применяться для отключения драйвера событий, если в регистрах ES:DX указать его адрес и при этом задать в регистре CX значение маски, равное нулю.

2.1.4.1.25. Прерывание 33h, функция 00019h:

Получить адрес альтернативного драйвера событий

Функция 19h предназначена для получения адреса драйвера событий с заданной маской событий. Получив адрес, вы можете установить новый драйвер, использующий эту же маску.

Регистры на входе: AX = 0019h;
CX = маска событий, для которой требуется получить адрес драйвера
Регистры на выходе: CX = маска событий или 0000h, если заданной маске не соответствует ни один установленный драйвер событий;
ES:DX = адрес драйвера событий, использующий заданную маску событий

2.1.4.1.26. Прерывание 33h, функция 0001Ah:

Установить чувствительность мыши

Функция 1Ah является комбинацией функций 0Fh и 13h. Она позволяет одновременно устанавливать чувствительность мыши и порог удвоения скорости.

Регистры на входе AX = 00lAh;
BX = горизонтальная чувствительность в миках на пиксел;
CX = вертикальная чувствительность в миках на пиксел;
DX = значение порога удвоения, мики в секунду
Регистры на выходе: Регистры не используются

2.1.4.1.27. Прерывание 33h, функция 0001Bh:

Определить чувствительность мыши

Функция 1Bh позволяет определить текущие значения для чувствительности мыщи и порога удвоения.

Регистры на входе: AX=001Bh
Регистры на выходе: BX = горизонтальная чувствительность в миках на пиксел;
CX = вертикальная чувствительность в миках на пиксел;
DX = значение порога удвоения, мики в секунду

2.1.4.1.28. Прерывание 33h, функция 0001Ch:

Установить частоту прерываний для Inport Mouse

Мышь периодически вырабатывает сигнал прерывания, по которому драйвер считывает текущее состояние мыши. С помощью функции 1Ch вы можете изменять частоту появления прерываний, но только для мыши системы Inport Mouse (тип мыши нетрудно определить с помощью функции 24h).

Регистры на входе: AX = 001Ch;
BX = код скорости прерываний:
1 - нет прерываний;
2 - 30 прерываний в секунду;
4 - 50 прерываний в секунду;
8 - 100 прерываний в секунду;
16 - 200 прерываний в секунду
Регистры на Регистры не используются выходе:

Чем больше частота прерываний, тем больше точность определения состояния мыши.

2.1.4.1.29. Прерывание 33h, функция 0001Dh:

Установить номер страницы видеопамяти

Функция 1Dh задает номер страницы видеопамяти, на которой будет отображаться курсор мыши. По умолчанию для отображения используется страница 0.

Регистры на входе: AX=001Dh;
BX = номер страницы видеопамяти
Регистры на выходе: Регистры не используются

2.1.4.1.30. Прерывание 33h, функция 0001Eh:

Определить номер страницы видеопамяти

Функция 1Eh возвращает номер страницы видеопамяти, на которой в настоящее время отображается курсор мыши.

Регистры на входе: AX = 001Eh
Регистры на выходе: BX = номер страницы видеопамяти

2.1.4.1.31. Прерывание 33h, функция 0001Fh:

Отключить драйвер мыши

После вызова функции 1Fh драйвер мыши полностью отключается. Вектор, прерывания INT 33h остается определенным, однако теперь выполняется только одна функция прерывания INT 33h - функция 21h (программный сброс мыши).

Регистры на входе: AX = 00lFh
Регистры на выходе: AX = результат выполнения:
001Fh-драйвер отключен;
FFFFh - отключение невозможно;
ES:DX = адрес предыдущего драйвера мыши

Функцию 1Fh удобно использовать для временной замены драйвера на собственную систему обслуживания мыши. Сначала вы отключаете драйвер функцией 1Fh, запоминая адрес предыдущего драйвера, возвращаемого в регистрах ES:DX. Затем устанавливаете собственную систему обслуживания мыши, а после ее использования - восстанавливаете значение этого адреса.

2.1.4.1.32. Прерывание 33h, функция 00020h:

Восстановить драйвер мыши

Функция 20h восстанавливает связь между мышью и драйвером, отключенную вызовом функции 1Fh.

Регистры на входе: AX = 0020h
Регистры на выходе: Регистры не используются

2.1.4.1.33. Прерывание 33h, функция 00021h:

Сбросить драйвер мыши

Функция 21h аналогична функции 00h, но она не выполняет аппаратный сброс оборудования мыши.

Регистры на входе: AX = 0021h
Регистры на выходе: AX результат:
0021h - драйвер сброшен успешно;
FFFFh - невозможно сбросить драйвер (например, из-за того, что драйвер не установлен);
BX = количество клавиш на корпусе мыши

2.1.4.1.34. Прерывание 33h, функция 00024h:

Определить тип мыши

Функция 24h дает информацию о типе используемой мыши, версии драйвера мыши и об используемом номере прерывания.

Регистры на входе: AX = 0024h
Регистры на выходе: BH = верхний (major) номер версии драйвера;
BL = нижний (minor) номер версии драйвера;
CH = тип мыши:
1 - Bus Mouse;
2 - Serial Mouse;,
3 - Inport Mouse;
4 - PS/2 Mouse;
5 - HP Mouse;
CL = номер используемого прерывания (IRQ):
0 - IBM PS/2;
2, 3, 4, 5, 7- IBM PC

2.1.4.2. Непосредственная работа с мышью типа MS Mouse

Рассмотрим непосредственную работу с мышью на примере манипуляторов, использующих стандартный формат MS Mouse. Процесс установки нового драйвера мыши включает целый ряд операций, которые необходимо выполнять в следующем порядке.

1. Запретить прерывания от последовательного порта.
2. Загрузить в регистры COM-порта параметры, соответствующие протоколу передачи данных.
3. Установить вектор прерывания на новую программу-обработчик.
4. Разрешить аппаратуре контроллера прерываний формировать прерывания от используемого мышью COM-порта.
5. Подать на мышь напряжение питания.
6. Разрешить генерацию прерываний COM-портом.

Разрешить или запретить прохождение прерываний от последовательного порта можно при помощи регистра маски ведущей микросхемы контроллера прерываний, к которой присоединены линии запросов от COM1 (линия IRQ4) и COM2 (линия IRQ3). Работа с контроллером прерываний рассмотрена в Приложении 1 в разделе «Контроллер прерываний». После завершения обработки прерывания необходимо разблокировать контроллер, послав ему команду завершения обработки прерывания EOI.

Настройка вектора прерывания заключается в записи начального адреса программы-обработчика в соответствующие вектору ячейки оперативной памяти. Для записи вектора необходимо занести в сегментный регистр данных нулевое значение, в индексный регистр поместить номер вектора прерывания, помноженный на 4, записать в ячейку памяти (используя косвенную адресацию) слово — смещение, увеличить значение индексного регистра на 2 и записать слово — сегмент адреса обработчика прерывания.

Рассмотрим более подробно последние два этапа, поскольку они имеют ряд неочевидных особенностей. В процессе разработки способа подключения мыши к последовательным портам был использован радиолюбительский трюк: питание мыши осуществляется от сигнальных линий DTR и RTS интерфейса. Для управления генераций прерываний применяется линия OUT2 контроллера COM-порта. Таким образом, чтобы подать питание на мышь и разрешить порту вырабатывать прерывания, необходимо установить в 1 биты 0, 1 и 3 регистра управления модемом MCR, то есть записать по адресу xFCh значение 0Bh.

Чаще всего мышь подключают к порту COM1. Однако, если заранее не известно, к какому порту подключена мышь (и подключена ли вообще), то можно выполнить операцию идентификации мыши. Обычно процедура идентификации начинается с проверки того, подключено ли к порту какое-либо устройство. Вообще говоря, после подачи питания любое устройство, подключенное к последовательному порту, должно выдать сигнал готовности к работе DSR, значение которого можно прочитать в регистре состояния модема MSR по адресу xFEh. Однако для мыши такой метод неприменим, поскольку сигнал DSR обычно отключают с целью сокращения количества проводников в соединительном кабеле мыши.

Чтобы убедиться, что к порту подключена мышь, поддерживающая протокол MS Mouse, нужно временно отключить ее питание (сбросив сигналы DTR и RTS), подождать 0,2 с (4 - 5 тиков системных часов) и подать питание вновь. После этого мышь должна выдать код 4Dh (латинская заглавная буква «M» в кодировке ASCII) или цепочку символов, начинающуюся с этого кода. У трехкнопочной мыши идентификационный код состоит не менее чем из двух символов (начинается с буквы M и следующей за ней цифры 3). Идентификационные коды для основных изготовителей оборудования можно при необходимости найти в спецификации PC 99 System Design Guide [16], однако на рынке постоянно появляются новые игроки.

После того как завершено конфигурирование системы (определен используемый мышью порт) и произведена установка обработчика прерываний, новый драйвер мыши начинает работать. Драйвер должен принимать пакеты данных от мыши, вычислять координаты курсора и управлять перемещением курсора по экрану, то есть выполнять те же операции, что и стандартный драйвер Microsoft Mouse. Однако, в отличие от стандартного, «самодельный» драйвер можно настраивать на работу с любым видеоконтроллером в любом возможном видеорежиме. Кроме того, стандартный драйвер работает только в реальном режиме DOS, а собственные драйверы при необходимости можно написать для любого режима работы процессора x86, например, для защищенного или виртуального.

Процедуры, предназначенные для непосредственной работы с мышью на уровне регистров последовательного порта, собраны в листинге 2.7. Поиск мыши по последовательным портам COM1 и COM2 выполняется процедурой MSMouseSearch. После определения порта, к которому подключена мышь, следует запустить подпрограмму SetMSMoiuseInterrupt, которая настраивает обработчик прерывания (устанавливая параметры порта и соответствующий вектор прерывания), а затем активизирует порт (подавая на мышь напряжение питания и снимая блокировку прерывания). Прерывания обрабатываются процедурой MSMouseInterrupt, принимающей от мыши пакеты данных и вычисляющей приращения координат курсора. После завершения работы с мышью необходимо восстановить старый вектор прерывания, вызвав процедуру RestoreOldMSMouseInterrupt.

Листинг 2.7. Универсальные процедуры для работы с мышью MS Mouse [6] (подключаемый файл list5_01.inc)

```
DATASEG
; Предыдущее значение системного таймера
Time DD ?
; Номер порта (0 - COM1, 1 - COM2,
; 2 - мышь не обнаружена)
COMPortNum DW 0
; Базовый адрес порта мыши
COMPortBaseAddr DW 3F8h
; Область сохранения старого вектора прерывания
OldMSMouseInterruptOffset  DW ?
OldMSMouseInterruptSegment DW ?
; Номер принимаемого от мыши байта
MouseByteNumber DB 0
; Трехбайтовая структура данных, передаваемая мышью
FirstByte       DB 0
SecondByte      DB 0
ThirdByte       DB 0
; Текущее состояние кнопок
ButtonsStatus   DB 0
; Текущие координаты курсора мыши
XCoordinate     DW 0
YCoordinate     DW 0
; Предыдущая позиция курсора мыши
OldXCoordinate  DW 0
OldYCoordinate  DW 0
; Состояние курсора мыши
; (0 - не отображается, 1 - отображается)
MouseCursorStatus DB 0
ENDS
CODESEG
;************************************************
;*      ПОИСК МЫШИ ПО ПОСЛЕДОВАТЕЛЬНЫМ ПОРТАМ     *
;* Процедура подготавливает глобальные переменные *
;* COMPortNum и COMPortBaseAddr для подпрограммы  *
;* установки обработчика прерывания               *
;************************************************
PROC MSMouseSearch NEAR
        pusha
        push    ES
; Запретить прерывание COM1 и COM2
        cli
        in      AL,21h  ;прочитать маску прерываний
        or      AL,18h  ;запретить IRQ3 и IRQ4
        out     21h,AL  ;заменить маску

        sti
; Настроить ES на сегмент данных BIOS
; для работы с системным таймером
        mov     AX,0
        mov     ES,AX

; ПОИСК МЫШИ ЧЕРЕЗ COM-ПОРТЫ
@@MouseSearch:
        ; Устанавливаем скорость
        ; приема/передачи 1200 бод
        mov     DX,[COMPortBaseAddr]
        add     DX,3
        in      AL,DX
        or      AL,80h  ;установить бит DLAB
        out     DX,AL
        mov     DX,[COMPortBaseAddr]
        mov     AL,60h  ;1200 бод
        out     DX,AL
        inc     DX
        mov     AL,0
        out     DX,AL
        ; Установить длину слова 7 бит, 1 стоповый бит,
        ; четность не контролировать
        mov     DX,[COMPortBaseAddr]
        add     DX,3
        mov     AL,00000010b
        out     DX,AL
        ; Запретить все прерывания
        mov     DX,[COMPortBaseAddr]
        inc     DX
        mov     AL,0
        out     DX,AL
; Проверить, что устройство подключено и является
; мышью типа MSMouse
        ; Запомнить текущее время
        mov     EAX,[ES:046Ch]
        mov     [Time],EAX
        ; Отключить питание мыши и прерывания
        mov     DX,[COMPortBaseAddr]
        add     DX,4    ;регистр управления модемом
        mov     AL,0    ;сбросить DTR, RTS и OUT2
        out     DX,AL
        ; Ожидать 5 "тиков" (0,2 с)
@@dT:   mov     EAX,[ES:046Ch]
        sub     EAX,[Time]
        cmp     EAX,5
        jb      @@dT
```

Листинг 2.7. (продолжение)

```
        ; Включить питание мыши
        mov     AL,11b  ;установить DTR и RTS
        out     DX,AL
        ; Очистить регистр данных
        mov     DX,[COMPortBaseAddr]
        in      AL,DX
; Цикл опроса порта
@@WaitData:
        ; Ожидать еще 10 "тиков"
        mov     EAX,[ES:046Ch]
        sub     EAX,[Time]
        cmp     EAX,5+10
        jae     @@NoMouse
        ; Проверить наличие идентификационного байта
        mov     DX,[COMPortBaseAddr]
        add     DX,5
        in      AL,DX
        test    AL,1    ;Данные готовы?
        jz      @@WaitData
        ; Ввести данные
        mov     DX,[COMPortBaseAddr]
        in      AL,DX
        ; Устройство является мышью?
        cmp     AL,'M'
        je      @@End
@@NoMouse:
        inc     [COMPortNum]
        cmp     [COMPortNum],1
        ja      @@End
        sub     [COMPortBaseAddr],100h
        jmp     @@MouseSearch
@@End:  pop     ES
        popa
        ret
ENDP MSMouseSearch
;************************************************
;* УСТАНОВИТЬ ВЕКТОР ПРЕРЫВАНИЯ ОТ ПОСЛЕДОВАТЕЛЬНОГО *
;*           ПОРТА НА НОВЫЙ ДРАЙВЕР МЫШИ          *
;************************************************
PROC SetMSMouseInterrupt NEAR
       pusha
       push    DS
       push    ES
       mov     AX,[CS:MainDataSeg]
       mov     DS,AX
; Запретить прерывание от COM-порта
       ; Вычислить маскируемый разряд
       mov     CL,[byte ptr COMPortNum]
       mov     CH,10h
```

```
        shr     CH,CL   ;маскируемый разряд - в CH
        cli
        in      AL,21h  ;IMR 1-го контроллера прерываний
        or      AL,CH   ;запретить прерывание
        out     21h,AL
        sti
; Обнулить счетчик байтов
        mov     [MouseByteNumber],0
; Установка вектора прерывания на обработчик мыши
        ; Настроить ES на область векторов
        mov     AX,0
        mov     ES,AX
        ; Вычислить адрес вектора
        mov     BX,0Ch
        sub     BX,[COMPortNum]
        shl     BX,2
        ; Сохранить старый вектор
        mov     AX,[ES:BX]
        mov     [OldMSMouseInterruptOffset],AX
        mov     AX,[ES:BX+2]
        mov     [OldMSMouseInterruptSegment],AX
        ; Установить новый вектор
        cli
        mov     AX,offset MSMouseInterrupt
        mov     [ES:BX],AX
        mov     AX,CS
        mov     [ES:BX+2],AX
        sti

; АКТИВИЗИРОВАТЬ ПОРТ
; Разрешаем прерывание по приему данных
        mov     DX,[COMPortBaseAddr]
        inc     DX          ;xF9h
        mov     AL,1
        out     DX,AL
; Установить DTR, RTS и OUT2
; (подать напряжение питания и разрешить прерывания)
        mov     DX,[COMPortBaseAddr]
        add     DX,4        ;xFCh
        mov     AL,00001011b
        out     DX,AL
; Очистить регистр данных
        mov     DX,[COMPortBaseAddr]   ;xF8h
        in      AL,DX
; Разрешить прерывания от COM1
        not     CH
        cli
        in      AL,21h ;IMR 1-го контроллера прерываний
        and     AL,CH  ;разрешить прерывания
        out     21h,AL
        sti
        pop     ES
        pop     DS
        popa
        ret
ENDP SetMSMouseInterrupt

;*********************************************
;* ВОССТАНОВИТЬ СТАРЫЙ ВЕКТОР ПРЕРЫВАНИЯ МЫШИ *
;*********************************************
PROC RestoreOldMSMouseInterrupt NEAR
        pusha
        push    ES
        ; Настроить ES на область векторов
        mov     AX,0
        mov     ES,AX
        ; Вычислить адрес вектора
        mov     BX,0Ch
        sub     BX,[COMPortNum]
        shl     BX,2
        ; Восстановить старый вектор
        cli
        mov     AX,[OldMSMouseInterruptOffset]
        mov     [ES:BX],AX
        mov     AX,[OldMSMouseInterruptSegment]
        mov     [ES:BX+2],AX
        sti
        pop     ES
        popa
        ret
ENDP RestoreOldMSMouseInterrupt

;*************************************
;* НОВЫЙ ОБРАБОТЧИК ПРЕРЫВАНИЯ ОТ МЫШИ *
;*************************************
PROC MSMouseInterrupt NEAR
        pusha
        push    DS
        sti     ;разрешить маскируемые прерывания
        mov     AX,[CS:MainDataSeg]
        mov     DS,AX

; Проверить наличие данных
        mov     DX,[COMPortBaseAddr]
        add     DX,5          ;xFDh
        in      AL,DX
        test    AL,1   ;Данные готовы?
        jz      @@Error
; Ввести данные
        mov     DX,[COMPortBaseAddr]   ;xF8h
        in      AL,DX
; Разблокировать контроллер прерываний
        mov     AH,AL  ;запомнить данные в AH
        mov     AL,20h
        out     20h,AL ;послать команду EOI
        mov     AL,AH  ;восстановить данные в AL
; Сбросить старший незначащий бит
        and   AL,01111111b

; Определить порядковый номер принимаемого байта
        cmp     [MouseByteNumber],0
        je      @@FirstByte
        cmp     [MouseByteNumber],1
        je      @@SecondByte
        cmp     [MouseByteNumber],2
        je      @@ThirdByte
        jmp     @@Error

; Сохранить первый байт данных
@@FirstByte:
        test    AL,1000000b       ;Первый байт посылки?
        jz      @@Error
        mov     [FirstByte],AL
        inc     [MouseByteNumber] ;увеличить счетчик
        jmp     @@EndMouseInterrupt
; Сохранить второй байт данных
@@SecondByte:
        test    AL,1000000b
        jnz     @@Error
        mov     [SecondByte],AL
        inc     [MouseByteNumber] ;увеличить счетчик
        jmp     @@EndMouseInterrupt
; Сохранить третий байт данных
@@ThirdByte:
        test    AL,1000000b
        jnz     @@Error
        mov     [ThirdByte],AL
        mov     [MouseByteNumber],0 ;Обнуление счетчика
; (Пакет данных от мыши принят полностью).

; Записать новое значение состояния кнопок мыши
        mov     AL,[FirstByte]


        shr     AL,4
        mov     [ButtonsStatus],AL

; Прибавить перемещение по X к координате X
        mov     AL,[FirstByte]
        shl     AL,6
```

```
        or      AL,[SecondByte]
        cbw
        add     AX,[XCoordinate]
        ; Курсор не должен выходить за левую или
        ; правую границу экрана
        js      @@X1
        cmp     AX,ScreenLength
        jb      @@X2
        ; Установить координату X по правой границе
        mov     AX,ScreenLength-1
        jmp     @@X2
@@X1:   ; Установить координату X по левой границе
        xor     AX,AX
@@X2:   mov     [XCoordinate],AX

; Прибавить перемещение по Y к координате Y
        mov     AL,[FirstByte]
        and     AL,00001100b
        shl     AL,4
        or      AL,[ThirdByte]
        cbw
        add     AX,[YCoordinate]
        ; Курсор не должен выходить за верхнюю или
        ; нижнюю границу экрана
        js      @@Y1
        cmp     AX,ScreenHeigth
        jb      @@Y2
        ; Установить координату Y по нижней границе
        mov     AX,ScreenHeigth-1
        jmp     @@Y2
@@Y1:   ; Установить координату Y по верхней границе
        xor     AX,AX
@@Y2:   mov     [YCoordinate],AX

; Показать курсор в новой позиции
        cmp     [MouseCursorStatus],0
        je      @@EndMouseInterrupt
        call    ShowNewMouseCursorPosition
        jmp short @@EndMouseInterrupt

@@Error:
; Произошел сбой в порядке передачи информации от
; мыши, обнулить счетчик байтов пакета данных
        mov     [MouseByteNumber],0
 @@EndMouseInterrupt:
         pop    DS
         popa
         iret
 ENDP MSMouseInterrupt
 ENDS
```

Приведенная в листинге 2.8 программа MSMouseMain [6], демонстрирующая работу с мышью через последовательный порт, использует процедуры из листинга 2.7 (подключаемый файл list5_01.inc). Для вывода курсора на экран вызывается процедура ShowNewMouseCursorPosition, которая инвертирует байт атрибута символа в позиции курсора (запоминая предварительно старое значение атрибута и восстанавливая его при перемещении курсора). Основная программа контролирует (в цикле) состояние левой кнопки мыши и завершает свою работу при нажатии на нее, восстанавливая предварительно старый вектор прерывания.

ПРИМЕЧАНИЕ

Данный пример не предъявляет никаких особых требований к составу оборудования. Он может быть запущен на любом AT-совместимом компьютере, к которому подключена мышь типа MS Mouse.

Листинги подключаемых файлов "list1_02.inc", "list1_03.inc" и "list1_04.inc" представлены в разделе 1.2.7.1.

Листинг 2.8. Поиск мыши MS Mouse и установка обработчика прерывания [6]

```
IDEAL
P386
LOCALS
MODEL MEDIUM

; Параметры экрана в текстовом режиме
ScreenLength equ 80  ;количество символов в строке
ScreenHeigth equ 25  ;количество строк на экране

; Подключить файл мнемонических обозначений
; кодов управляющих клавиш и цветовых кодов
include "list1_03.inc"
; Подключить файл макросов
include "list1_04.inc"

SEGMENT sseg para stack 'STACK'
DB 400h DUP(?)
ENDS

DATASEG
; Старое значение фона символа

 OldCharBackground DB 0Fh
 ; Текстовые сообщения
 Txt0 DB YELLOW,0,24
      DB "Тестирование мыши типа MS Mouse",0
      DB YELLOW,24,35,"Ждите ...",0
 Txt1 DB WHITE,9,26,"Мышь подключена к порту COM1",0
 Txt2 DB WHITE,9,26,"Мышь подключена к порту COM2",0
 Txt3 DB LIGHTGREEN,7,32,"Результаты поиска:",0
      DB LIGHTCYAN,15,26
      DB "Драйвер MS Mouse установлен",0
      DB LIGHTBLUE,17,9,"Отображение курсора "
      DB "осуществляется инверсией атрибута символа",0
      DB YELLOW,24,21
      DB "Для выхода нажмите левую клавишу мыши",0
 Txt4 DB 12,26,"Мышь MS Mouse не обнаружена",0
 ENDS

 CODESEG
 ;*****************************
 ;* Основной модуль программы *
 ;*****************************
 PROC MSMouseMain
        mov     AX,DGROUP
        mov     DS,AX
        mov     [CS:MainDataSeg],AX
 ; Установить текстовый режим и очистить экран
        mov     AX,3
        int     10h
 ; Скрыть текстовый курсор
        mov     [ScreenString],25
        mov     [ScreenColumn],0
        call    SetCursorPosition
 ; Вывести текстовые сообщения на экран
        MShowColorText 2,Txt0
 ; Произвести поиск мыши по последовательным портам
        call    MSMouseSearch
        cmp     [COMPortNum],1  ;номер порта больше 1?
        ja      @@MouseNotFound ;мышь не найдена
 ; Выдать сообщение об обнаружении мыши
        cmp     [COMPortNum],0
        jne     @@COM2
        mov     SI,offset Txt1
        jmp     @@COM1
 @@COM2: mov    SI,offset Txt2
 @@COM1: call   ShowColorString
```

```
; Выдать сообщение о запуске драйвера
        MShowColorText 4,Txt3

; ДЕМОНСТРАЦИЯ РАБОТЫ ДРАЙВЕРА МЫШИ
; Обнулить позицию курсора мыши

        mov     [XCoordinate],0
        mov     [YCoordinate],0
        mov     [OldXCoordinate],0
        mov     [OldYCoordinate],0
; Отобразить курсор мыши первый раз
        mov     [MouseCursorStatus],1
        call    ShowNewMouseCursorPosition
; Установить новый обработчик прерывания
        call    SetMSMouseInterrupt
; Цикл, пока не нажата левая кнопка
@@Next: test    [ButtonsStatus],10b
        jz      @@Next
; Восстановить прежний обработчик прерывания
        call    RestoreOldMSMouseInterrupt
; Переустановить текстовый режим и очистить экран
@@End:  mov     AX,3
        int     10h
; Выход в DOS
        mov     AH,4Ch
        int     21h
; Выдать сообщение "мышь не обнаружена"
@@MouseNotFound:
        MFatalError Txt4
ENDP MSMouseMain
;*****************************************
;* ОТОБРАЖЕНИЕ КУРСОРА МЫШИ ПУТЕМ ИНВЕРСИИ *
;*   АТРИБУТА СИМВОЛА В ПОЗИЦИИ КУРСОРА    *
;*****************************************
PROC ShowNewMouseCursorPosition NEAR
        pusha
        push    ES
        ; Настроить ES на видеопамять
        mov     AX,0B800h
        mov     ES,AX
        ; Вычислить старую координату курсора
        mov     AX,[OldYCoordinate]
        mov     DX,160
        mul     DX
        add     AX,[OldXCoordinate]
        add     AX,[OldXCoordinate]
        inc     AX
        mov     DI,AX
        ; Восстановить атрибут символа
        mov     AL,[OldCharBackground]
        mov     [ES:DI],AL
        ; Вычислить новую координату курсора
        mov     AX,[YCoordinate]
        mov     DX,160
        mul     DX
        add     AX,[XCoordinate]
        add     AX,[XCoordinate]
        inc     AX
        mov     DI,AX
        ; Сохранить атрибут символа
        mov     AL,[ES:DI]
        mov     [OldCharBackground],AL
        ; Инвертировать атрибут
        xor     [byte ptr ES:DI],1111111b
        ; Запомнить координаты символа
        mov     AX,[XCoordinate]
        mov     [OldXCoordinate],AX
        mov     AX,[YCoordinate]
        mov     [OldYCoordinate],AX
        pop     ES
        popa
        ret
ENDP ShowNewMouseCursorPosition
ENDS
; Подключить процедуры вывода данных на экран
include "list1_02.inc"
; Подключить универсальные процедуры для работы
; с мышью MS Mouse
include "list5_01.inc"

END
```

2.1.4.3. Непосредственная работа с мышью типа MS Mouse в среде AFD.

В листинге 2.9 приведен программный модуль, реализующий: инициализацию порта COM1, инициализацию вектора прерывания INT 0Ch (IRQ 4), разрешение прерываний от порта COM1, включение питания мыши и ожидание заполнения обработчиком прерывания буфера памяти (модуль CMPR.BIN), а в листинге 2.10 - программный модуль обработчика прерывания INT 0Ch (модуль CMINT.BIN). Программные модули сориентированы на их выполнение в среде отладчика AFD. Первый модуль загружается с адреса CS:0200h, а обработчик - с адреса CS:0100h. Запуск модуля CMPR.BIN осуществляется командой отладчика G 0200. Он прекращает свою работу после заполнения обработчиком прерывания INT 0Ch (модуль CMINT.BIN) буфера памяти размером 64 байта с базовым адресом DX:0000h и содержимое буфера отображается в окне памяти экрана отладчика AFD. После заполнения буфера питание мыши отключается и запрещаются прерывания от контроллера порта COM 1.

Листинг 2.9 Программный модуль CMPR.TXT

```
2C06:0200  BAFB03    MOV    DX,03FB ;Установка в регистре LCR (адр. 3FBh)
2C06:0203  B080      MOV    AL,80   ;признака SET=1 (D7), разрешающего обраще-
2C06:0205  EE        OUT    DX,AL   ;ние к регистрам делителя.
2C06:0206  B060      MOV    AL,60   ;Установка скорости приема/передачи 1200
2C06:0208  BAF803    MOV    DX,03F8 ;бод, занесением в младший байт делителя
2C06:020B  EE        OUT    DX,AL   ;константы 60h,
2C06:020C  B000      MOV    AL,00   ;а в старший
2C06:020E  BAF903    MOV    DX,03F9 ;байт делителя
2C06:0211  EE        OUT    DX,AL   ;константы 00h.
2C06:0212  BAFB03    MOV    DX,03FB ;Установка режима (без контроля паритета,
2C06:0215  B006      MOV    AL,06   ;2 стоповых бите и 7 бит в поле данных),
2C06:0217  EE        OUT    DX,AL   ;занесением в LCR константы 06h.
2C06:0218  BAF903    MOV    DX,03F9 ;Запрещение выработки контроллером порта
2C06:021B  B000      MOV    AL,00   ;COM1 сигнала IRQ4 по всем событиям,
2C06:021D  EE        OUT    DX,AL   ;происходящим в порту (00h → IER).
2C06:021E  E421      IN     AL,[21] ;Разрешение обработки прерывания IRQ4
2C06:0220  24EF      AND    AL,EF   ;ведущим контроллером прерываний
```

```
2C06:0222  EB00      JMP    0224    ;установкой в единицу разряда D4
2C06:0224  EB00      JMP    0226    ;его регистра маски (адрес 21h).
2C06:0226  E621      OUT    [21],AL
2C06:0228  B80000    MOV    AX,0000 ;Занесение в регистр сегмента ES CPU
2C06:022B  8EC0      MOV    ES,AX   ;адреса 0000h.
2C06:022D  B80001    MOV    AX,0100 ;Формирование вектора прерывания INT 0Ch
2C06:0230  26A33000  MOV    ES:[0030],AX  ;по адресу 0000:0030h, указывающего
2C06:0234  8CC8      MOV    AX,CS   ;адрес обработчика прерывания, начинающе-
2C06:0236  26A33200  MOV    ES:[0032],AX  ;гося с адреса CS:0100h.
2C06:023A  BAFD03    MOV    DX,03FD ;Сброс флаговых разрядов регистра LSR
2C06:023D  EC        IN     AL,DX   ;его чтением по адресу 3FDh.
2C06:023E  BAFE03    MOV    DX,03FE ;Сброс флаговых разрядов регистра MSR
2C06:0241  EC        IN     AL,DX   ;его чтением по адресу 3FEh.
2C06:0242  BAFA03    MOV    DX,03FA ;Запрещение использования буфера FIFO
2C06:0245  B006      MOV    AL,06   ;в контроллере COM порта, записью
2C06:0247  EE        OUT    DX,AL   ;кода 06h по адресу 3FAh (регистр FCR).
2C06:0248  BAF903    MOV    DX,03F9 ;Разрешить выработку запроса прерывания
2C06:024B  B001      MOV    AL,01   ;IRQ4 при очередном приеме байта данных
2C06:024D  EE        OUT    DX,AL   ;во входной буферный регистр RBR.
2C06:024E  B90000    MOV    CX,0000 ;Сброс счетчика принятых байтов.
2C06:0251  B00B      MOV    AL,0B   ;Подача питания на мышь и разрешение
2C06:0253  BAFC03    MOV    DX,03FC ;генерации контроллером сигнала IRQ4
2C06:0256  EE        OUT    DX,AL   ;по событию, вызывающему прерывание.
2C06:0257  F6C180    TEST   CL,40   ;Проверка заполнения обработчиком
2C06:025A  74FB      JZ     0257    ;буфера памяти, размером 64 байта.
2BF9:025C  B000      MOV    AL,00   ;Отключение питания мыши (+12 В)
2BF9:025E  EE        OUT    DX,AL   ;и запрещение сигнала прерывания.
2BF9:025F  CC        INT3           ;Останов и передача управления в AFD.
```

Листинг 2.10. Обработчик прерывания INT 0Ch (линия IRQ4) (CMINT.BIN).

```
2C06:0100  52        PUSH   DX      ;Запоминание в стеке
2C06:0101  57        PUSH   DI      ;содержимого
2C06:0102  50        PUSH   AX      ;общих регистров
2C06:0103  06        PUSH   ES      ;процессора: DX,DI, AX, ES.
2C06:0104  BAFD03    MOV    DX,03FD ;Сброс флаговых разрядов регистра LSR
2C06:0107  EC        IN     AL,DX   ;его чтением по адресу 3FDh.
2C06:0108  BAFE03    MOV    DX,03FE ;Сброс флаговых разрядов регистра MSR
2C06:010B  EC        IN     AL,DX   ;его чтением по адресу 3FEh.
2C06:010C  8CD8      MOV    AX,DS   Позиционирование на сегмент буфера
2C06:010E  8EC0      MOV    ES,AX   ;памяти обработчика.
2C06:0110  B80000    MOV    AX,0000 ;Очистка регистра AX.
2C06:0113  BAF803    MOV    DX,03F8 ;Чтение очередного байта данных из
2C06:0116  EC        IN     AL,DX   ;буферного регистра RBR приемника.
2C06:0117  89CF      MOV    DI,CX   Позиционирование на очередной байт буф.
2C06:0119  268805    MOV    ES:[DI],AL  ;Запись очередного байта в буфер.
2C06:011C  41        INC    CX      ;Инкрементирование счетчика байтов.
2C06:011D  B020      MOV    AL,20   ;Посылка команды EOI (End Of Interrupt)
2C06:011F  E620      OUT    [20],AL ;в ведущий контроллер прерыв. (адр. 20h).
2C06:0121  07        POP    ES      ;Восстановление из стека
2C06:0122  58        POP    AX      ;содержимого
2C06:0123  5F        POP    DI      ;общих регистров
2C06:0124  5A        POP    DX      ;процессора: ES, AX, DI, DX.
2C06:0125  FB        STI            ;Разрешение прерываний.
2C06:0126  CF        IRET           ;Возврат из обработчика прерываний
```

В данном разделе использовались материалы из [3 - 7, 9, 19].

Контрольные вопросы.

1. К какому типу устройств относятся манипуляторы мышь?
2. Какую информацию передает в систему COM мышь?
3. Как преобразуется механическое перемещение мыши в электрические сигналы?
4. Что понимается под названием "шинная мышь"?
5. Что представляет собой манипулятор трекбол?
6. Охарактеризуйте оптические манипуляторы мышь.

7. Что понимается под 3D-мышью?
8. По каким интерфейсам могут подключаться к системе манипуляторы мышь?
9. Чем отличается Serial Mouse от PS/2-Mouse?
10. Можно ли подключить PS/2-Mouse к COM-порту через переходник?
11. Можно ли подключить USB-мышь через переходник к порту PS/2-мыши?
12. Как реагирует ОС на подключение к системе нескольких устройств-указателей?
13. Охарактеризуйте основные причины сбоев в работе манипуляторов мышь.
14. Охарактеризуйте особенности "беспроводных мышей".
15. Как подключается к ПК Serial Mouse?
16. Охарактеризуйте две разновидности Serial Mouse.
17. Охарактеризуйте системную поддержку Serial Mouse на уровне BIOS и DOS.
18. Охарактеризуйте электрическую схему универсальной Serial Mouse.
19. Охарактеризуйте сигналы в контрольных точках электрической схемы Serial Mouse.
20. Дайте краткую характеристику COM-портам ПК.
21. Дайте краткую характеристику интерфейсу RS-232C.
22. Какие типы разъемов RS-232C используются в COM-портах ПК?
23. Что понимается под нуль-модемным кабелем?
24. Охарактеризуйте сигналы COM-порта ПК.
25. Охарактеризуйте последовательность управляющих сигналов протокола RS-232C.
26. Для чего в COM-портах используются преобразователи уровней?
27. Охарактеризуйте правила, по которым производится инверсия сигналов в COM-портах ПК.
28. Охарактеризуйте родственные RS-232C интерфейсы RS-422A и RS-423A.
29. Охарактеризуйте интерфейс RS-485.
30. Охарактеризуйте интерфейс "Токовая петля".
31. Охарактеризуйте формат посылки байта асинхронного режима RS-232C.
32. Каковы требования к рассогласованию скоростей передачи и приема в асинхронном режиме RS-232C?
33. Какие ошибки передачи позволяет выявить формат асинхронной посылки?
34. Что понимается под выражением: "количество стоп-битов равно полтора бита"?
35. В чем разница между квитированием и управлением потоком?
36. Охарактеризуйте аппаратный протокол управления потоком RTS/CTS.
37. Охарактеризуйте программный протокол управления потоком XON/XOFF.
38. Какие микросхемы асинхронных приемопередатчиков использовались в COM-портах в PC/XT и PC/AT?
39. Какие операции должен выполнять УАПП при передаче символа?
40. Какие операции должен выполнять УАПП при приеме символа?
41. Охарактеризуйте УАПП семейства 8250 / 16540.
42. Охарактеризуйте функциональные возможности УАПП 8250 / 16450.
43. Охарактеризуйте блок-схему УАПП 8250 / 16450 (рис. 2.17).
44. Охарактеризуйте особенности УАПП 16550.
45. Какие функции выполняет УАПП 16550?
46. Охарактеризуйте блок-схему УАПП 16550 (рис. 2.18).
47. Перечислите программно-доступные регистры УАПП 8250 / 16450.
48. Чем отличается регистровая архитектура УАПП 16550 от архитектуры 8250 / 16450?
49. Охарактеризуйте регистры THR и RBR.
50. Охарактеризуйте регистры DLL и DLM.
51. Охарактеризуйте регистр IER.
52. Охарактеризуйте регистр IIR.
53. Охарактеризуйте регистр FCR.
54. Охарактеризуйте регистр LCR.
55. Охарактеризуйте регистр MCR.
56. Охарактеризуйте регистр LSR.
57. Охарактеризуйте регистр MSR.
58. Охарактеризуйте диагностический режим УАПП.
59. В чем заключается системная поддержке COM-портов на уровне BIOS?
60. Что понимается под инициализацией COM-портов на стадии POST?
61. Какие функции поддерживает драйвер COM-порта INT 14h?
62. Каково назначение ячеек памяти рабочей области данных BIOS, выделяемых под COM-порты?
63. Дайте краткую характеристику драйверу COM-порта INT 14h.
64. Охарактеризуйте функцию 00h прерывания INT 14h.
65. Охарактеризуйте функции 01h и 02h прерывания INT 14h.
66. Охарактеризуйте функцию 03h прерывания INT 14h.
67. Что понимается под конфигурированием COM-портов?
68. Охарактеризуйте области использования COM-портов.
69. Как поддерживается спецификация PnP устройствами, подключаемыми к COM-портам?

70. Как маркируется строка идентификатора PnP, принимаемая через COM-порт?
71. Как проверяется отключение устройства PnP, подключенного к COM-порту?
72. Охарактеризуйте ошибки конфигурирования COM-портов.
73. Охарактеризуйте особенности функционального тестирования COM-портов.
74. Охарактеризуйте особенности питания маломощных устройств от выходных линий COM-порта.
75. Охарактеризуйте стандартный формат передачи данных манипулятора NS Mouse.
76. Охарактеризуйте формат Microsoft Plus (M+) трехкнопочной мыши.
77. Охарактеризуйте формат 3D Serial Mouse.
78. Охарактеризуйте нестандартный 8-разрядный формат MS Mouse с двумя альтернативными протоколами передачи данных.
79. Охарактеризуйте формат PC Mouse.
80. Охарактеризуйте требования к средствам работы с Serial Mouse.
81. Кто устанавливает обработчик прерывания INT 33h?
82. Какие функции выполняет прерывание INT 33h?
83. Охарактеризуйте функцию 0000h прерывания INT 33h.
84. Охарактеризуйте функцию 0001h прерывания INT 33h.
85. Охарактеризуйте функцию 0002h прерывания INT 33h.
86. Объясните работу программы, представленной в листинге 2.1.
87. Охарактеризуйте функцию 0003h прерывания INT 33h.
88. Объясните работу программы, представленной в листинге 2.2.
89. Охарактеризуйте функцию 0004h прерывания INT 33h.
90. Охарактеризуйте функцию 0005h прерывания INT 33h.
91. Охарактеризуйте функцию 0006h прерывания INT 33h.
92. Охарактеризуйте функцию 0007h и 0008h прерывания INT 33h.
93. Охарактеризуйте функцию 0009h прерывания INT 33h.
94. Объясните работу программы, представленной в листинге 2.3.
95. Охарактеризуйте функцию 000Ah прерывания INT 33h.
96. Объясните работу программы, представленной в листинге 2.4.
97. Охарактеризуйте функцию 000Bh прерывания INT 33h.
98. Охарактеризуйте функцию 000Ch прерывания INT 33h.
99. Объясните работу программы, представленной в листинге 2.5.
100. Объясните работу программы, представленной в листинге 2.6.
101. Охарактеризуйте функцию 000Dh, 000Eh и 000Fh прерывания INT 33h.
102. Охарактеризуйте функцию 0010h и 0012h прерывания INT 33h.
103. Охарактеризуйте функцию 0013h и 0014h прерывания INT 33h.
104. Охарактеризуйте функцию 0015h, 0016h и 0017h прерывания INT 33h.
105. Охарактеризуйте функцию 0018h прерывания INT 33h.
106. Охарактеризуйте функцию 0019h и 001Ah прерывания INT 33h.
107. Охарактеризуйте функцию 001Bh и 001Ch прерывания INT 33h.
108. Охарактеризуйте функцию 001Dh и 001Fh прерывания INT 33h.
109. Охарактеризуйте функцию 0020h и 0021h прерывания INT 33h.
110. Охарактеризуйте функцию 0024h прерывания INT 33h.
111. Охарактеризуйте процесс установки нового драйвера мыши.
112. Как разрешается прерывание от Serial Mouse?
113. Как настраивается вектор прерывания COM-порта?
114. Как подается питание на Serial Mouse и разрешение контроллеру COM-порта генерировать запросы на прерывание?
115. Как определить: подключена ли к COM-порту мышь?
116. Какие действия выполняет "новый" драйвер мыши?
117. Объясните работу процедур, представленных в листинге 2.7.
118. Объясните работу программы, представленной в листинге 2.8.
119. Объясните работу программного модуля, представленного в листинге 2.9.

3. Игровой порт

В большинстве настольных компьютеров есть встроенный игровой порт, куда подключаются один или два джойстика - с двумя цифровыми и двумя аналоговыми входными линиями (одно устройство) или с четырьмя цифровыми и четырьмя аналоговыми (два устройства). Цифровые линии позволяют считывать входные данные, а аналоговые - измерять подключенное к ним сопротивление (оно должно быть в пределах 0 - 100 кОм).

Джойстик имеет два потенциометра 100 кОм для индикации координат X и Y, а также две нормально разомкнутые кнопки. На соответствующих им линиях порта устанавливается высокий уровень, а при нажатии на кнопку - низкий.

3.1. Разъем

Порт имеет 15-контактную розетку D-типа. Назначение выводов приведено в таблице 3.1, а типовое соединение с джойстиком - на рис. 3.1.

Таблица 3.1. Контакты разъема, их наименование и назначение.

Номера контакто&	Наименование	Описание
1. 8. 9, 15	+5В	Питание со стороны ПК
4, 5, 12	GND	"Земля"
2	BA1	Первая кнопка джойстика А
3	AX	X—потенциометр джойстика А
6	AY	Y—потенциометр джойсrnuка А
7	BA2	Вторая кнопка джойстика А
10	BB1	Первая кнопка джойстика В
11	BX	X—потенциометр джойстика В
13	BY	Y—потенциометр джойстика В
14	BB2	Вторая кнопка джойстика В

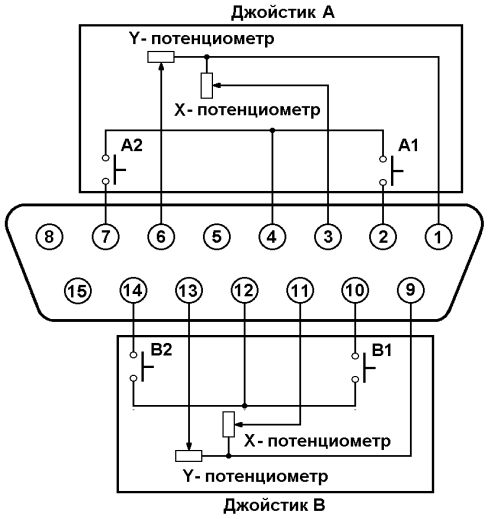

Рис. 3.1. Соединение контактов порта с резисторами и переключателями джойстика.

На звуковых картах через разъем игрового порта вместе с джойстиками могут подключаться и внешние MIDI-устройства, используя специальный кабель-адаптер, обеспечивающий гальваническую развязку входного сигнала и ограничение выходного тока [4, 5]. Для интерфейса MIDI используются контакты 12 (MIDI Out (Tx) — выход) и 15 (MIDI In (Rx) — вход), ранее предназначавшиеся для шин GND и +5V. Такое назначение делает безопасным подключение адаптера MIDI к «чистому» игровому порту и обычного джойстика к игровому порту с сигналами MIDI.

3.2. Внутреннее устройство игрового порта

Внутренняя электрическая схема типового игрового порта изображена на рис. 3.2, а логическая структура - на рис. 3.3.

Восьмибитовые линии шины данных состоят из четырех выходов микросхемы DD5a и четырех выходов микросхемы DD5b. Четыре входа состояния кнопок подключены к питанию +5В через нагрузочные резисторы R9 - R12 величиной 4,7 кОм и поступают на входы шинного формирователя с тремя состояниями на выходе (DD5b).

В адресном пространства портов ввода/вывода PC-совместимых ПК игровой порт имеет адрес 201 h. Функции битов порта приведены ниже:

D7	D6	D5	D4	D3	D2	D1	D0
BB2	BB1	BA2	BA1	BY	BX	AY	AX

BB2, BB1, BA2 и BA1 - цифровые входы; BY, BX, AY и AX - аналоговые входы.

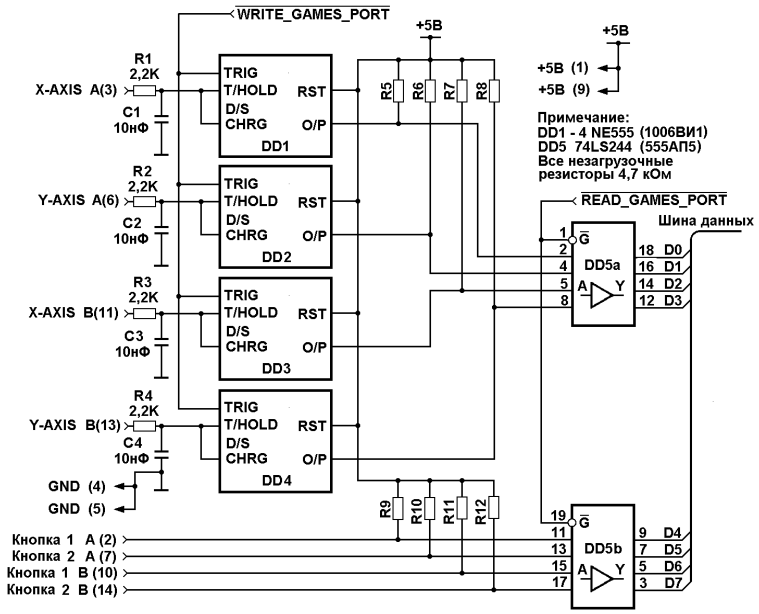

Рис. 3.2. Внутренняя электрическая схема типового игрового порта

Метод измерения сопротивления основан на программном определении длительности импульса, пропорционального сопротивлению. Преобразование начинается после вывода любого байта в регистр порта (201h), при этом биты 0 - 3 устанавливаются в 1. Время измеряется до возврата этих битов в нулевое состояние. Каждый бит соотносится с соответствующим аналоговым каналом.

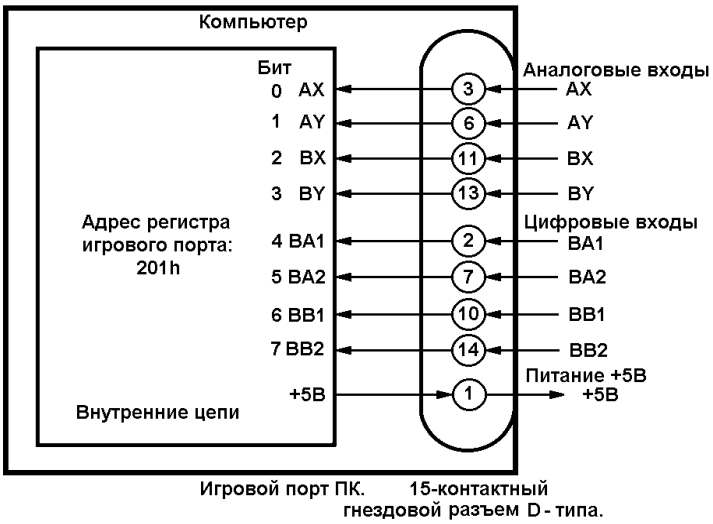

Рис. 3.3. **Логическая структура игрового порта**

Если аналоговый вход закорочен на «землю» или цепь измеряемого сопротивления разорвана, соответствующий бит не обнулится. Поэтому в программе преобразования должен быть предусмотрен тайм-аут. Взаимосвязь временного интервала и подключенного сопротивления характеризуется следующей формулой [7]:

$$\text{Сопротивление (Ом)} = \frac{\text{Временной интервал} - 24{,}2\ \text{мкс}}{0{,}011} \qquad (3.1)$$

Временной интервал изменяется в пределах от 24,2 мкс до 1124 мкс при изменении сопротивления от 0 до 100кОм. Однако погрешности внутреннего конденсатора и резистора делают формулу неточной. На практике необходимо производить калибровку, которая включает в себя измерение интервалов времени, когда входное сопротивление близко к нулю или равно 100 кОм.

Проверка автором соответствия формулы (3.1) реальным измерениям показала, что для некоторых компьютеров их результатам более соответствует ниже приведенная формула (3.2). Причем, если время измерялось с помощью счетчика канала 0 системного таймера, то более точные результаты получались по формуле (3.1), а если измерения велись с помощью счетчика канала 2, то более точные результаты получаются по формуле (3.2).

$$\text{Сопротивление (Ом)} = \frac{\text{(Временной интервал)/2 - 24,2 мкс}}{0{,}011} \qquad (3.2)$$

3.3. Программное управление

Ниже приводятся примеры реализации программного взаимодействия с джойстиком через игровой порт, реализованные на различных языках программирования.

3.3.1. QBASIC.

В языке программирования QBASIC существуют два оператора, предназначенные для игрового порта, - STICK(x) и STRIG(x). Оператор STICK(x) может принимать значения 0 - 3 и используется для чтения информации с потенциометров X и Y джойстиков A и B.

X=0 считывает X-координату джойстика A
X=1 считывает Y-координату джойстика A
X=2 считывает X-координату джойстика B
X=3 считывает Y-координату джойстика B

Применяя эту команду, необходимо сначала вызвать STICK(0), а затем STICK(1), STICK(2) и STICK(3). Функция STICK возвращает значение координаты, которое изменяется от 6 (нулевое сопротивление) до 150 (сопротивление 100 кОм).

STRIG(x) возвращает -1, если условие верно; в противном случае функция возвращает 0. X может принимать значения от 0 до 7 и используется для выбора определенного состояния кнопки джойстика.

X=0 Первая кнопка джойстика A была нажата с момента последней команды STRIG(0)
X=1 Первая кнопка джойстика A была только что нажата
X=2 Первая кнопка джойстика B была нажата с момента последней команды STRIG(2)
X=3 Первая кнопка джойстика B была только что нажата
X=4 Вторая кнопка джойстика A была нажата с момента последней команды STRIG(4)
X=5 Вторая кнопка джойстика B была только что нажата
X=6 Вторая кнопка джойстика A была нажата с момента последней команды STRIG(6)
X=7 Вторая кнопка джойстика B была только что нажата

Следующая программа на языке QBASIC выводит на экран координаты X и Y джойстика A и показывает состояния двух кнопок [7].

```
10 dummy=STICK(0)
20 print"Coordinate of X:",STICK(0)
30 print"Coordinate of Y:",STICK(1)
40 print"Current status of 1st button:",STRIG(1)
50 print"Current status of 2nd button:",STRIG(5)
60 end
```

3.3.2. Turbo Pascal 6.

Функция на языке Turbo Pascal 6 (TP6), приведенная ниже (листинг 3.1), возвращает состояние конкретного бита, определенного переменной bitx. Содержимое регистра игрового порта считывается в компьютер и присваивается переменной Input_byte. Затем состояние конкретного бита определяется наложением маски на считанный байт [7].

Листинг 3.1. Чтение регистра игрового порта

```
(*-Библиотека ресурсов № A14 (чтение регистра игрового порта).-*)
Function read_game_port(bitx:integer):integer;
(*Адрес игрового порта: 201h
Bitx (1-8) выбирает состояние AX, AY, BX, BY, BA1, BA2, BB1 и BB2.*)
var
   input_byte:byte;
begin
   input_byte:=port($201);
   read_game_port:=round((input_byte and bit_weight(bitx))/bit_weight(bitx));
end;
```

Чтобы определить величину сопротивления, необходимо записать любой байт в порт 201h для запуска одновибраторов. Запущенные одновибраторы устанавливают соответствующие биты в регистре порта 201h в единичное состояние на время, определяемое величиной сопротивления потенциометра соответствующей координаты джойстиков. Нужно постоянно опрашивать интересующий вас бит, пока он снова не перейдет в нулевое состояние. В результате получим требуемый временной интервал. Наиболее удобно измерять временной интервал с помощью второго канала таймера i8254 компьютера, который можно настроить как независимый таймер с обратным отсчетом времени. Если установить во втором канале режим с автозагрузкой (режим 2 -генератор импульсов, или режим 3 - генератор импульсов со скважностью 2) а в младший и старший байты счетчика записать число 255 (0FFh), то его значение будет обнуляться каждые 55 мс. Это позволяет измерять временные интервалы в "тиках" величиной 0,84 мкс каждый.

Разрешается применять только счетчик второго канала, так как первые два канала уже используются операционной системой компьютера (нулевой канал - для поддержки системного времени, первый - для регенерации динамической памяти). Можно пользоваться и счетчиком нулевого канала, который настраивается BIOS на стадии POST на двоичный счет в режиме 3, с записью в младший и старший байты счетчика чисел 255 (0FFh). Однако в этом случае разрешается только чтение содержимого счетчика канала без операций задания режима работы канала. Пример такой программы, написанной на ассемблере представлен в [15].

Следующая функция на языке TP6 (листинг 3.2) позволяет измерить временной интервал [7]. Переменная **x** указывает, по какому аналоговому входу он будет определяться. Функция port($201):=0 выполняет операцию записи в игровой порт для запуска одновибратора. Сразу после этого считывается значение счетчика таймера 8253/8254 и полученный результат присваивается переменной Time1. Потом в цикле постоянно проверяется, когда соответствующий бит регистра игрового порта станет равным нулю. Как только это произойдет, значение счетчика таймера снова считывается и присваивается переменной Time2. Затем вычисляется временной интервал в "тиках" величиной 0,84 мкс. Данная функция использует процедуру init_8253 и функцию Read_8253. Первая записывает число 255 в старший и младший разряды счетчика канала 2 и настраивает его как независимый счетчик, вторая считывает старший и младший байты счетчика канала 2.

Листинг 3.2. Функция измерения временного интервала.

```
(*-Библиотека ресурсов № A16 (получение длительности периода одновибратора).-*)
Function interval_game_port(x:integer):integer;
(*x выбирает AX(x=1), AY(x=2), BX(x=3), BY(x=4).*)
var
   time1,time2,dummy:integer;

Procedure init_8253;
(*Настройка 8253.*)
begin
(* Управляющее слово = b6h = 10110110b:
   10 = выбор счетчика 2;
   11 = сначала чтение/запись младшего байта, затем старшего;
   011 = режим 3;
   0 = шестнадцатибитовый двоичный счет.*)
   port($43):=$b6;                  (*Запись управляющего слова в управляющий регистр 8253.*)
   port($42):=255;                  (*Загрузка младшего байта.*)
   port($42):=255;                  (*Загрузка старшего байта.*)
   port($61):=port($61) or 1;       (*Разрешение работы счетчика и запрет динамика.*)
end;

Function read_8253:integer;

(*Считывание младшего и старшего байтов счетчика.*)
var
   low_byte,high_byte:byte;
begin
   port($43):=$80;                  (*80h - команда фиксации для счетчика 3.*)
   low_byte:=port($42);
   high_byte:=port($42);
   read_8253:=low_byte+256*high_byte;
end;
var
   i:integer;
begin
   init_8253;
   for i:=1 to 100 do i:=i;

  i:=0;
   dummy:=bit_weight(x);
   port($201):=0;
   time1:=read_8253;
   repeat i:=i+1 until (port($201) and dummy=0) or (i>=5000);
   time2:=read_8253;
   interval_game_port:=time2-time1;
   if i>=5000 then interval_game_port:=0;
end;
```

В листинге 3.3 показано как выглядят эти функции, написанные на языке Turbo Pascal для Windows.

Листинг 3.3. Функции работы с джойстиком через игровой порт.

```
(*-Библиотека ресурсов № A14 (чтение регистра игрового порта).-*)
Function read_game_port(bitx:integer):integer; export;
(*Адрес игрового порта: 201h
Bitx выбирает состояние AX, AY, BX, BY, BA1, BA2, BB1 и BB2.*)
var
   input_byte:byte;
begin
   input_byte:=port($201);
   read_game_port:=round((input_byte and bit_weight(bitx))/bit_weight(bitx));
end;

(*-Библиотека ресурсов № A15 (запись в регистр игрового порта).-*)
Function write_game_port:integer; export;
(*Вводит 0 в игровой порт для запуска мультивибратора.*)
begin
   port($201):=0;
end;
(*-Библиотека ресурсов № A16 (получение длительности периода одновибратора).-*)
Function interval_game_port(x:integer):integer; export;
(*x выбирает AX(x=1), AY(x=2), BX(x=3), BY(x=4).*)
var
   time1,time2,dummy:integer;
Procedure init_8253;
(*Настройка 8253.*)
begin
(* Управляющее слово = b6h = 10110110b:
   10 = выбор счетчика 2;
   11 = сначала чтение/запись младшего байта, затем старшего;
   011 = режим 3;
   0 = шестнадцатибитовый двоичный счет.*)
   port($43):=$b6;              (*Запись управляющего слова в управляющий регистр 8253.*)
   port($42):=255;              (*Загрузка младшего байта.*)
   port($43):=255;              (*Загрузка старшего байта.*)
   011 = режим 3;
   0 = шестнадцатибитовый двоичный счет.*)
   port($43):=$b6;              (*Запись управляющего слова в управляющий регистр 8253.*)
   port($42):=255;              (*Загрузка младшего байта.*)
                                (*Загрузка старшего байта.*)
   port($61):=port($61) or 1;   (*Разрешение работы счетчика и запрет динамика.*)
end;
Function read_8253:integer;
 (*Считывание младшего и старшего байтов счетчика.*)
var
   low_byte,high_byte:byte;
begin
   port($43):=$80;              (*80h - команда фиксации для счетчика 2.*)
   low_byte:=port($42);
   high_byte:=port($42);
   read_8253:=low_byte+256*high_byte;
end;
var
   i:integer;
begin
   init_8253;
   for i:=1 to 10 do i:=i;
   i:=0;
   dummy:=bit_weight(x);
   port($201):=0;
   time1:=read_8253;
   repeat i:=i+1 until (port($201) and dummy=0) or (i>=10000);
   time2:=read_8253;
   interval_game_port:=time2-time1;
   if i>=10000 then interval_game_port:=0;
end;
```

3.3.3. Assembler

В листинге 3.4 представлен набор процедур, написанных на Ассемблере, и выполняющих основные действия по взаимодействию с джойстиком через игровой порт. Эти процедуры выполняют аналогичные действия, реализованные в функциях TP6 , представленных в листинге 3.3. Процедуры собраны в

подключаемый файл Game_pt.inc, который подключается к основной программе и транслируется, и компонуются вместе с ней и другими подключаемыми файлами. В комментариях к текстам процедур приведены пояснения к их назначению, определены используемые входные и выходные параметры, даны пояснения к используемым параметрам и к основным командам, реализующим алгоритмы процедур.

Листинг 3.4.Процедуры, используемые при работе с джойстиком (подключаемый файл Game_pt.inc).

```
DATASEG
; Значение порта 201h
Input_Byte DB   0
; Исходное значение счетчика таймера
Time1      DW   0
; Значение счетчика таймера после события
Time2      DW   0
; Номер аналогового канала:
; AX(X=1), AY(X=2), BX(X=3), BY(X=4)
X          DW   0
; Номер разряда байта:
; D0(Nbitex=0), D1(Nbitex=1),..,D7(Nbitex=7)
Nbitex     DW   0
; Значение бита: 0 - Bitex=00h
;                1 - Bitex=01h
Bitex      DB   0
; Период одновибратора
Interv_Game_Port DW  0
; Значение счетчика таймера
Time       DW   0
ENDS

CODESEG
;**********************************
;*Чтение порта 201h и определение  *
;*состояния его заданного разряда. *
;*Входные параметры:               *
;* Nbitex -номер разряда (0-7)     *
;*Возвращает:                      *
;* Input_Byte - значение порта 201h*
;* Bitex - значение заданного бита *
;*         порта 201h               *
;**********************************
PROC Read_Game_Port near
      pusha
      mov   DX,201h
      in    AL,DX
      mov   [Input_Byte],AL
      mov   BX,[Nbitex] ;Номер разряда
      clc   ;Сброс флага переноса FC
      bt    AX,BX ;Проверка заданного разряда
      jc    @@A   ;Переход по 1
      mov   [Bitex],00h
      jmp   @@A1
@@A:  mov   [Bitex],01h
@@A1: popa
      ret
ENDP Read_Game_Port

;**********************************
;*Запись в порт 201h. Инициирует *
;*одновибраторы джойстика.       *
;**********************************
PROC Write_Game_Port near
      pusha
      mov   DX,201h
      mov   AL,00h
      out   DX,AL
      popa
      ret
ENDP Write_Game_Port

;**************************************
;*Инициализация канала 2 системного     *
;*таймера 8254:                        *
;*управляющее слово для таймера:       *
;* B6h=10110110b где:                  *
;*  10 - выбор канала 2 таймера;       *
;*  11 - чтение/запись сначала младшего,*
;*       а затем старшего байта счетчика*
;*       канала;                        *
;*  011- режим 3 - генератор импульсов  *
;*       со скважностью 2;              *
;*  0  - двоичный режим счетчика        *
;**************************************
PROC Inst_8254  near
      pusha
      mov   AL,0B6h ;Установка режима работы
      out   43h,AL  ;канала 2
      mov   AL,0FFh
      out   42h,AL  ;Запись в счетчик канала 2
      out   42h,AL  ;константы FFFFh (65535)
      in    AL,61h  ;Разрешение работы счетчика
      or    AL,01h  ;канала 2, но запрет вывода
      out   61h,AL  ;на динамик.
      mov   AL,80h  ;Команда защелки содержимого
      out   43h,AL  ;счетчика канала 2
      popa
      ret
ENDP Inst_8254

;**************************************
;*Считывание младшего и старшего байта*
;*счетчика канала 2 таймера 8254      *
;*Вызывается после Inst_8254          *
;*Возвращает: Time - значение счетчика*
;* канала 2 таймера, звщелкнутое      *
;* процедурой Inst_8254.              *
;**************************************
PROC Read_8254  near
      pusha
      in    AL,42h
      xchg  AH,AL
      in    AL,42h
      xchg  AH,AL
      mov   [Time],AX
      popa
      ret
ENDP Read_8254

;*******************************************
;*Получение длительности периода одновибратора*
;*выбранного аналогового канала джойстика.    *
;*Входные параметры:                          *
;* X - выбранный канала {AX(X=1), AY(X=2),    *
;*     BX(X=3), BY(X=4)}                       *
;*Возвращает:                                 *
;* Interv_Game_Port - шестнадцатеричная       *
;*     длительность периода одновибратора в   *
;*     "тиках", каждый тик - 0,84 мкс.;       *
;* Time1 - исходное значение таймера.          *
;* Time2 - значение таймера в момент окончания*
;*   периода одновибратора.                    *
;*******************************************
PROC Interval_Game_Port  near
      pusha
      call   Inst_8254
      call   Write_Game_Port
      call   Read_8254
      mov    AX,[Time]
      mov    [Time1],AX
      mov    CX,0FFFFh
      mov    AX,[X]
      sub    AX,1
      mov    [Nbitex],AX
@@B:  call   Read_Game_Port ;Цикл ожидания
      test   [Bitex],01h    ;окончания периода
      loopnz @@B            ;одновибратора
      call   Read_8254
      mov    AX,[Time]
      mov    [Time2],AX
      mov    AX,[Time1]
      sub    AX,[Time2]
      mov    [Interv_Game_Port],AX
```

```
      cmp    CX,0000h
      jne    @@B1
      mov    [Interv_Game_Port],0
@@B1: popa
      ret
ENDP Interval_Game_Port
ENDS
```

3.3.3.1. Определение сопротивления потенциометров джойстика.

Ниже в листинге 3.5 представлена программа, использующая процедуры подключаемого файла game_pt.inc для измерения величины сопротивления резисторов, подключенных к аналоговым каналам игрового порта. Величина сопротивления подсчитывается по формуле (3.2) и выводится на экран монитора. В программе используются вспомогательные процедуры ввода-вывода, макросы и константы, сгруппированные в подключаемые файлы list1_02.inc, list1_04.inc и list1_03.inc соответственно [6]. Описание этих файлов приведено в разделе 1.2.7.1 данного пособия. В программе используются также процедуры перевода чисел из двоичной кодировки в десятичную и наоборот, сгруппированные в подключаемый файл list2_05.inc [6], описание которых приводится в следующем разделе 3.3.3.2.

Листинг 3.5. Программа, определяющая величину сопротивления, подключаемого к выбранному аналоговому каналу игрового порта (Tese_g2.asm).

```
IDEAL
P386
LOCALS
MODEL MEDIUM
; Подключение файла мнемонических обозначений
; кодов управляющих клавиш и цветовых кодов
include "list1_03.inc"
; Подключение файла макросов
include "list1_04.inc"

SEGMENT sseg para stack 'STACK'
DB 400h DUP(?)
ENDS

DATASEG
X1     DW ?
Tic    DD 0.84
K1     DD 24.2
K2     DD 11.0
K3     DD 2.0
K4     DW ?
K5     DQ ?
; Текст
Text   DB 0,18
       DB "Тестирование джойстика",0
Txt1   DB LIGHTMAGENTA,1,18
       DB "Ввод номера канала",0
       DB LIGHTGREEN,2,0,"Введите ном.канала",0
Txt2   DB LIGHTGREEN,6,0
       DB "Значение периода и сопротивления",0
       DB LIGHTGREEN,7,1
       DB " HEX       DEC       R(КОм)",0
ENDS

CODESEG
start:
PROC  Test1
      mov   AX,DGROUP
      mov   DS,AX
      mov  [CS:mainDataSeg],AX
; Текстовый режим и очистка экрана
      mov   AX,3
      int   10h
; Скрыть курсор
      mov   [ScreenString],25
      mov   [ScreenColumn],0
      call  SetCursorPosition
; Вывести текст
      mov   CX,1
      mov   SI,offset Text
@@OutText:
      call  ShowString
      loop  @@OutText
; Программа
; Вывести запрос
      MShowColorText 2,Txt1
; Задать цвет символов
      mov   [TextColorAndBackground],LIGHTGREY
; Ввести и вывести целое число
      ;Ввести целое число
      mov   [ScreenString],2
      mov   [ScreenColumn],21
      call  GetInteger
      ; Выв. hex и dec
      MShowHexDWord 4,21,[Data_Int32]
      mov   [ScreenString],5
      mov   [ScreenColumn],21
      call  Int32_to_String
      call  ShowDataString
      mov   EAX,[Data_Int32]
      mov   [X],AX    ; X-номер канала (см.
                      ; Game_pt.inc)
; Опр.периода одновибратора выбр.канала
      call  Interval_Game_Port
      mov   AX,[Interv_Game_Port]
; Печать Txt2
      MShowColorText 2,Txt2
; Позиция вывода
      mov   [ScreenString],8
      mov   [ScreenColumn],2
      mov   [X1],AX
; Печать в шестнадцатеричном виде
      xchg  AH,AL
      call  ShowHexByte
      mov   [ScreenColumn],4
      xchg  AL,AH
      call  ShowHexByte
; Печать в десятичном виде
      mov   [ScreenColumn],12
      call  ShowDecWord
; Подсчет сопротивления и его печать
      fninit      ;Сброс сопроцессора
      fild  [X1]  ;Подсчет
      fmul  [Tic] ;величины
      fdiv  [K3]  ;сопротивления
      fsub  [K1]  ;заданного канала
      fdiv  [K2]  ;джойстика (формула 3.2)
      fist  [K4]  ;Выгрузка целой части
      fst   Data_Double] ;Выгрузка вещественного
      mov   AX,[K4]      ;Печать целой
      mov   [ScreenColumn],22 ;части
      call  ShowDecWord  ;сопротивления (кОм).
      Mov   [ScreenColumn],30 ;Печать
вещественного
      Call  DoubleFloat_to_String ;значения
      call  ShowDataString   ;сопротивления
(кОм).
      mov   ah,4ch  ;Завершение работы
      int   21h     ;программы в среде DOS.
ENDP  Test1
ENDS
; Подключить процедуры вывода данных на экран
include "list1_02.inc"
; Подключение процедур работы с джойстиком
include "Game_pt.inc"
; Подключение процедур перевода чисел
include "list2_05.inc"
END start
```

3.3.3.2. Перевод чисел из десятичного кода в двоичный и наоборот

Перевод чисел из одной системы счисления в другую может выполняется при помощи математического сопроцессора очень просто — используется тот же механизм преобразования форматов данных, с помощью которого целые числа превращаются в вещественные и наоборот. Возможность применения этого способа имелась уже в самом первом сопроцессоре (i8087).

Математический сопроцессор понимает семь различных форматов представления чисел (параметры которых кратко описаны в табл. 3.2) и может выполнять любые преобразования из одного формата в другой, хотя при этом возможна потеря точности или переполнение разрядной сетки. Список команд передачи данных, которые можно использовать для преобразования чисел в различные форматы, приведен в табл. 3.3.

Таблица 3.2. Форматы данных математического сопроцессора

Тип	Размер, бит	Точность, бит	Диапазон
Короткое вещественное	32	24	От 2^{-128} до 2^{127}
Длинное вещественное	64	53	От 2^{-1022} до 2^{1023}
Расширенное вещественное	80	64	От 2^{-16382} до 2^{16383}
Целое слово	16	15	От -2^{15} до $2^{15}-1$
Короткое целое	32	31	От -2^{31} до $2^{31}-1$
Длинное целое	64	63	От -2^{63} до $2^{63}-1$
Упакованное десятичное	80	18 дес. цифр	От $(-10^{18}+1)$ до $(10^{18}-1)$

Перевод целого десятичного числа, записанного в ASCII-коде, в двоичный код выполняется в следующем порядке:

1. Преобразовать число из ASCII-кода в упакованный десятичный формат. В случае наличия в числе посторонних знаков, не являющихся цифрами, выдать сообщение об ошибке.
2. Число в упакованном формате занести в стек сопроцессора.
3. Извлечь число из стека сопроцессора и сохранить его в памяти в заданном целочисленном или вещественном формате.

Таблица 3.3. Команды передачи данных математического сопроцессора

Команда	Выполняемая операция
FBLD	Преобразовать упакованное десятичное число в расширенный вещественный формат и занести в стек сопроцессора
FBSTP	Число, находящееся в вершине стека сопроцессора, сохранить в упакованном десятичном формате и извлечь из стека
FILD	Преобразовать целое число в расширенный вещественный формат и занести в стек сопроцессора
FIST	Число, находящееся в вершине стека сопроцессора, сохранить в заданном целочисленном формате
FISTP	Число, находящееся в вершине стека сопроцессора, сохранить в заданном целочисленном формате и извлечь из стека
FLD	Преобразовать вещественное число в расширенный формат и занести в стек сопроцессора
FST	Число, находящееся в вершине стека сопроцессора, сохранить в заданном вещественном формате
FSTP	Число, находящееся в вершине стека сопроцессора, сохранить в заданном вещественном формате и извлечь из стека

Примечание

Целое число в двоичном коде можно хранить как в целочисленных, так и в вещественных форматах.

Операция перевода целого числа из двоичного кода в десятичный выполняется в обратном порядке.

1. Число, хранящееся в одном из возможных двоичных форматов (целочисленном или вещественном), занести в стек сопроцессора при помощи соответствующей команды загрузки.
2. Извлечь число из стека сопроцессора в упакованном десятичном формате.
3. Преобразовать число из упакованного формата в ASCII-код.

При переводе вещественных чисел из десятичного формата в двоичный необходима дополнительная операция — масштабирование. Масштабирование заключается в умножении преобразуемого числа на 10^{-N} где N — количество десятичных знаков после запятой. Порядок действий следующий:

1. Преобразовать число из ASCII-кода в упакованный десятичный формат как целое. Запятая при этом игнорируется, однако подсчитывается количество знаков после запятой N.
2. Число в упакованном формате занести в стек сопроцессора.
3. Умножить число на константу 10^{-N}. Для хранения констант в оперативной памяти можно использовать специальный массив вещественных чисел, индексом в котором служит N.
4. Результат извлечь из стека сопроцессора и сохранить в памяти в заданном вещественном формате.

При переводе в двоичный код десятичного числа, записанного в научном формате с порядком, равным K, умножение выполняется на 10^{K-N}, где N — количество знаков мантиссы числа после запятой.

При переводе вещественного числа из двоичного кода X в десятичный D также приходится выполнять дополнительные преобразования. После завершения перевода число должно состоять из нормализованной мантиссы M и порядка P:

$D=M\times10^{P}$

Порядок выполнения операций представлен ниже.

1. Вычислить десятичный логарифм числа X. Целая часть результата представляет собой порядок числа P.
2. Разделить число X на 10^{P}. Результат является нормализованной мантиссой числа M.
3. Извлечь из стека математического сопроцессора M и P и сохранить в памяти в упакованном формате.
4. Преобразовать M и P из упакованного формата в ASCII-код.

Вычисление логарифмов осуществляется сопроцессором с помощью универсальной команды FYL2X (выполняющей вычисление значения $\log_2X$ и умножение результата на Y). Десятичный логарифм вычисляется по формуле:

$\log_{10}X=\log_{10}2\times\log_2X$

Для обеспечения максимальной точности преобразования в наборе команд сопроцессора есть группа операций загрузки в стек констант, наиболее часто используемых при вычислениях. Значение $\log_{10}2$ загружается командой FLDLG2.

Значение 10^{P} можно вычислить многими различными способами, в том числе — с помощью сопроцессора, однако в приведенных ниже примерах используется наиболее простой способ — табличный.

Все математические функции сгруппированы в одном модуле — в листинге 3.5 [6]. Назначение подпрограмм, включенных в листинг 3.5, следующее:

- процедура Int32_to_String осуществляет перевод 32-разрядного двоичного целого числа (со знаком) в текстовую строку, представляющую собой десятичное число в ASCII-коде;
- процедура Str1ng_to_Int32 осуществляет перевод целого десятичного числа, записанного в виде текстовой строки в ASCII-коде, в двоичное 32-разрядное число;
- процедуры ShowDecByte, ShowDecWord и ShowDecDWord осуществляют преобразование в десятичный код (при помощи функции Int32_to_String) и вывод в заданную позицию экрана 8-, 16- и 32-разрядных двоичных чисел соответственно;
- процедура DoublFloat_to_Stning преобразует вещественное число из двоичного формата (с удвоенной точностью) в текстовую строку, представляющую собой десятичное число в формате с плавающей точкой (в ASCII-коде);
- процедура DoublFloat_to_ExpForm преобразует вещественное число из двоичного формата (с удвоенной точностью) в текстовую строку, представляющую собой десятичное число в научном формате (то есть с мантиссой и экспонентой);
- процедура String_to_Doub1eF1oat осуществляет перевод вещественного десятичного числа, записанного в формате с плавающей точкой в виде текстовой строки в ASCII-коде, в двоичное число удвоенной точности;
- процедура BCD_to_ASCII преобразует целое десятичное число из кода BCD в ASCII-код (процедура является вспомогательной и не должна вызываться извне, то есть из программ, не принадлежащих к данному модулю);
- процедура ShowDataString предназначена для вывода в заданную позицию экрана текстовой строки, полученной в результате преобразования двоичного числа (любого типа) в десятичное;
- процедура GetInteger обеспечивает ввод с клавиатуры десятичного целого числа со знаком;
- процедура GetFloat обеспечивает ввод вещественного десятичного числа в формате с плавающей точкой.

Листинг 3.6 Набор подпрограмм, предназначенных для перевода целых и вещественных чисел из двоичной системы счисления в десятичную и наоборот (подключаемый файл list2_05.inc).

```
DATASEG
; Количество разрядов мантиссы (1-18)
MaxPositions DW 10
; Количество знаков числа после запятой (1-17)
NumberSymbolsAD DW 5
; Машинный ноль
Data_Minimum DQ 1.E-110
; Константы (10 в степени N)
MConst DQ 1.0E1,1.0E2,1.0E3,1.0E4,1.0E5
       DQ 1.0E6,1.0E7,1.0E8,1.0E9,1.0E10
       DQ 1.0E11,1.0E12,1.0E13,1.0E14,1.0E15
DQ 1.0E16,1.0E17,1.0E18,1.0E19,1.0E20
DQ 1.0E21,1.0E22,1.0E23,1.0E24,1.0E25
DQ 1.0E26,1.0E27,1.0E28,1.0E29,1.0E30
DQ 1.0E31,1.0E32,1.0E33,1.0E34,1.0E35
DQ 1.0E36,1.0E37,1.0E38,1.0E39,1.0E40
     DQ 1.0E41,1.0E42,1.0E43,1.0E44,1.0E45
     DQ 1.0E46,1.0E47,1.0E48,1.0E49,1.0E50
     DQ 1.0E51,1.0E52,1.0E53,1.0E54,1.0E55
     DQ 1.0E56,1.0E57,1.0E58,1.0E59,1.0E60
     DQ 1.0E61,1.0E62,1.0E63,1.0E64,1.0E65
```

Листинг 3.6 (продолжение)

```
        DQ 1.0E66,1.0E67,1.0E68,1.0E69,1.0E70
        DQ 1.0E71,1.0E72,1.0E73,1.0E74,1.0E75
        DQ 1.0E76,1.0E77,1.0E78,1.0E79,1.0E80
        DQ 1.0E81,1.0E82,1.0E83,1.0E84,1.0E85
        DQ 1.0E86,1.0E87,1.0E88,1.0E89,1.0E90
        DQ 1.0E91,1.0E92,1.0E93,1.0E94,1.0E95
        DQ 1.0E96,1.0E97,1.0E98,1.0E99,1.0E100
        DQ 1.0E101,1.0E102,1.0E103,1.0E104,1.0E105
        DQ 1.0E106,1.0E107,1.0E108,1.0E109,1.0E110
        DQ 1.0E111,1.0E112,1.0E113,1.0E114,1.0E115
        DQ 1.0E116,1.0E117,1.0E118,1.0E119,1.0E120
        DQ 1.0E121,1.0E122,1.0E123,1.0E124,1.0E125
        DQ 1.0E126,1.0E127,1.0E128
; Данные передаются в математические процедуры
; через общую область памяти (главный сегмент данных)
; 32-разрядное целое число
Data_Int32    DD ?
; Число с плавающей точкой двойной точности
Data_Double   DQ ?
; Модуль числа с плавающей точкой
Data_Abs      DQ ?
; Число в BCD-формате
Data_BCD      DT ?
; Управляющее слово сопроцессора
Data_Control  DW ?
; Вспомогательный флаг
Data_Flag     DB ?
; Целая часть десятичного логарифма числа
Data_Log10    DW ?
; Знак результата (если не 0 - отрицательное число)
Data_Sign     DB ?
; Строка для хранения числа в коде ASCII
Data_String   DB 32 DUP (?)
; Структура для вывода результата
OutD_String   DB 34 DUP (?)
EVEN

CODESEG
;*************************************************
;* ПРЕОБРАЗОВАТЬ 32-РАЗРЯДНОЕ ЦЕЛОЕ ЧИСЛО В СТРОКУ *
;* Входные параметры:                            *
;* Data_Int32 - 32-разрядное число.              *
;* Выходные параметры:                           *
;* Data_String - строка-результат.               *
;*************************************************
PROC Int32_to_String near
        pushad
        push    DS
        push    ES
        mov     AX,[CS:MainDataSeg]
        mov     DS,AX
        mov     ES,AX
        cld
; Перевести число из двоичного кода в код BCD
        fninit              ;сброс сопроцессора
        ; Загрузить число в двоичном коде
        fild    [Data_Int32]
        ; Извлечь число в коде BCD
        fbstp   [Data_BCD]
; Результат записывать в строку Data_String
        mov     DI,offset Data_String
        call    BCD_to_ASCII
; Записать признак конца строки (код 0)
        xor     AL,AL
        stosb
        pop     ES
        pop     DS
        popad
        ret
ENDP Int32_to_String

;*************************************************
;*  ПРЕОБРАЗОВАТЬ СТРОКУ В 32-РАЗРЯДНОЕ ЦЕЛОЕ ЧИСЛО  *
;* Входные параметры:                            *
;* Data_String - число в коде ASCII.             *
;* Выходные параметры:                           *
;* Data_Int32 - 32-разрядное число в двоичном коде.  *
;*************************************************
PROC String_to_Int32 near
        pushad
        push    DS
        mov     AX,[CS:MainDataSeg]
        mov     DS,AX
        cld
        ; Очищаем Data_BCD
        mov     [dword ptr Data_BCD],0
        mov     [dword ptr Data_BCD+4],0
        mov     [word ptr Data_BCD+8],0
        ; Очищаем байт знака
        mov     [Data_Sign],0
        ; Заносим в SI указатель на строку
        mov     SI,offset Data_String

        ; Пропускаем пробелы перед числом
        mov     CX,64 ;защита от зацикливания
@@ShiftIgnore:
        lodsb
        cmp     AL,' '
        jne     @@ShiftIgnoreEnd
        loop    @@ShiftIgnore
        jmp short @@Error
@@ShiftIgnoreEnd:
        ; Проверяем знак числа
        cmp     AL,'-'
        jne     @@Positive
        mov     [Data_Sign],80h
        lodsb
@@Positive:
        ; 32-битное двоичное число соответствует
        ; десятичному, имеющему до 9 разрядов
        mov     CX,9
@@ASCIItoBCDConversion:
        ; Символы числа должны быть цифрами
            cmp     AL,'0'
            jb      @@Error
            cmp     AL,'9'
            ja      @@Error
            ; Пишем очередную цифру в младшую тетраду BCD
            and     AL,0Fh
        or      [byte ptr Data_BCD],AL
        ; Проверка на конец строки
        cmp     [byte ptr SI],0
        je      @@ASCIItoBCDConversionEnd
        ; Сдвигаем BCD на 4 разряда влево
        ; (сдвигаем старшие 2 байта)
        mov     AX,[word ptr Data_BCD+6]
        shld    [word ptr Data_BCD+8],AX,4
        ; (сдвигаем средние 4 байта)
        mov     EAX,[dword ptr Data_BCD]
        shld    [dword ptr Data_BCD+4],EAX,4
        ; (сдвигаем младшие 4 байта)
        shl     [dword ptr Data_BCD],4
        ; Загружаем следующий символ в AL
        lodsb
        loop    @@ASCIItoBCDConversion
        cmp     AL,0
        jne     @@Error :переполнение разрядной сетки

; ПРЕОБРАЗОВАТЬ ЧИСЛО ИЗ КОДА BCD В ЦЕЛОЕ ЧИСЛО
@@ASCIItoBCDConversionEnd:
        ; Вписать знак в старший байт
        mov     AL,[Data_Sign]
        mov     [byte ptr Data_BCD+9],AL
        ; Сбросить регистры сопроцессора
        fninit
        ; Загрузить в сопроцессор число в BCD-формате
        fbld    [Data_BCD]
```

Листинг 3.6 (продолжение)

```
        ; Выгрузить число в двоичном формате
        fistp   [Data_Int32]
        jmp short @@End

@@Error:; При любой ошибке обнулить результат
        mov     [Data_Int32],0
@@End:  pop     DS
        popad
        ret
ENDP String_to_Int32

;*****************************************************
;*       ВЫВОД БАЙТА НА ЭКРАН В ДЕСЯТИЧНОМ КОДЕ      *
;* Подпрограмма выводит содержимое регистра AL в     *
;* шестнадцатеричном коде в указанную позицию экрана.*
;* Координаты позиции передаются через глобальные    *
;* переменные ScreenString и ScreenColumn. После     *
;* выполнения операции вывода байта происходит       *
;* автоматическое приращение значений этих переменных.*
;*****************************************************
PROC ShowDecByte NEAR
        push    EAX
        and     EAX,0FFh
        call    ShowDecDWord
        pop     EAX
        ret
ENDP ShowDecByte

;*****************************************************
;*         ВЫВОД 16-РАЗРЯДНОГО СЛОВА НА ЭКРАН        *
;*                 В ДЕСЯТИЧНОМ КОДЕ                 *
;* Параметры:                                        *
;* AX - число, которое будет выведено на экран.      *
;* Номер строки передается через глобальную          *
;* переменную ScreenString, номер столбца - через    *
;* переменную ScreenColumn, цвет текста определяется *
;* глобальной переменой TextColorAndBackground.      *
;*****************************************************
PROC ShowDecWord NEAR
        push    EAX
        and     EAX,0FFFFh
        call    ShowDecDWord
        pop     EAX
        ret
ENDP ShowDecWord

;*****************************************************
;*         ВЫВОД 32-РАЗРЯДНОГО СЛОВА НА ЭКРАН        *
;*                 В ДЕСЯТИЧНОМ КОДЕ                 *
;* Параметры:                                        *
;* EAX - число, которое будет выведено на экран.     *
;* Номер строки передается через глобальную          *
;* переменную ScreenString, номер столбца - через    *
;* переменную ScreenColumn, цвет текста определяется *
;* глобальной переменой TextColorAndBackground.      *
;*****************************************************
PROC ShowDecDWord NEAR
        pushad
        push    DS
        push    ES
; Настроить регистр DS на глобальный сегмент данных
        mov     DI,[CS:MainDataSeg]
        mov     DS,DI
; Перевести число в десятичный код
        mov     [Data_Int32],EAX
        call    Int32_to_String
; Настроить регистры ES:DI для
; прямого вывода в видеопамять
        ; Загрузить адрес сегмента видеоданных в ES
        mov     AX,0B800h
        mov     ES,AX
        ; Умножить номер строки на длину
        ; строки в байтах
        mov     AX,[ScreenString]
        mov     DX,160
        mul     DX
        ; Прибавить к полученному произведению номер
        ; колонки (дважды)
        add     AX,[ScreenColumn]
        add     AX,[ScreenColumn]
        ; Переписать результат в индексный регистр
        mov     DI,AX
        cld
; Использовать цвет символов, заданный по умолчанию
        mov     AH,[TextColorAndBackground]
; Вывести число на экран
        mov     CX,10
        mov     SI,offset Data_String
@@NextChar:
        lodsb               ;загрузить цифру в AL
        and     AL,AL       ;проверка на 0 (конец строки)
        jz      @@EndOfString
        stosw               ;вывести цифру на экран
        loop    @@NextChar
@@EndOfString:
        pop     ES
        pop     DS
        popad
        ret
ENDP ShowDecDWord

;*****************************************************
;*  ПРЕОБРАЗОВАНИЕ ЧИСЛА С ПЛАВАЮЩЕЙ ТОЧКОЙ В СТРОКУ  *
;* Число имеет формат с удвоенной точностью, результат *
;* выдается в десятичном коде, в "бытовом" формате с  *
;* фиксированным количеством знаков после запятой.    *
;* Входные параметры:                                  *
;* Data_Double - преобразуемое число;                  *
;* NumberSymbolsAD - количество знаков после           *
;*                   запятой (0-17).                   *
;* Выходные параметры:                                 *
;* Data_String - строка-результат.                     *
;*****************************************************
PROC DoubleFloat_to_String near
        pushad
        push    DS
        push    ES
        mov     AX,[CS:MainDataSeg]
        mov     DS,AX
        mov     ES,AX
        ; Результат записывать в строку Data_String
        mov     DI,offset Data_String

        ; Сдвигаем число влево на NumberSymbolsAD
        ; десятичных разрядов
        fninit                  ;сброс сопроцессора
        fld     [Data_Double]   ;загрузить число
        mov     BX,[NumberSymbolsAD]
        cmp     BX,0
        je      @@NoShifts      ;нет цифр после запятой
        jl      @@Error         ;ошибка
        dec     BX
        shl     BX,3            ;умножаем на 8
        add     BX,offset MConst
        fmul    [qword ptr BX] ;умножить на константу
@@NoShifts:
        ; Извлечь число в коде BCD
        fbstp   [Data_BCD]
; Проверить результат на переполнение
        mov     AX,[offset Data_BCD + 8]
        cmp     AX,0FFFFh  ;"десятичное" переполнение?
        je      @@Overflow
; Выделить знак числа и записать его в ASCII-коде
        mov     AL,[offset Data_BCD + 9]
        and     AL,AL
        jz      @@NoSign
        mov     AL,'-'
        stosb
```

Листинг 3.6 (продолжение)

```
@@NoSign:
; Распаковать число в код ASCII
        mov     BX,8      ;смещение последней пары цифр
        mov     CX,9      ;счетчик пар цифр
        ; Определить позицию десятичной точки в числе
        mov     DX,18
        sub     DX,[NumberSymbolsAD]
        js      @@Error   ;ошибка, если отрицательная
        jz      @@Error   ;или нулевая позиция
@@NextPair:
        ; Загрузить очередную пару разрядов
        mov     AL,[BX + offset Data_BCD]
        mov     AH,AL
        ; Выделить, перевести в ASCII и
        ; сохранить старшую тетраду
        shr     AL,4
        add     AL,'0'
        stosb
        dec     DX
        jnz     @@N0
        mov     AL,'.'
        stosb
@@N0:   ; Выделить, перевести в ASCII и
        ; сохранить младшую тетраду
        mov     AL,AH
        and     AL,0Fh
        add     AL,'0'
        stosb
        dec     DX
        jnz     @@N1
        mov     AL,'.'
        stosb
@@N1:   dec     BX
        loop    @@NextPair
        mov     AL,0
        stosb

; Убрать незначащие нули слева
        mov     DI,offset Data_String
        mov     SI,offset Data_String
        ; Пропустить знак числа, если он есть
        cmp     [byte ptr SI],'-'
        jne     @@N2
        inc     SI
        inc     DI
@@N2:   ; Загрузить в счетчик цикла количество разрядов
        ; числа плюс 1 (байт десятичной точки)
        mov     CX,18+1+1
        ; Пропустить незначащие нули
@@N3:   cmp     [byte ptr SI],'0'
        jne     @@N4
        cmp     [byte ptr SI+1],'.'
        je      @@N4
        inc     SI
        loop    @@N3
        ; Ошибка - нет значащих цифр
        jmp short @@Error
; Скопировать значащую часть числа в начало строки
@@N4:   rep movsb
        jmp short @@End

; Ошибка
@@Error:
        mov     AL,'E'
        stosb
        mov     AL,'R'
        stosb
        mov     AL,'R'
        stosb
        xor     AL,AL
        stosb
        jmp short @@End
; Переполнение разрядной сетки
@@Overflow:
        mov     AL,'#'
        stosb
        xor     AL,AL
        stosb
; Конец процедуры
@@End:  pop     ES
        pop     DS
        popad
        ret
ENDP DoubleFloat_to_String

;*****************************************************
;*  ПРЕОБРАЗОВАТЬ ЧИСЛО С ПЛАВАЮЩЕЙ ТОЧКОЙ В СТРОКУ    *
;* Число имеет формат с удвоенной точностью, результат *
;* выдается в десятичном коде в научном формате, т.е.  *
;* с нормализованной мантиссой и порядком.             *
;* Входные параметры:                                  *
;* Data_Double - преобразуемое число;                  *
;* MaxPositions - количество разрядов мантиссы (1-18). *
;* Выходные параметры:                                 *
;* Data_String - строка-результат.                     *
;*****************************************************
PROC DoubleFloat_to_ExpForm near
        pushad
        push    DS
        push    ES
        mov     AX,[CS:MainDataSeg]
        mov     DS,AX
        mov     ES,AX
        ; Результат записывать в строку Data_String
        mov     DI,offset Data_String
; Определить знак числа
        fninit                  ;сброс сопроцессора
        fld     [Data_Double]   ;загрузить число
        fxam                    ;протестировать число
        fstsw   AX
        test    ah,10b          ;проверить знак
        jz      @@z0
        ; Число отрицательное
        mov     AL,'-'          ;записать знак "минус"
        stosb
; Запомнить модуль числа
        fabs
@@z0:   fst     [Data_Abs]
; Произвести проверку на "машинный ноль"
        fcom    [Data_Minimum]
        fstsw   AX
        and     AH,1000001b
        jnz     @@Zero

; Устанавить режим округления "вниз" (01)
        fstcw   [Data_Control] ;считать слово состояния
        and     [Data_Control],0F7FFh
        or      [Data_Control],0400h
        fldcw   [Data_Control]

; Вычислить десятичный логарифм числа
        fldlg2
        fxch    ST(1)
        fyl2x
        ; Записать десятичный порядок
        fistp   [Data_Log10]

; Устанавить режим округления "к ближайшему" (00)
        fstcw   [Data_Control] ;считать слово состояния
        and     [Data_Control],0F3FFh
        fldcw   [Data_Control]

; Нормализовать десятичную мантиссу
        fninit                  ;сброс сопроцессора
        fld     [Data_Abs]      ;загрузить модуль числа
        mov     BX,[MaxPositions]
        sub     BX,[Data_Log10]
```

Листинг 3.6 (продолжение)

```
        cmp     BX,0
        je      @@z2            ;сдвиги не нужны
        jl      @@z1            ;нужен сдвиг вправо
        ; Сдвинуть число влево
        ; на [BX] десятичных разрядов
        dec     BX
        cmp     BX,127
        ja      @@Zero          ;машинный ноль
        shl     BX,3            ;умножаем на 8
        add     BX,offset MConst
        fmul    [qword ptr BX] ;умножить на константу
        jmp short @@z2
        ; Сдвинуть число вправо
        ; на [BX] десятичных разрядов
@@z1:   neg     BX
        dec     BX
        cmp     BX,127
        ja      @@Overflow      ;переполнение
        shl     BX,3            ;умножаем на 8
        add     BX,offset MConst
        fdiv    [qword ptr BX] ;разделить на константу
@@z2:   fbstp   [Data_BCD]      ;извлечь число в коде BCD

; Записать точку перед мантиссой
        mov     SI,DI  ;запомнить позицию точки
        mov     AL,'.'
        stosb
; Преобразовать мантиссу в код ASCII
        call    BCD_to_ASCII
; Поменять местами точку и первую цифру мантиссы
        mov     AX,[SI]
        xchg    AL,AH
        mov     [SI],AX

; Перевести порядок результата в код BCD
        ; Проверить значение порядка
        cmp     [Data_Log10],0
        je      @@End
        ; Записать признак порядка
        mov     AL,'e'
        stosb
        fninit                  ;сброс сопроцессора
        ; Загрузить десятичный порядок в двоичном коде
        fild    [Data_Log10]
        ; Извлечь порядок в коде BCD
        fbstp   [Data_BCD]
        ; Результат записать в строку Data_String
        call    BCD_to_ASCII
        jmp short @@End

@@Zero: ; Машинный ноль
        mov     AL,'0'
        stosb
        jmp short @@End
@@Overflow:
        ; Переполнение разрядной сетки
        mov     AL,'#'
        stosb
@@End:  ; Записать признак конца строки (код 0)
        xor     AL,AL
        stosb

        pop     ES
        pop     DS
        popad
        ret
ENDP DoubleFloat_to_ExpForm

;*************************************************
;* ПРЕОБРАЗОВАТЬ СТРОКУ В ЧИСЛО С ПЛАВАЮЩЕЙ ТОЧКОЙ  *
;*      (число имеет обычный, "бытовой" формат)     *
;* Входные параметры:                              *
;* Data_String - число в коде ASCII.               *
;* Выходные параметры:                             *
;* Data_Double - число в двоичном коде.            *
;*************************************************
PROC String_to_DoubleFloat near
        pushad
        push    DS
        mov     AX,[CS:MainDataSeg]
        mov     DS,AX
        cld
        ; Очищаем Data_BCD
        mov     [dword ptr Data_BCD],0
        mov     [dword ptr Data_BCD+4],0
        mov     [word ptr Data_BCD+8],0
        ; Очищаем байт знака
        mov     [Data_Sign],0
        ; Заносим в SI указатель на строку
        mov     SI,offset Data_String
        ; Пропускаем пробелы перед числом
        mov     CX,64 ;защита от зацикливания
@@ShiftIgnore:
        lodsb
        cmp     AL,' '
        jne     @@ShiftIgnoreEnd
        loop    @@ShiftIgnore
        jmp     @@Error
@@ShiftIgnoreEnd:
        ; Проверяем знак числа
        cmp     AL,'-'
        jne     @@Positive
        mov     [Data_Sign],80h
        lodsb
@@Positive:
        mov     [Data_Flag],0 ;признак наличия точки
        mov     DX,0          ;позиция точки
        mov     CX,18         ;макс. число разрядов
@@ASCIItoBCDConversion:
        cmp     AL,'.'        ;точка?
        jne     @@NotDot
        cmp     [Data_Flag],0 ;точка не встречалась?
        jne     @@Error
        mov     [Data_Flag],1
        lodsb
        cmp     AL,0          ;конец строки?
        jne     @@NotDot
        jmp     @@ASCIItoBCDConversionEnd
@@NotDot:
        ; Увеличить на 1 значение позиции точки,
        ; если она еще не встречалась
        cmp     [Data_Flag],0
        jnz     @@Figures
        inc     DX
@@Figures:
        ; Символы числа должны быть цифрами
        cmp     AL,'0'
        jb      @@Error
        cmp     AL,'9'
        ja      @@Error
        ; Пишем очередную цифру в младшую тетраду BCD
        and     AL,0Fh
        or      [byte ptr Data_BCD],AL
        ; Проверка на конец строки
        cmp     [byte ptr SI],0
        je      @@ASCIItoBCDConversionEnd
        ; Сдвигаем BCD на 4 разряда влево
        ; (сдвигаем старшие 2 байта)
        mov     AX,[word ptr Data_BCD+6]
        shld    [word ptr Data_BCD+8],AX,4
        ; (сдвигаем средние 4 байта)
        mov     EAX,[dword ptr Data_BCD]
        shld    [dword ptr Data_BCD+4],EAX,4
        ; (сдвигаем младшие 4 байта)
        shl     [dword ptr Data_BCD],4
        ; Загружаем следующий символ в AL
        lodsb
        loop    @@ASCIItoBCDConversion
```

Листинг 3.6 (продолжение)

```
        ; Если 19-й символ не 0 и не точка,
        ; то ошибка переполнения
        cmp     AL,'.'
        jne     @@NotDot2
        inc     CX
        lodsb
@@NotDot2:
        cmp     AL,0
        jne     @@Error ;переполнение разрядной сетки

; ПРЕОБРАЗОВАТЬ ЧИСЛО ИЗ КОДА BCD В ВЕЩЕСТВЕННОЕ ЧИСЛО
@@ASCIItoBCDConversionEnd:
        ; Вписать знак в старший байт
        mov     AL,[Data_Sign]
        mov     [byte ptr Data_BCD+9],AL
        ; Сбросить регистры сопроцессора
        fninit
        ; Загрузить в сопроцессор число в BCD-формате
        fbld    [Data_BCD]
        ; Вычислить номер делителя
        mov     BX,18+1
        sub     BX,CX
        sub     BX,DX
        cmp     BX,0
        je      @@NoDiv
        dec     BX
        shl     BX,3            ;умножаем на 8
        add     BX,offset MConst
        fdiv    [qword ptr BX] ;разделить на константу
@@NoDiv:; Выгрузить число в двоичном формате
        fstp    [Data_Double]
        jmp short @@End

@@Error:; При любой ошибке обнулить результат
        fldz    ;занести ноль в стек сопроцессора
        fstp    [Data_Double]
@@End:  pop     DS
        popad
        ret
ENDP String_to_DoubleFloat

;*************************************************
;*   ПРЕОБРАЗОВАТЬ ЧИСЛО ИЗ КОДА BCD В КОД ASCII    *
;* (вспомогательная функция, регистры не сохраняет) *
;* Входные параметры:                              *
;* Data_BCD - число в BCD-формате.                 *
;* Регистр DI - указатель на строку результата.    *
;*************************************************
PROC BCD_to_ASCII near
; Выделить знак числа и записать его в ASCII-коде
        mov     AL,[offset Data_BCD + 9]
        and     AL,AL
        jz      @@n0
        mov     AL,'-'
        stosb
; Пропустить незначащие (нулевые) разряды слева
@@n0:   mov     BX,8
        mov     CX,9
@@n1:   ;проверяем на 0 очередную пару разрядов
        cmp     [byte ptr BX+offset Data_BCD],0
        jne     @@n2
        dec     BX
        loop    @@n1
; Если значение числа равно нулю, записать символ
; нуля в строку результата и выйти из программы
        mov     AL,'0'
        stosb
        jmp short @@End
; Пропустить незначащий ноль в старшей
; тетраде (если он есть)
@@n2:   ; Загрузить первую значащую пару разрядов
        mov     AL,[BX + offset Data_BCD]
        mov     AH,AL
        ; Выделить, перевести в ASCII и
        ; сохранить старшую тетраду
        shr     AL,4
        cmp     AL,0
        ; Если 0 - пропустить старшую тетраду
        je      @@n3
        add     AL,'0'
        stosb
        ; Выделить, перевести в ASCII и
        ; сохранить младшую тетраду
@@n3:   mov     AL,AH
        and     AL,0Fh
        add     AL,'0'
        stosb
        dec     BX
        dec     CX
        jz      @@End ;выход, если это последний разряд

; Распаковать остальные разряды числа (если они есть)
@@n4:   ; Загрузить очередную пару разрядов
        mov     AL,[BX + offset Data_BCD]
        mov     AH,AL
        ; Выделить, перевести в ASCII и
        ; сохранить старшую тетраду
        shr     AL,4
        add     AL,'0'
        stosb
        ; Выделить, перевести в ASCII и
        ; сохранить младшую тетраду
        mov     AL,AH
        and     AL,0Fh
        add     AL,'0'
        stosb
        dec     BX
        loop    @@n4
@@End:
        ret
ENDP BCD_to_ASCII

;*************************************************
;*     ОТОБРАЗИТЬ ЧИСЛО В КОДЕ ASCII НА ЭКРАН     *
;* Подпрограмма отображает Data_String на экран.  *
;* Координаты позиции передаются через глобальные *
;* переменные ScreenString и ScreenColumn. После  *
;* выполнения операции происходит автоматическое  *
;* приращение значений этих переменных.           *
;*************************************************
PROC ShowDataString near
        pusha
        push    ES
        mov     AX,[CS:MainDataSeg]
        mov     ES,AX
        cld
; Занести координаты сторки
        mov     DI,offset OutD_String
        mov     AX,[ScreenString]
        stosb
        mov     AX,[ScreenColumn]
        stosb
; Копировать Data_String в OutD_String
        mov     SI,offset Data_String
        mov     CX,31
        rep     movsb
; Поставить символ-ограничитель
        mov     AL,0
        stosb
        mov     SI,offset OutD_String
        call    ShowString
        pop     ES
        popa
        ret
ENDP ShowDataString
```

Листинг 3.6 (продолжение)

```
;************************************************
;*         ВВЕСТИ ЦЕЛОЕ ЧИСЛО С КЛАВИАТУРЫ      *
;* Подпрограмма обеспечивает ввод 9-разрядного  *
;* десятичного целого числа со знаком.          *
;* Координаты поля ввода задаются через переменные *
;* ScreenString и ScreenColumn. Цвет определяется  *
;* переменной TextColorAndBackground.           *
;************************************************
PROC GetInteger near
        pushad
; Инициализировать переменные
@@ClearField:
        ; Обнулить счетчик цифр
        mov     BX,0
        ; Установить координаты поля ввода
        mov     DI,offset OutD_String
        mov     AX,[ScreenString]
        mov     [DI],AL
        inc     DI
        mov     AX,[ScreenColumn]
        mov     [DI],AL
        inc     DI
        ; Очистить строку (заполнить пробелами)
        mov     [dword ptr DI],20202020h
        mov     [dword ptr DI+4],20202020h
        mov     [word ptr DI+8],2020h
        mov     [byte ptr DI+10],0 ;конец строки

; Цикл ввода
@@GetNextChar:
        ; Отобразить число на экране
        mov     SI,offset OutD_String
        call    ShowString
        ; Установить курсор в позицию ввода
        push    [ScreenColumn]
        add     [ScreenColumn],BX
        call    SetCursorPosition
        pop     [ScreenColumn]
        ; Принять байт с клавиатуры
        call    GetChar
        ; Цифра или команда?
        cmp     AL,0
        jz      @@Command  ;если ноль - введена команда
        ; Проверка на специальные символы
        cmp     AL,'-'
        je      @@M
        ; Проверка на диапазон '0'-'9'
        cmp     AL,'0'
        jb      @@Error
        cmp     AL,'9'
        ja      @@Error
        ; Проверяем количество символов (не более 10)
@@M:    cmp     BX,10
        jae     @@Error
        inc     BX
        ; Записываем символ в число
        mov     [DS:DI],AL
        inc     DI
        jmp short @@GetNextChar
; Проанализировать код команды
@@Command:
        cmp     AH,B_Esc        ;очистить поле
        je      @@ClearField
@@TestRubout:
        cmp     AH,B_RUBOUT     ;"Забой"
        jne     @@TestEnter
        cmp     BX,0
        je      @@Error
        ; Передвинуть признак конца строки
        ; на разряд влево
        dec     DI
        dec     BX
        mov     [byte ptr DS:DI],' '
        jmp short @@GetNextChar
@@TestEnter:
        cmp     AH,B_Enter      ;завершение ввода числа
        jne     @@Error
        ; Перевести число в двоичный код
        mov     SI,offset Data_String
        mov     DI,offset OutD_String
        add     DI,2
        mov     CX,10
@@i1:   mov     AL,[DI]
        cmp     AL,' '          ;конец числа?
        jbe     @@EndOfString
        mov     [SI],AL
        inc     SI
        inc     DI
        loop    @@i1
@@EndOfString:
        mov     [byte ptr SI],0 ;конец строки
        call    String_to_Int32
        jmp short @@End

; Ошибка при вводе
@@Error:
        call    Beep
        jmp     @@GetNextChar

@@End:  popad
        ret
ENDP GetInteger

;************************************************
;*      ВВЕСТИ ВЕЩЕСТВЕННОЕ ЧИСЛО С КЛАВИАТУРЫ  *
;* Подпрограмма обеспечивает ввод вещественного *
;* числа в обычном формате (до 18 цифр).        *
;* Координаты поля ввода задаются через переменные *
;* ScreenString и ScreenColumn. Цвет определяется  *
;* переменной TextColorAndBackground.           *
;************************************************
PROC GetFloat near
        pushad
; Инициализировать переменные
@@ClearField:
        ; Обнулить счетчик цифр
        mov     BX,0
        ; Установить координаты поля ввода
        mov     DI,offset OutD_String
        mov     AX,[ScreenString]
        mov     [DI],AL
        inc     DI
        mov     AX,[ScreenColumn]
        mov     [DI],AL
        inc     DI
        ; Очистить строку (заполнить пробелами)
        mov     [dword ptr DI],20202020h
        mov     [dword ptr DI+4],20202020h
        mov     [dword ptr DI+8],20202020h
        mov     [dword ptr DI+12],20202020h
        mov     [dword ptr DI+16],20202020h
        mov     [byte ptr DI+20],0 ;конец строки
; Цикл ввода
@@GetNextChar:
        ; Отобразить число на экране
        mov     SI,offset OutD_String
        call    ShowString
        ; Установить курсор в позицию ввода
        push    [ScreenColumn]
        add     [ScreenColumn],BX
        call    SetCursorPosition
        pop     [ScreenColumn]
        ; Принять байт с клавиатуры
```

Листинг 3.6 (продолжение)

```
        call    GetChar
        ; Цифра или команда?
        cmp     AL,0
        jz      @@Command  ;если ноль - введена команда
        ; Проверка на специальные символы
        cmp     AL,'-'
        je      @@M
        cmp     AL,'.'
        je      @@M
        ; Проверка на диапазон '0'-'9'
        cmp     AL,'0'
        jb      @@Error
        cmp     AL,'9'
        ja      @@Error
        ; Проверяем количество символов (не более 20)
@@M:    cmp     BX,20
        jae     @@Error
        inc     BX
        ; Записываем символ в число
        mov     [DS:DI],AL
        inc     DI
        jmp short @@GetNextChar
; Проанализировать код команды
@@Command:
        cmp     AH,B_Esc        ;очистить поле
        je      @@ClearField
@@TestRubout:
        cmp     AH,B_RUBOUT     ;"Забой"
        jne     @@TestEnter
        cmp     BX,0
        je      @@Error
        ; Передвинуть признак конца строки
        ; на разряд влево
        dec     DI
        dec     BX
        mov     [byte ptr DS:DI],' '
        jmp short @@GetNextChar
@@TestEnter:
        cmp     AH,B_Enter      ;завершение ввода числа
        jne     @@Error
        ; Перевести число в двоичный код
        mov     SI,offset Data_String
        mov     DI,offset OutD_String
        add     DI,2
        mov     CX,20
@@i1:   mov     AL,[DI]
        cmp     AL,' '          ;конец числа?
        jbe     @@EndOfString
        mov     [SI],AL
        inc     SI
        inc     DI
        loop    @@i1
@@EndOfString:
        mov     [byte ptr SI],0 ;конец строки
        call    String_to_DoubleFloat
        jmp short @@End

; Ошибка при вводе
@@Error:
        call    Beep
        jmp     @@GetNextChar

@@End:  popad
        ret
ENDP GetFloat
ENDS
```

ПРИМЕЧАНИЕ. Обратите внимание: математический сопроцессор автоматизирует перевод числа из упакованного двоично-десятичного кода BCD в двоичный и наоборот, но при этом программист все еще должен самостоятельно выполнять преобразование из ASCII-кода в код BCD, а также контролировать корректность поступающей информации. Вводимое число не должно содержать посторонних символов и его значение не должно выходить за пределы некоторого заданного диапазона; для выводимых чисел также необходимо выполнять проверку на диапазон.

3.3.4. Borland C/C++.

В листинге 3.7 представлена программа Djoistv.cpp, написанная на языке C/C++ и реализующая измерение величины сопротивления резисторов, подключенных к аналоговым каналам игрового порта, т. е. решает задачу, аналогичную решаемой программой Tese_g2.asm (листинг 3.5). В программу включены написанные на C++ процедуры выполняющие основные действия по взаимодействию с джойстиком через игровой порт, аналогичные процедурам, представленным в листингах 3.3 и 3.4. Следует отметить, что в программе Djoistv.cpp добавлена процедура Rinst_8254 восстановления системного значения счетчика второго канала таймера, которое изначально устанавливает BIOS на стадии выполнения POST (Power On Self (Start) Test).

Листинг 3.7. Программа, определяющая величину сопротивления, подключенного к выбранному аналоговому каналу игрового порта (Djoistv.cpp).

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

//Процедура определения состояния одновибратора
//выбранного канала (0-3).
//Nb - номер канала,
//Inp_B - состояние порта 201h,
//Bit - состояние выбранного разряда порта 201h.
void Re_G_P(int Nb,unsigned char *Inp_B,int *Bit)
{       int n,n1=0x0001;
        *Inp_B=inp(0x201);
```

```
      n=n1<<Nb;
      if ((*Inp_B & n)==0)
            *Bit=0;
      else
            *Bit=1;
}

//Процедура запуска одновибраторов каналов порта джойстика.
void Wr_G_P()
{     outp(0x201,0x00);
}

//Процедура инициализации 2-ого канала таймера 8254
//и разрешения работы его счетчика.
void Inst_8254()
{   char cr;
      outp(0x43,0xb6);
      outp(0x42,0xff);
      outp(0x42,0xff);
      cr=inp(0x61);
      outp(0x61,cr|0x01);
}

//Процедура восстановления стандартного значения коэффициента
//деления счетчика канала 2 таймера и остановки его работы.
void Rinst_8254()
{     char cr;
      outp(0x42,0xe8);
      outp(0x42,0x03);
      cr=inp(0x61);
      outp(0x61,cr&0xfe);
}

//Процедура чтения содержимого счетчика канала 2 таймера.
long int  Read_8254()
{   long int T, T0;
      outp(0x43,0x80);
      T=inp(0x42);
      T0=inp(0x42); T0=T0<<8;
   return (T+T0);
}

//Программа подсчета значения сопротивления, подклю-
//ченного к выбранному каналу порта джойстика.
int main (void)
{     int Nb=0, Bit=0, C=60000;
      float R;
      long int T1, T2;
      unsigned char Inp_B;

      cout << "Input N of bit gameport (0 - 3):";
      cin >> Nb;

      Inst_8254();      //холостая операция
      Inp_B=inp(0x201); //холостая операция
      Wr_G_P();
      Inst_8254();
      T1=Read_8254();
      do
      {     Re_G_P(Nb,&Inp_B,&Bit); //Ожидание срабатывания
            --C;                    //
            if(C==0)                //одновибратора.
            Bit=0;
      }
      while(Bit==1);
      T2=Read_8254();
      R=(.84*(T1-T2)*0.5-24.2)/11.; //Вычисление сопротивления.
      Rinst_8254();

      cout<<"R (kOm) = "<<R<<"  "<<"Bit = "<<Bit<<"  "<<"T1 = "<<T1<<"  "<<"T2 =
"<<T2<<endl;
```

```
        getch();
        return 0;
}
```

Следует отметить, что значения сопротивлений определенных с помощью программ Tese_g2.asm и Djoistv.cpp могут отличаться друг от друга более чем на 30%, причем программа Djoistv.cpp всегда дает меньшее значение, чем программа Tese_g2.asm. Проверка проводилась на ПК с процессорами пятого поколения в среде DOS и в среде Windows 95/98 в "сеансе MS-DOS".

В данном разделе использовались материалы из [4 - 7, 15].

Контрольные вопросы

1. Для чего используется игровой порт ПК?
2. Как подключается джойстик к игровому порту?
3. К каким игровым портам могут подключаться NIDI-устройства?
4. Охарактеризуйте электрическую схему игрового порта (рис. 3.2).
5. Охарактеризуйте логическую структуру игрового порта (рис. 3.3).
6. Объясните функции битов порта 201h.
7. Дайте описание метода измерения сопротивления датчиков джойстиков, используемого контроллером игрового порта.
8. Охарактеризуйте средства взаимодействия с джойстиком, используемые в языке программирования Qbasic.
9. Объясните работу программы, представленной в листинге 3.1.
10. Опишите методику определения сопротивления датчиков джойстика, использующую каналы системного таймера i8254.
11. Объясните работу программы, представленной в листинге 3.2.
12. Объясните работу программы, представленной в листинге 3.3.
13. Объясните работу процедур, написанных на Ассемблере и представленных в листинге 3.4.
14. Объясните работу программы, представленной в листинге 3.5.
15. Объясните принципы перевода чисел из десятичного кода в двоичный и наоборот с помощью математического сопроцессора.
16. Какие форматы данных использует математический сопроцессор?
17. Охарактеризуйте команды передачи данных математического сопроцессора.
18. Охарактеризуйте процедуры, представленные в листинге 3.6.
19. Объясните программу, представленную на языке C/C^{++} и представленную в листинге 3.7.

ПРАКТИКУМ

Практикум включает в себя семь лабораторных работ, посвященных изучению принципов организации подсистем электронной клавиатуры и манипуляторов типа мышь. В описаниях к лабораторным работам рассмотрены: основные принципы функционирования периферийных устройств типа клавиатуры, PS/2 Mouse и Serial Mouse; их электрические схемы; аппаратные интерфейсы их подключения к системному блоку ПК; принципы функционирования адаптера клавиатуры и PS\2 Mouse и контроллера COM-порта. Также рассмотрены принципы программного взаимодействия системы с компонентами подсистем клавиатуры и манипуляторов мышь на уровне базовых средств СВВ (на уровне их регистровой архитектуры, на уровне команд логического интерфейса (канального уровня) и на уровне функций BIOS).

Для изучения принципов функционирования периферийных устройств подсистемы на уровне их электрических схем в лабораторных работах используются клавиатуры и манипуляторы мышь со встроенными зондовыми устройствами доступа к контрольным точкам их электрических схем и к линиям интерфейсов их подключения к системному блоку ПК. При выполнение лабораторных работ электрические сигналы в контрольных точках и на линиях интерфейсов регистрируются с помощью измерительных устройств (например, осциллографа).

Фрагменты программ, реализующих алгоритмы и протоколы взаимодействия с устройствами подсистемы клавиатуры и манипуляторов мышь, пишутся в мнемокодах Ассемблера, а отлаживаются и выполняются в среде полноэкранного отладчика реального режима типа AFD (SDT) () или отладчика DEBUG (). Эти отладчики имеют режимы ассемблера и дизассемблера. Они также имеют набор команд, позволяющих набирать, отлаживать и выполнять программы прямо в своей среде. Как правило, они работают в среде DOS, однако с некоторыми ограничениями они могут функционировать и в окне сеанса MS-DOS операционных систем типа Windows 95/98.

1. Общие указания по выполнению и оформлению лабораторных работ

Выполнение каждой лабораторной работы состоит из трех этапов:

1. подготовка к лабораторной работе;
2. проведение работ по непосредственному практическому изучению двунаправленного параллельного порта ПК, принципов его работы и программирования в изучаемом режиме;
3. оформление отчета и защита лабораторной работы.

Подготовка к лабораторной работе заключается в изучении описания лабораторной работы, описания соответствующего режима работы контроллера параллельного порта, отладчика AFD, ознакомление с примерами программ и временными диаграммами, поясняющими правила обмена информацией через параллельный порт в соответствующем режиме его работы. Подготовка к работе проводится студентом самостоятельно в рамках внеаудиторной работы.

Проведение работ по непосредственному изучению принципов работы двунаправленного параллельного порта разбивается на два этапа.

На первом этапе выполняются все пункты задания, сформулированные в разделе описания "Порядок выполнения лабораторной работы". Они включают в себя: набор, отладку и выполнение фрагментов программ, приведенных в описании, в среде отладчика AFD. Во время отладки и выполнения фрагментов программ контролируется состояние линий интерфейса и контроллера порта как по изменению содержимого окон отладчика AFD, так и с помощью осциллографа или других измерительных средств.

На втором этапе выполняются индивидуальные задания, сформулированные преподавателем после проверки результатов первого этапа работы.

Оформление отчета по лабораторной работе осуществляется во внеаудиторное время.

Отчет по лабораторной работе должен содержать следующий материал:

- постановку задачи выполняемой лабораторной работы;
- краткое описание работы параллельного порта в соответствующем режиме;
- тексты программ на ассемблере (мнемокоды команд процессора 8086) с подробными комментариями;
- осциллограммы, таблицы и т.п.;
- выводы по проделанной работе.

Отчет должен быть оформлен грамотно и аккуратно.

Сдача (защита) лабораторной работы производится во время выполнения следующей лабораторной работы или в дополнительное время по согласованию с преподавателем.

2. Лабораторная работа 1 (описание) "Адаптер клавиатуры и PS/2 Mouse"

1. Цель работы

Целью лабораторной работы является ознакомление студентов с принципами работы адаптера, изучение набора команд, выполняемых адаптером и принципов программного управления его режимами работы.

2. Технические и программные средства, используемые в работе

В работе используется персональный компьютер типа Pentium, имеющий интерфейс подключения манипулятора PS/2 Mouse и работающего под управлением операционной системы Windows 95/98/Melenium. Операционная система загружается в режиме DOS (Command prompt only) с оболочкой типа Norton Commander. Программное взаимодействие с адаптером осуществляется в среде отладчика типа AFD. Используется также электронный осциллограф с полосой пропускания не менее 0 - 10 МГц и кабель AMAM стандарта IEEE 1284 с разъемами A (вилка) на обоих концах, подключаемый к разъему LPT-порта компьютера с одной стороны, а с другой - к щупу осциллографа.

3. Краткое описание адаптера клавиатуры и мыши PS/2

Адаптер клавиатуры и мыши PS/2 используется для подключения к системе электронной клавиатуры и манипулятора PS/2 Mouse через последовательные двунаправленные интерфейсы. Помимо этого адаптер используется для выполнения ряда системных функций (программный сброс системы, управление адресной линией A20 и др.).

Адаптер выполнен на базе однокристального микрокомпьютера (микроконтроллера) типа i8042. В микроконтроллере постоянно исполняется микропрограмма, реагирующая на сигналы интерфейсных линий клавиатуры и PS/2 Mouse, а также на команды и данные, поступающие от центрального процессора. Эта микропрограмма хранится в ПЗУ микроконтроллера. Адаптер может работать в одном из двух режимов: в режиме AT или режиме PS/2. При инициализации (по аппаратному сбросу) адаптер устанавливается в режим PS/2 или AT, в зависимости от соответствующей настройки BIOS Setup. В режиме AT адаптер не выполняет функции, связанные с обслуживанием интерфейса мыши и игнорирует все команды, относящиеся к мыши. В режиме PS/2 адаптером поддерживается работа, как с клавиатурой, так и с манипулятором мышь. В этом режиме модифицируются некоторые разряды регистра состояния, регистра командного байта, а также изменяется назначение некоторых выводов портов адаптера (см. ниже).

Порты микроконтроллера

В микроконтроллере имеется 8-разрядный входной и выходной порты и дополнительный 2-разрядный входной порт. Назначение выводов портов имеет свои особенности в режимах AT и PS/2.

Порт ввода **P1** доступен для чтения по команде C0h; в режиме PS/2 имеет следующее назначение разрядов:

- бит 7 - ключ блокировки клавиатуры *(KeyLock):* 0 - клавиатура заблокирована;
- бит 6 - перемычка установки видеорежима: 0 - цветной (Color), 1 - монохромный (Mono);
- бит 5 - системная перемычка: 0 - замкнута;
- бит 4 - перемычка задания объема ОЗУ: 0 - 256 Кбайт, 1 - 512 Кбайт и более;
- биты 3, 2 - не используются;
- бит 1 - линия MS-Data, в режиме AT не используется;
- бит 0 - линия KB-Data, в режиме AT не используется.

В современных ПК используются только биты 0, 1 и 7; в режиме AT биты 0 и 1 не используются.

Порт вывода **P2,** доступный для записи и чтения по командам D1h и D0h соответственно, имеет следующее назначение разрядов:

- бит 7 - выходная линия KB-Data;
- бит 6 - выходная линия KB-Clock;
- бит 5 - запрос прерывания от дополнительного интерфейса (IRQ12), в режиме AT неиспользуется;
- бит 4 - запрос прерывания от клавиатуры (IRQ1);
- бит 3 - линия MS-Clock, в режиме AT не используется;
- бит 2 - линия MS-Data, в режиме AT не используется;
- бит 1 - вентиль линии адреса A20 *(Gate A20*): 0 - A20 обнулен, 1 - управляется выводом

процессора;

- бит 0 - альтернативный сброс процессора (сигнала INIT, без формирования общего сигнала сброса); при переключении значения из "1" в "0" возможен сброс или остановки по исключительным ситуациям.

*Тестовый порт T***,** доступен для чтения командой E0h, имеет следующее назначение разрядов:

- бит 0 (T0) - входная линия KB-Clock;
- бит 1 (T1) - входная линия MS-Clock, в режиме AT - входная линия KB-Data.

Регистры адаптера

В адаптере имеются также программно-доступные регистры: регистр состояния, регистр командного байта, регистр команд и регистр данных.

Регистр состояния представляет собой 8-разрядный регистр, который доступен системе только как читаемый порт с адресом 64h. Он содержит информацию о состоянии адаптера и интерфейсов клавиатуры и мыши. Может быть прочитан центральным процессором в любое время.

Значение разрядов регистра следующее:

Значение разрядов регистра следующее:

- бит 0 — признак "выходной буфере заполнен" (OBF) (0 — буфер пуст, 1 — буфер заполнен);
- бит 1 — признак "входной буфер заполнен" (0 — буфер пуст, 1 — буфер заполнен);
- бит 2 — признак типа последнего общесистемного сброса (0 — сброс по включению питания, 1 — программный сброс);
- бит 3 — признак записи команды (0 — последняя операция записи являлась операцией записи данных, 1 — записи команды);
- бит 4 — состояние «замка» клавиатуры (0 — клавиатура блокирована, 1 — не блокирована);
- бит 5 — в AT-режиме: признак "ошибка тайм-аута передачи" (0 — нормальное завершение передачи, 1 — произошла ошибка); в PS/2-режиме: признак "выходной буфер мыши заполнен" (Mouse_OBF) (0 — буфер пуст, 1 — буфер заполнен);
- бит 6 — в AT-режиме: признак "ошибка тайм-аута приема" (0 — нормальное завершение приема, 1 — произошла ошибка); в PS/2-режиме: общий признак "ошибка тайм-аута при приеме или передаче данных" (0 — нормальное завершение операции, 1 — произошла ошибка);
- бит 7 — признак: "ошибка паритета при приеме или передаче данных" (0 — нет ошибки, 1 — обнаружена ошибка по четности).

Командный байт адаптера клавиатуры хранится во внутреннем регистре адаптера и определяет режимы работы адаптера и интерфейсов. Прочитать и записать командный байт можно с помощью команд адаптера 20h и 60h соответственно (подробнее см. Команды адаптера).

Разряды регистра имеют следующее назначение:

Разряды регистра имеют следующее назначение:

- бит 0 — управление выдачей сигнала прерывания по готовности данных в выходном буфере клавиатуры (выходной буфер адаптера) (0 — генерация прерывания запрещена, 1 — разрешена);
- бит 1 — управление выдачей сигнала прерывания по готовности данных в выходном буфере мыши (0 — генерация прерывания запрещена, 1 — разрешена);
- бит 2 — установка признака системного сброса (значение, записанное в этот разряд, переносится в разряд 2 регистра состояния);
- бит 3 — управление функцией блокировки клавиатуры (0 — функция блокировки разрешена, 1 — функция запрещена и блокировка игнорируется);
- бит 4 — управление интерфейсом клавиатуры (0 — обмен данными с клавиатурой разрешен, 1 — запрещен);
- бит 5 — признак типа протокола передачи данных в AT-режиме (0 — протокол IBM); управление интерфейсом мыши в PS/2-режиме (0 — обмен данными с мышью разрешен, 1 — запрещен);
- бит 6 — преобразование скан-кодов в PC-совместимые (0 — выключено, 1 — включено); по умолчанию имеет значение 1, то есть скан-коды преобразуются в PC-совместимые;
- бит 7 — не используется (зарезервирован).

Регистр команд адаптера доступен только по записи через порт 64h. В него заносятся коды команд, которые должен выполнять адаптер. Команды можно посылать в порт 64h только при условии, что входной буфер адаптера пуст (бит 1 регистра состояния =0).

Регистр данных доступен для записи и считывания через порт 60h. В режиме считывания он служит для приема информации от клавиатуры, адаптера и мыши. В режиме записи регистр данных служит для передачи команд клавиатуре, координатному устройству (мыши PS/2-типа) и их адаптеру. Какого-либо особого формата данный регистр не имеет.

Команды адаптера

Управление адаптером клавиатуры и манипулятора мышь осуществляется одно– или двухбайтными командами. Первый, возможно единственный, байт команды записывается в порт 64h, а следующий (атрибут команды), если он есть, записывается в порт 60h.

Коды команд адаптера приведены в табл. 1.2.

Табл. 1.2. Коды команд адаптера клавиатуры и мыши

Код	Функция
20h	Прочитать текущий командный байт. Адаптер помещает значение командного байта в порт 060h.
60h	Загрузить новый командный байт. Это часть 2-байтовой операции. Чтобы записать новый командный байт, необходимо 1) записать 60h в порт 064h, 2) записать командный байт в порт 060h
A1h	Чтение номера версии адаптера
A4h	Проверка пароля (признака подключения клавиатуры). При успешном завершение команды через выходной буфер возвращается код F1h (PS/2 адаптер).
A5h	Специальное чтение - 8042 помещает в выходной буфер истинное значение порта 2 за исключением разрядов 4 и 5, которым дается новое определение. Не генерируется состояние «Буфер полон». Если разряд 5 = 0, используется 9-разрядная клавиатура (XT); 5 = 1, используется 11-разрядная клавиатура; 4 = 0, запрещается прерывание по заполнению выходного буфера; 4 = 1, разрешается прерывание по заполнению выходного буфера. В PS/2 адаптерах данная команда инициирует процедуру установки пароля.
A7h	Запретить работу с мышью. Выполняется только в режиме PS/2.
A8h	Разрешить работу с мышью. Выполняется только в режиме PS/2.
A9h	Тест интерфейса мыши. При успешном выполнение команды в выходной буфер помещается код 00h.
AAh	Инициализация – 8042 инициализирует порты 1 и 2, запрещает клавиатуру и очищает указатели буфера. Затем он помещает код 55h в выходной буфер.
ABh	Контроль интерфейса клавиатуры - направляет 8042 на тестирование линий данных и синхронизации интерфейса клавиатуры. Выходной буфер (порт ввода 60h) получает результаты тестирования: 00h - ошибок не обнаружено; 01h - линия синхронизации клавиатуры «остается нулевой»; 02h - линия синхронизации клавиатуры «остается единичной»; 03h - линия данных клавиатуры «остается нулевой»; 04h - линия данных клавиатуры «остается единичной».
ACh	Дамп диагностирования. Резервируется для диагностирования
ADh	Запретить клавиатуру. Устанавливает разряд 4 командного байта 8042, который запрещает интерфейс клавиатуры. До разрешения клавиатуры данные не передаются и не принимаются
AEh	Разрешить клавиатуру. Сбрасывает разряд 4 командного байта 8042, который разрешает интерфейс клавиатуры
AFh	Выдать номер версии
C0h	Прочитать входной порт. Настраивает 8042 на передачу содержимого входного порта и помещение его в выходной буфер (порт ввода 60h). *Пользуйтесь этой командой только при пустом выходном буфере*
D0h	Прочитать выходной порт. Настраивает 8042 на передачу текущего байта выходного порта в выходной буфер (порт ввода 60h). *Пользуйтесь этой командой только при пустом выходном буфере.*
D1h	Записать в выходной порт. Поместить следующий байт, записанный по адресу 60h, в выходной порт. Разряд 0 выходного порта подключен к системному сбросу. Этот разряд не должен быть нулевым.
D2h	Записать байт в выходной буфер клавиатуры.
D3h	Записать байт в выходной буфер мыши (PS/2 режим).
D4h	Передать манипулятору мышь байт данных (PS/2 режим). Команда D4h

	автоматически разрешает интерфейс мыши, сбрасывая D5 командного байта адаптера в ноль (т.е. дополнительно выполняется команда A8h)
E0h	Прочитать входы TEST0 и TEST1 (тестовый порт). Настраивает 8042 на передачу текущего состояния выводов TEST0 и TEST1 в выходной буфер (порт вывода 60h). TEST0 - это разряд 0, а TEST1 - 1
F0h – FFh	Пульсация выходного порта. Может быть вызвана пульсация разрядов 0-3 выходного порта (перевод в нулевое состояние примерно на 2-6 мкс). Разряды 0-3 в этой команде указывают, какой разряд выходного порта должен пульсировать, если 0, пульсация разрешена, если 1 - запрещена. Заметим, что разряд 0 выходного порта связан со сбросом системы. Пульсация разряда 0 может привести к сбросу системы. Код команды FFh выполняется адаптером ничего не изменяя в нем (нулевая или холостая команда).

Некоторые адаптеры в режиме PS/2 могут выполнять дополнительный набор команд, использовавшийся в компьютерах PS/2. В основном эти команды расширяют диагностические возможности и позволяют организовывать парольную защиту. Ниже приводится описание этого набора команд.

Считать байт памяти (20h-3Fh). Данные команды предназначены для считывания памяти адаптера клавиатуры. Биты 5-0 команды определяют шестнадцатеричный адрес внутренней памяти 8042, т.е. сам код команды несет в себе параметр. Команда с кодом 20h (считать управляющий (командный) байт), таким образом, является всего лишь частным случаем этой группы команд. Запрошенный байт памяти помещается адаптером в выходной буфер.

Записать байт памяти (60h-7Fh). Данные команды позволяют изменить содержимое памяти адаптера клавиатуры, при этом, аналогично командам 20h-3Fh, биты 5-0 команды определяют шестнадцатеричный адрес памяти, по которому записывается байт данных, передаваемый вслед за командой по адресу 60h.

Проверить установку пароля (A4h). Данная команда предназначена для проверки наличия установленного пароля. Результат проверки в виде одного байта помещается в ответ на эту команду в выходной буфер (адрес 60h). При этом значение FAh означает, что пароль установлен, а значение F1h, что пароль не установлен.

Установить пароль (A5h). Данная команда инициирует процедуру установки пароля. После получения команды адаптер считывает из входного буфера байты, составляющие собственно пароль. Нулевой байт в передаваемой последовательности означает, что все байты, составляющие пароль, переданы. Байты, составляющие пароль, должны передаваться в формате скан-кодов клавиш. Пароль может быть установлен в любое время, при этом старое значение пароля теряется.

Задание пароля, состоящего из единственного нулевого байта, равносильно отмене пароля.

Следует отметить, что адаптер не предоставляет никакой команды для считывания установленного пароля.

Включить парольную защиту (A6h). По этой команде адаптер клавиатуры включает функцию парольной защиты, что означает, что до получения адаптером последовательности скан-кодов, определенной в пароле, адаптер не передаст в систему никакой информации и не реагирует на команды со стороны системы. Обычно данная команда выполняется при начальном включение компьютера, а также при выполнении системной утилиты, входящей в состав базового программного обеспечения для PS/2. С точки зрения пользователя, после выполнения этой команды компьютер оказывается заблокированным до введения пароля.

Функционирование адаптера

При обеспечение передачи данных с клавиатуры и манипулятора мышь в систему адаптер под управлением своего внутреннего программного обеспечения выполняет ряд предусмотренных интерфейсом действий.

Например, приняв посылку от клавиатуры, адаптер выполняет внутреннюю трансляцию скан-кода (если это не запрещено командным байтом) и помещает его в выходной буфер адаптера (клавиатуры), устанавливает в регистре состояния флажок OBF=1 (разряд 0), что приводит к генерации запроса прерывания IRQ1 (если это не запрещено командным байтом). В ответ на это хост должен прочитать данные (транслированный скан-код, префиксы и т. п.) из порта данных (60h). После чтения адаптер сбрасывает флажок OBF в ноль. Трансляция обеспечивает программную совместимость клавиатур XT и AT (см. ниже) по чтению скан-кодов из порта 60h.

При получении посылки от мыши адаптер не выполняет никаких преобразований и помещает ее в выходной буфер мыши, устанавливает в регистре состояния флажок Mouse_OBF=1 (разряд 5), что приводит к генерации запроса прерывания IRQ12 (если это не запрещено командным байтом). Затем, примерно через 15 мкс, адаптер переписывает полученный байт из выходного буфера мыши в выходной буфер адаптера (клавиатуры) и устанавливает в регистре состояния флажок OBF=1 (разряд 0). По этому флагу данные от мыши должны быть считаны хостом (системой) из порта 60h. После чтения адаптер сбрасывает флажок OBF а флажок Mouse_OBF оставляет в единичном состоянии (подробности см. в

разделах 1.1.8.3.5. и 1.1.8.3.6.).

При передаче команд и данных в клавиатуру адаптер выполняет сравнительно простые операции: он просто передает в клавиатуру байт данных, который был записан системой в его входной буфер по адресу 60h и ожидает ответного кода от клавиатуры. Получив ответный код, адаптер просто помещает его в выходной буфер без трансляции. Остальные действия соответствуют ранее рассмотренным в процедуре получения посылки от клавиатуры.

При передаче команд и данных в PS-Mouse необходимо сначала по адресу 64h передать в адаптер команду D4h "Передать мыши байт данных". После освобождения входного буфера по адресу 60h необходимо записать байт, который адаптер должен передать манипулятору мышь. Адаптер передает этот байт мыши и ожидает ответного кода. Получив от мыши ответный код, адаптер помещает его в выходной буфер мыши. Остальные действия соответствуют ранее рассмотренным в описании процедуры получения посылки от PS Mouse.

После получения по адресу 64h команды, по которой адаптер должен возвращать данные, он помещает байт данных в выходной буфер и устанавливает в регистре состояния OBF=1, что приводит к генерации запроса прерывания IRQ1 (если это не запрещено командным байтом). После этого данные должны быть считаны из порта данных (по адресу 60h). Если команда возвращает несколько байтов данных, прерывание генерируется для каждого байта.

После получения по адресу 64h кода команды, которая должна содержать дополнительный байт атрибутов, адаптер инициирует процедуру выполнения этой команды. Далее он освобождает входной буфер, устанавливает в регистре состояния признак "Входной буфер пуст" (разряд 1 устанавливается в 0) и ожидает получения от системы через регистр данных (входной буфер) с адресом 60h атрибутного байта команды. После его получения адаптер завершает выполнение этой команды и освобождает входной буфер.

Следует отметить, что на время выполнения адаптером команд, адаптер блокирует интерфейсы клавиатуры и мыши, устанавливая низкий уровней на их линиях синхронизации KB-Clock и MS-Clock. Аналогично при приеме данных от клавиатуры адаптер блокирует интерфейс мыши на время передачи одного байта и наоборот.

Информация о программировании адаптера представлена в разделе 1.1.8. учебного пособия.

Более подробную информацию о адаптере клавиатуры и PS/2 Mouse представлена в разделе 1.1 учебного пособия.

4. Указания по подготовке к лабораторной работе.

При подготовке к лабораторной работе, ее выполнении и оформлении отчета необходимо следовать рекомендациям раздела 4.1."Общие указания по выполнению и оформлению лабораторных работ".

Подготовка к данной лабораторной работе заключается в изучении описания лабораторной работы "Адаптер клавиатуры и PS/2 Mouse" и раздела 1.1.8. "Программирование адаптера " учебного пособия. Необходимо также ознакомиться с основными командами отладчика AFD и с командами процессора i8086.

5. Порядок выполнения лабораторной работы.

1. Ознакомиться с приведенными ниже программами.
2. В окне дизассемблера отладчика AFD набрать Программу 1, приведенную в разделе 6 описания, проверить и запустить ее командой "G" отладчика.
3. Проанализировать результаты выполнения программы.
4. Выполнить пункты 2 и 3 для Программ 2, 3, и 4.
5. Провести расчеты и сравнить полученные результаты с приведенными в описании к программам.
6. Получить дополнительное задание от преподавателя.

6. Примеры программ

Программа 1. Программа 1 аналогична программе "Чтение состояния выходного порта адаптера (CKLD0.BIN)", представленной в Листинге 1.1 раздела 1.1.8.3.1. Она позволяет получить информацию о времени выполнения команд адаптера, в которых предусматривается получение результата их выполнения через выходной буфер адаптера (адрес 60h). Текст программы 1 (Kkl99.txt) представлен ниже в листинге 1. В программе измерение времени выполнения команды осуществляется подсчетом в регистре CX числа циклов ожидания появления в регистре состояния адаптера (адрес 64h) признака "выходной буфер заполнен" (флаговый разряд регистра состояния D0=1). Число циклов помещается программой в регистр BX. Результат выполнения команды адаптера программа помещает в регистр AL. На время работы программы запрещается обработка маскируемых прерываний с целью устранения вмешательства обработчиков прерываний клавиатуры и канала 0 таймера в работу программы.

Для определения времени выполнения команды в микросекундах необходимо определение времени выполнения одного цикла проверки значения флагового разряда регистра состояния. Это значение можно определить с помощью программ 2 или 3, представленных в листингах 2 и 3 соответственно.

Программы 1, 2 и 3 выполняются в среде отладчика AFD. В зависимости от ОС, из которой запускался отладчик, временные параметры выполнения команд адаптера получаются значительно отличающимися друг от друга.

Листинг 1. Программа выполнения команд чтения адаптера (Kkl99.txt).

```
252D:0100  FA      CLI            ;Запрет прерываний.
252D:0101  E464    IN     AL,[64] ;Ожидание освобождения
252D:0103  A802    TEST   AL,02   ;входного буфера
252D:0105  75FA    JNZ    0101    ;адаптера.
252D:0107  B9FFFF  MOV    CX,FFFF ;Исходное значен. счетчика циклов.
252D:010A  B0D0    MOV    AL,D0   ;Команда "прочитать выходной порт
252D:010C  E664    OUT    [64],AL ;посылается адаптеру.
252D:010E  E460    IN     AL,[60] ;Очистка вых. буфера адаптера.
252D:0110  E464    IN     AL,[64] ;Проверка поступл. результата вып.
252D:0112  A801    TEST   AL,01   ;команды D0h в вых. буф. контр. со
252D:0114  E1FA    LOOPZ  0110    ;счетом в CX числа циклов проверки.
252D:0116  E460    IN     AL,[60] ;Чтение результата вып. ком. D0h.
252D:0118  FB      STI            ;Разрешение прерываний.
252D:0119  BBFFFF  MOV    BX,FFFF ;Подсчет числа циклов ожидания
252D:011C  29CB    SUB    BX,CX   ;результата выполнения команды D0h.
252D:011E  CC      INT3           ;Возврат в отладчик.
```

Программа 2. Программа 2 позволяет определить время выполнения цикла опроса регистра состояния адаптера клавиатуры и PS/2 Mouse, в привязке к частоте работы счетчика канала 0 системного программируемого таймера i8254. BIOS на стадии POST настраивает канал 0 таймера на двоичный счет константы пересчета FFFFh в режиме 3 (генератор меандра с частотой 18,3 Гц на выходе канала - период 55 мс). Счетчик канала 0 работает в этом режиме постоянно после включения питания персонального компьютера. В программе реализуются 100 циклов опроса разряда D0 регистра состояния адаптера клавиатуры с фиксацией в регистре BX содержимого счетчика канала 0 таймера перед началом циклов и в регистре AX - после окончания циклов опроса. Разница значений счетчика помещается программой в регистр BX. Эта разница показывает длительность выполнения 100 циклов опроса регистра состояния адаптера измеренную в периодах синхросигналов, поступающего на вход 16- разрядного счетчика-делителя канала 0. По документации IBM PC/AT на входы счетчиков каналов программируемого таймера i8254 поступают сигналы с частотой 1,19 МГц (период 0,84 мкс).

Текст программы 2 представлен в листинге 2.

Листинг 2. Определение времени цикла опроса в привязке к каналу 0 таймера (Izmo.txt).

```
252D:0100  FA      CLI            ;Запрет прерываний.
252D:0101  B96400  MOV    CX,0064 ;100 циклов вып. программы.
252D:0104  E81100  CALL   0118    ;Значение счетчика помещается
252D:0107  89C3    MOV    BX,AX   ;в регистр BX.
252D:0109  E464    IN     AL,[64] ;Чтение регистра состояния,
252D:010B  A801    TEST   AL,01   ;проверка нулевого разряда -
252D:010D  E2FA    LOOP   0109    ;повторение 100 раз.
252D:010F  E80600  CALL   0118    ;Значение счетчика в AX.
252D:0112  29C3    SUB    BX,AX   ;Разность знач. счетчика в BX.
252D:0114  FB      STI            ;Разрешение прерываний.
252D:0115  CC      INT3           ;Возврат в отладчик.
252D:0116  90      NOP
252D:0117  90      NOP
252D:0118  90      NOP
252D:0119  E440    IN     AL,[40] ;Подпрограмма чтения содержимого
252D:011B  88C4    MOV    AH,AL   ;счетчика канала 0 таймера,
252D:011D  E440    IN     AL,[40] ;результат возвращается
252D:011F  86E0    XCHG   AH,AL   ;в регистре AX.
252D:0121  90      NOP
252D:0122  C3      RET            ;Возврат из подпрограммы.
```

Программа 3. Программа 3 позволяет определить время выполнения цикла опроса регистра состояния адаптера клавиатуры и PS/2 Mouse, в привязке к частоте работы счетчика канала 2 системного программируемого таймера i8254. BIOS на стадии POST настраивает канал 2 таймера на двоичный счет константы пересчета 4E6h в режиме 3 (генератор меандра с частотой 1000 Гц на выходе канала - период 1 мс). Счетчик канала 2 работает в этом режиме после установки в 1 разряда D0 системного порта 61h. В

программе производится настройка канала 2 таймера на двоичный счет константы пересчета FFFFh в режиме 3 и восстановление параметров канала в конце работы программы.

Как и программа 2 программа 3 производит измерение длительности выполнения 100 циклов опроса регистра состояния адаптера клавиатуры. Однако измерения производятся в периодах сигналов синхронизации, поступающих на вход 16-разрядного счетчика-делителя канала 2 таймера. Следует отметить что в относительно современных ПК (начиная с К6) значения частот сигналов, поступающих на программно доступные счетчики каналов 0 и 2 таймера могут не совпадать.

Текст программы 3 представлен в листинге 3.

Листинг 3. Определение времени цикла опроса в привязке к каналу 2 таймера (Izmo2.txt).

```
2C06:0100  FA      CLI            ;Запрет прерываний.
2C06:0101  B96400  MOV    CX,0064 ;100 циклов вып. программы.
2C06:0104  B0B6    MOV    AL,B6   ;Инициализация канала 2
2C06:0106  E643    OUT    [43],AL ;таймера на режим 3
2C06:0108  B0FF    MOV    AL,FF   ;с двоичным счетом
2C06:010A  E642    OUT    [42],AL ;константы пересчета
2C06:010C  E642    OUT    [42],AL ;FFFFh.
2C06:010E  B021    MOV    AL,21   ;Разрешение счета
2C06:0110  E661    OUT    [61],AL ;в канале 2 таймера.
2C06:0112  E81F00  CALL   0134    ;Значение счетчика помещается
2C06:0115  89C3    MOV    BX,AX   ;в регистр BX.
2C06:0117  E464    IN     AL,[64] ;Чтение регистра состояния,
2C06:0119  A801    TEST   AL,01   ;проверка нулевого разряда -
2C06:011B  E2FA    LOOP   0117    ;повторение 100 раз.
2C06:011D  E81400  CALL   0134    ;Значение счетчика в AX.
2C06:0120  29C3    SUB    BX,AX   ;Разность знач счетчика в BX.
2C06:0122  B0B6    MOV    AL,B6   ;Восстановление режима
2C06:0124  E643    OUT    [43],AL ;работы канала 2
2C06:0126  B0E8    MOV    AL,E8   ;таймера (режим 3 с двоичным
2C06:0128  E642    OUT    [42],AL ;счетом константы
2C06:012A  B003    MOV    AL,03   ;03E8h - 1000 Гц на
2C06:012C  E642    OUT    [42],AL ;выходе счетчика.
2C06:012E  B020    MOV    AL,20   ;Запрет счета в канале 2
2C06:0130  E661    OUT    [61],AL ;таймера
2C06:0132  FB      STI            ;Разрешение прерываний.
2C06:0133  CC      INT3           ;Возврат в отладчик.
2C06:0134  90      NOP
2C06:0135  E442    IN     AL,[42] ;Подпрограмма чтения содержимого
2C06:0137  88C4    MOV    AH,AL   ;счетчика канала 2 таймера,
2C06:0139  E442    IN     AL,[42] ;результат возвращается
2C06:013B  86E0    XCHG   AH,AL   ;в регистре AX.
2C06:013D  90      NOP
2C06:013E  C3      RET            ;Возврат из подпрограммы.
```

Результаты работы программы Kkl99.bin на различных ПК с операционными системами DOS 7, Windows 95/98 и Windows XP представлены в табл.1

Таблица 1.

ОС	DOS 7	Windows 95/98	Windows XP
Pent. III-500МГц.	72h (114) - /160/319 мкс	57h (87) - /417/409 мкс	FFFFh (65535)
K6 mmx-200 МГц.	3Ah (58) - /75/145 мкс	18h (24) - /187/187 мкс	-
K6 tm-133 МГц.	3Eh (62) - /161/155 мкс	13h (19) - /195/190 мкс	-

В скобках указано десятичное значение числа циклов.

После первого знака / представлено время выполнения команды D0h в микросекундах подсчитанное в привязке к частоте работы счетчика канала 0 таймера, а после второго - в привязке к частоте работы счетчика канала 2 (см. табл. 2 и 3).

Результаты работы программ Izmo.bin и Izmo2.bin на различных ПК с операционными системами MS DOS, Windows 95/98 и Windows XP представлены в таблице 2.

Таблица 2.

ОС	DOS 7		Windows 95/98		Windows XP	
	Канал 0	Канал 2	Канал 0	Канал 2	Канал 0	Канал 2
Pent. III-500МГц.	A9h (169)	150h (336)	238h (568)	22Eh (558)	209Dh (8350)	25C9h (9675)
K6 mmx-200 МГц.	95h (149)	128h (296)	3A0h (928)	3B2h (946)	-	-

K6 tm-133 МГц.	134h (308)	130h (304)	4CCh (1228)	4B0h (1200)	-	-

В скобках указано десятичное значение числа тиков счетчика.

Анализируя значения тиков счетчиков, представленных в табл.2 можно сделать следующие выводы:

- частота работы счетчика канала 0 таймеров компьютеров Pent. III-500МГц и K6 mmx-200 МГц в два раза ниже частоты счетчика канала 2 таймеров;
- в среде операционных систем Windows эта разница частот корректируется.

Используя значения тиков каналов 0 и 2 таймеров, представленных в табл.2 и принимая во внимание значение частоты работы счетчиков каналов (1,19 МГц, период - 0,84 мкс) можно определить время выполнения одного цикла опроса регистра состояния адаптера клавиатуры и PS/2 Mouse для соответствующих ПК, работающих под управлением соответствующих ОС. Для этого необходимо взять из табл. 2 соответствующее десятичное значение тиков, умножить его на период 0,84 мкс и разделить на 100. Полученные значения представлены в табл. 3.

Таблица 3.

ОС	DOS 7		Windows 95/98		Windows XP	
	Канал 0	Канал 2	Канал 0	Канал 2	Канал 0	Канал 2
Pent. III-500 МГц.	1,4 мкс	2,8 мкс	4,8 мкс	4,7 мкс	70 мкс	81 мкс
K6 mmx-200 МГц.	1,3 мкс	2,5 мкс	7,8 мкс	7,8 мкс		-
K6 tm-133 МГц.	2,6 мкс	2,5 мкс	10,3 мкс	10 мкс		-

Используя данные табл. 3 и табл. 1 можно определить время выполнения команды D0h контролером клавиатуры. Соответствующие подсчитанные значения времени в микросекундах представлены в табл.1.

Анализируя время выполнения команды D0h можно отметить следующие факты:

- время выполнения команды в среде DOS 7 не менее чем на 20% меньше, чем в среде Windows 95/98. Это связано с тем, что в среде Windows 95/98 программы реального режима адресации выполняются в среде виртуальной машины 8086;
- время выполнения команды на ПК типа Pentium III-500 МГц примерно в два раза больше, чем на ПК типа K6 как в DOS 7, так и в Windows . Это связано с тем, что адаптер i8042 в таких компьютерах эмулируется программно, что позволяет через порты 60h и 64h работать с клавиатурой и мышью как через интерфейс PS/2, так и через интерфейс USB. При обращении к портам 60h и 64h хостконтроллер USB вырабатывает прерывание SMI (System Management Interrupt - прерывание системного управления), обрабатывающееся на ПК с процессорами x86 в режиме SMM (System Management Mode - режим системного управления) невидимо для обычных программ. Обработка ведется в особой области памяти, недоступной процессору в обычных режимах работы.

Программа 4. Программа 4 позволяет измерять время выполнения адаптером клавиатуры команды D0h также и с помощью осциллографа, подключаемого к одной из линий шины данных LPT порта. Для этого в программу вставлены команды вывода констант FFh и 00h в регистр данных параллельного порта LPT 2. Эти команды выполняются после передачи кода команды D0h в порт 64h адаптера клавиатуры, и после установки в регистре состояния адаптера клавиатуры флага (разряд D0) "выходной буфере адаптера (клавиатуры) заполнен" (OBF) (0 — буфер пуст, 1 — буфер заполнен). Для обеспечения регистрации сигналов на линиях шины данных параллельного порта обычным (не запоминающим) осциллографом программа выполняется циклически. Число циклов определяется содержимым регистра BX.

Текст программы 4 представлен в листинге 4.

Листинг 4. Таймирование выполнения команды D0h через LPT порт (Kkl98.txt).

```
252D:0100  BB0001  MOV    BX,0100 ;Число циклов выполнения программы.
252D:0103  BA7803  MOV    DX,0378 ;Адрес RD LPT2.
252D:0106  FA      CLI            ;Запрет прерываний.
252D:0107  E464    IN     AL,[64] ;Ожидание освобождения
252D:0109  A802    TEST   AL,02   ;входного буфера
252D:010B  75FA    JNZ    0107    ;адаптера.
252D:010D  B9FFFF  MOV    CX,FFFF ;Счетчик циклов ожидания.
252D:0110  B0D0    MOV    AL,D0   ;Код команды
252D:0112  E664    OUT    [64],AL ;передается адаптеру.
252D:0114  E460    IN     AL,[60] ;Очистка вых. буфера адаптера.
252D:0116  B0FF    MOV    AL,FF   ;Пересылка в RD LPT2 константы FFh
252D:0118  EE      OUT    DX,AL   ;(уст.высок. уровня на лин.D0-D7).
252D:0119  E464    IN     AL,[64] ;Проверка поступл. результата вып.
252D:011B  A801    TEST   AL,01   ;команды D0h в вых.буф.контр.со
252D:011D  E1FA    LOOPZ  0119    ;счетом в CX числа циклов проверки.
252D:011F  B000    MOV    AL,00   ;Пересылка в RD LPT2 константы 00h
252D:0121  EE      OUT    DX,AL   ;(уст.низкого уровня на лин.D0-D7).
```

```
252D:0122  E460    IN     AL,[60] ;Результат вып.команды D0h читается в AL.
252D:0124  90      NOP
252D:0125  4B      DEC    BX      ;Определение окончания циклов
252D:0126  75DE    JNZ    0106    ;выполнения программы.
252D:0128  BBFFFF  MOV    BX,FFFF ;Подсчет числа циклов ожидания
252D:012B  29CB    SUB    BX,CX   ;результата выполнения команды D0h.
252D:012D  FB      STI            ;Разрешение прерываний.
252D:012E  CC      INT3           ;Возврат в отладчик.
```

Результаты работы программы Kkl98.bin на различных ПК с операционными системами DOS 7 и Windows 95/98 представлены в табл.4

Таблица 4.

ОС	DOS 7	Windows 95/98
Pent. III-500 МГц.	66h (102) - /146/142/285 мкс	7Bh (123) - /350/590/579 мкс
K6 mmx-200 МГц.	48h (72) - /96/94/180 мкс	15h (21) - /120/163/163 мкс
K6 tm-133 МГц.	46h (70) - /96/182/175 мкс	Fh (15) - /120/154/150 мкс

В скобках указано десятичное значение числа циклов.

После первого знака "/" представлено время выполнения команды D0h в микросекундах, измеренное по осциллографу, после второго - подсчитанное в привязке к скорости работы счетчика канала 0 таймера, после третьего - подсчитанное в привязке к скорости работы счетчика канала 2 таймера.

Анализируя данные, представленные в табл. 4 можно отметить следующие особенности:

- в среде DOS 7 время выполнения команды D0h, измеренное с помощью осциллографа совпало со временем, измеренным с помощью счетчиков таймера только в случае использования канала 0 таймеров ПК Pent.III-500 МГц и K6 mmx-200 МГц,; для каналов 2 таймеров всех использованных ПК, а для ПК K6 tm-133 МГц и для канала 0 значение времени превышает измеренное по осциллографу примерно в два раза;
- в среде Windows 95/98 время выполнения команды D0h, измеренное с помощью осциллографа примерно на 30% меньше, чем измеренное с помощью счетчиков каналов 0 и 2.

Первую особенность можно объяснить следующим образом. В архитектуре PC/AT сигналы с частотой 2,38 МГц поступали на входы каналов таймера через предварительный делитель на 2, т. е. счетчики каналов работали с частотой 1,19 МГц (период - 0,84 мкс). В архитектуре первых ПК Pentium (до K6) на вход счетчиков подавался сигнал с частотой 2,38 МГц (период - 0,42 мкс) а делитель на 2 ставился на выходе счетчиков-делителей каналов. Начиная с ПК K6 mmx и старше канал 0 работал с делителем на входе счетчика (частота 1,19 МГц, период - 0,84 мкс) а канал 2 - с делителем на выходе счетчика (частота 2,38 МГц, период - 0,42 мкс).

Вторая особенность автору не совсем понятна.

7. Рекомендуемая литература

При подготовке к лабораторной работе и к ее защите рекомендуется пользоваться информацией, представленной в описании лабораторной работы, в разделе 1.1. настоящего учебно-методического пособия и в дополнительной литературе [5, 6, 8].

8. Контрольные вопросы

1. Какие новые качества приобрел адаптер клавиатуры в архитектуре PC/AT?
2. Дайте краткое описание функционирования адаптера клавиатуры и PS-Mouse.
3. Охарактеризуйте блок-схему адаптера клавиатуры PC/AT (Рис. 1.2).
4. Охарактеризуйте схему подключения микроконтроллера i8042, представленную на Рис. 1.3.
5. Охарактеризуйте интерфейс подключения адаптера клавиатуры к системе.
6. Дайте краткое описание интерфейса подключения клавиатуры и PS-Mouse к адаптеру.
7. Дайте краткое описание формата регистра состояния адаптера клавиатуры и PS-Mouse.
8. Охарактеризуйте регистр командного байта адаптера.
9. Дайте общую характеристику командам адаптера клавиатуры и PS-Mouse.
10. Охарактеризуйте дополнительные команды, поддерживаемые адаптером клавиатуры в режиме PS/2.
11. Дайте краткую характеристику портам ввода и вывода адаптера клавиатуры и PS-Mouse.
12. Кратко опишите действия адаптера по обслуживанию взаимодействия системы с клавиатурой и PS-Mouse.
13. Сформулируйте правила программного взаимодействия с адаптером клавиатуры.
14. Сформулируйте правила программного взаимодействия с клавиатурой через адаптер клавиатуры.
15. Сформулируйте правила программного взаимодействия с PS-Mouse через адаптер клавиатуры.
16. Сформулируйте правила программирования портов адаптера клавиатуры.
17. Охарактеризуйте программу, представленную в листинге 1.1.

18. Дайте комментарии результатам выполнения команд адаптера, представленным в таблице раздела 1.1.8.3.1.
19. Поясните результаты, полученные при выполнении команд адаптера в рамках программы, представленной в листинге 1.2.
20. Поясните результаты, полученные при выполнении команд адаптера в рамках программы, представленной в листинге 1.3.
21. Поясните результаты, полученные при выполнении команд адаптера в рамках программы, представленной в листинге 1.4.
22. Поясните результаты, полученные при выполнении команд адаптера в рамках программы, представленной в листинге 1.5.
23. Поясните результаты, полученные при выполнении команд адаптера в рамках программы, представленной в листинге 1.6.
24. Поясните результаты, полученные при выполнении команд адаптера в рамках программы, представленной в листинге 1.7.
25. Как проверяется адаптер клавиатуры процедурой POST?

3. Лабораторная работа 2 (описание) "Электронная клавиатура и ее интерфейс"

1. Цель работы.

Целью лабораторной работы является ознакомление студентов с принципами работы электронной клавиатуры ПК; изучение функционирования электрической схемы клавиатуры с регистрвцией сигналов в контрольных точках схемы с помощью электронного осциллографа; изучение сигнального протокола интерфейса клавиатуры.

2. Технические и программные средства , используемые в работе.

В работе используется персональный компьютер с клавиатурой, имеющей зондовое устройство доступа к контрольным точкам ее электрической схемы (ЗУДКТ). ПК работает под управлением ОС Windows 95/98/Mellenium. Операционная система загружается в режиме DOS (Command prompt only) с оболочкой типа Norton Commander. Непосредственное низкоуровневое программное взаимодействие с клавиатурой осуществляется в среде отладчика типа AFD.

В работе используется также электронный осциллограф с полосой регистрации 0 - 10 МГц и выше.

3. Краткое описание клавиатуры и ее интерфейса.

3.1. Конструктивы 101-клавишной клавиатуры

На рис.1 представлен внешний вид используемой в лабораторной работе 101-клавишной клавиатуры X-RING (FCC ID: E8HKB-5311) (1). Клавиатура подключается к системному блоку ПК интерфейсным кабелем (2). В клавиатуру вмонтировано зондовое устройство доступа к контрольным точкам электрической схемы клавиатуры (ЗУДКТ) (3).

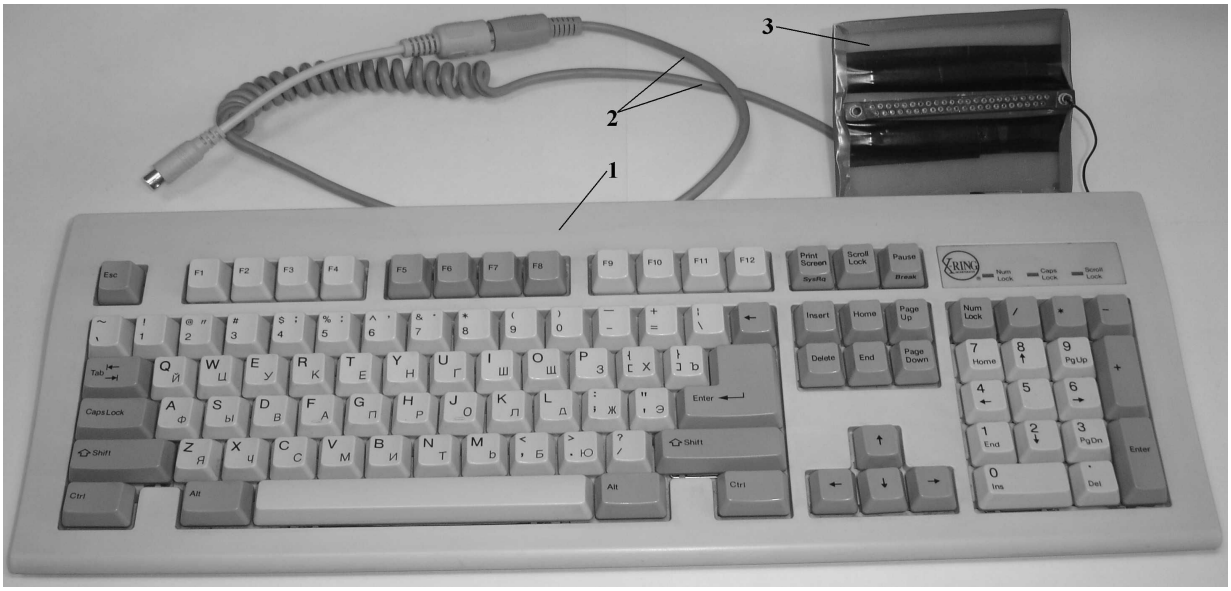

Рис. 1. 101-клавишная клавиатура с ЗУДКТ: 1 - клавиатура; 2 - интерфейсный кабель; 3 - ЗУДКТ.

В состав ЗУДКТ входит контактное поле доступа к контрольным точкам, представленное разъемом (6) типа ГРПМ1-45ГО2-В (см. рис. 2), матрицы (7) токоограничивающих резисторов номиналом 680 Ом и плоский кабель (8), подключенный своими проводниками-зондами к контрольным точкам электрической схемы с одной стороны, и к матрицам резисторов - с другой.

На рис. 3 приведена общая электрическая схема 101-клавишной клавиатуры фирмы X-RING. Основным элементом клавиатуры является микроконтроллер U1 (9306S или i8042). Отечественным аналогом данного микроконтроллера является микроконтроллер КМ1816ВЕ48. Так же на плате находится микросхема U2 (74 LS05). Аналогом данной микросхемы являебтся К555ЛН2 (шесть инверторов с открытым коллектором).

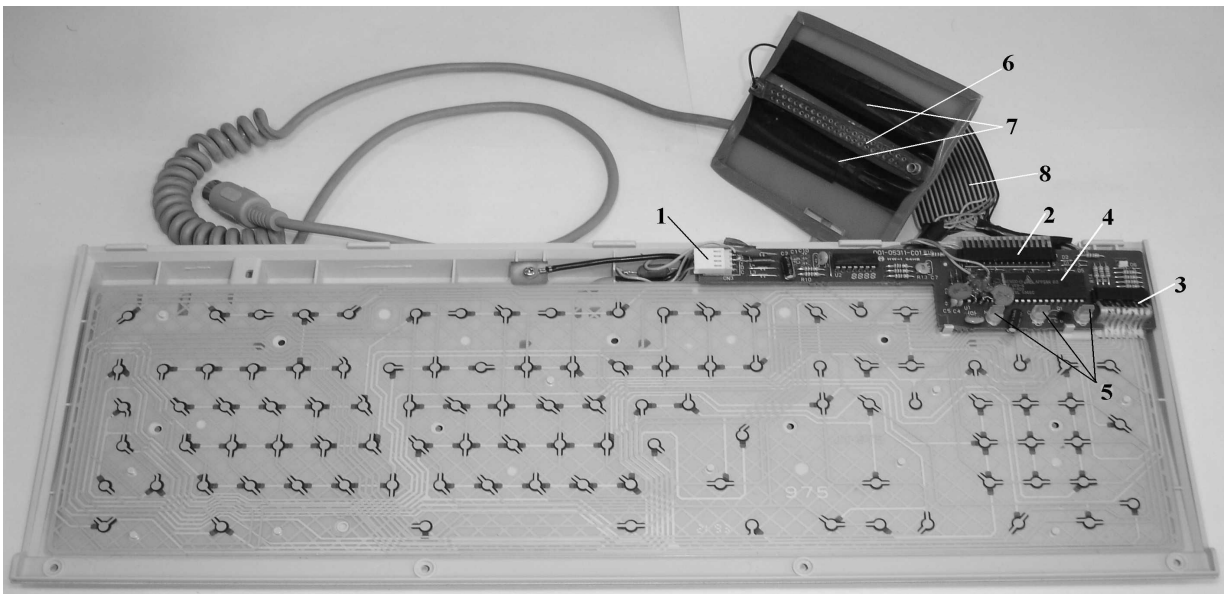

Рис. 2. Компоновка клавиатур с ЗУДКТ: 1 - разъем интерфейсного кабеля; 2 - разъем подключения столбцов матрицы контактов датчиков клавиш (15 групп (столбцов опроса)); 3 - разъем подключения строк матрицы контактов датчиков клавиш (8 групп (сток приема)); 4 - микроконтроллер; 5 - светодиоды; 6 - контактное поле ЗУДКТ; 7 - матрицы резисторов; 8 - зондовый кабель.

3.2. Схемотехника клавиатуры

В состав эклектической схемы клавиатуры входят следующие компоненты:

- Интерфейсный кабель, содержащий четыре линии:
  - **G** -сигнальная земля;
  - **V** -питание +5В;
  - **D** -двунаправленная линия передачи данных в последовательном формате;
  - **C** -двунаправленная линия передачи синхросигналов (линия синхронизации);
- **R9, R10** -нагрузочные резисторы передатчиков двунаправленных линий синхронизации и данных, выполненных на инверторах с открытым коллектором;
- **C9, C10** -конденсаторы фильтрации высокочастотных наводок на линиях данных и синхронизации;
- **C1, C2, C7, C8** -конденсаторы фильтрации высокочастотных наводок в шинах питания (развязка по питанию);
- **U2** -микросхема 74LS05 (К555ЛН2) - шесть инверторов с открытым коллектором: два инвертора используются для передачи сигналов по двунаправленным линиям интерфейса (линии данных (D) и линии синхронизации (C)); три - для управления светодиодами клавиатуры (SCR, CAP и NUM); один - для приема синхросигналов с двунаправленной линией синхронизации (C).
- **D1-D5, D7** -диоды, реализующие логическую операцию "ИЛИ";
- **X1** -кварцевый резонатор (5 мГц);
- **R1 - R8** -резисторы, обеспечивающие подачу напряжения смещения +5В на линии входного порта (D0-D7) микроконтроллера U1;
- **CN2** -входной разъем подключения строк матрицы контактных датчиков клавиш;
- **CN1** -выходной разъем подключения столбцов матрицы контактных датчиков клавиш;
- **U1** -микроконтроллер (однокристальная микро-ЭВМ) типа 9306S (аналог i8042, КМ1816ВЕ48).

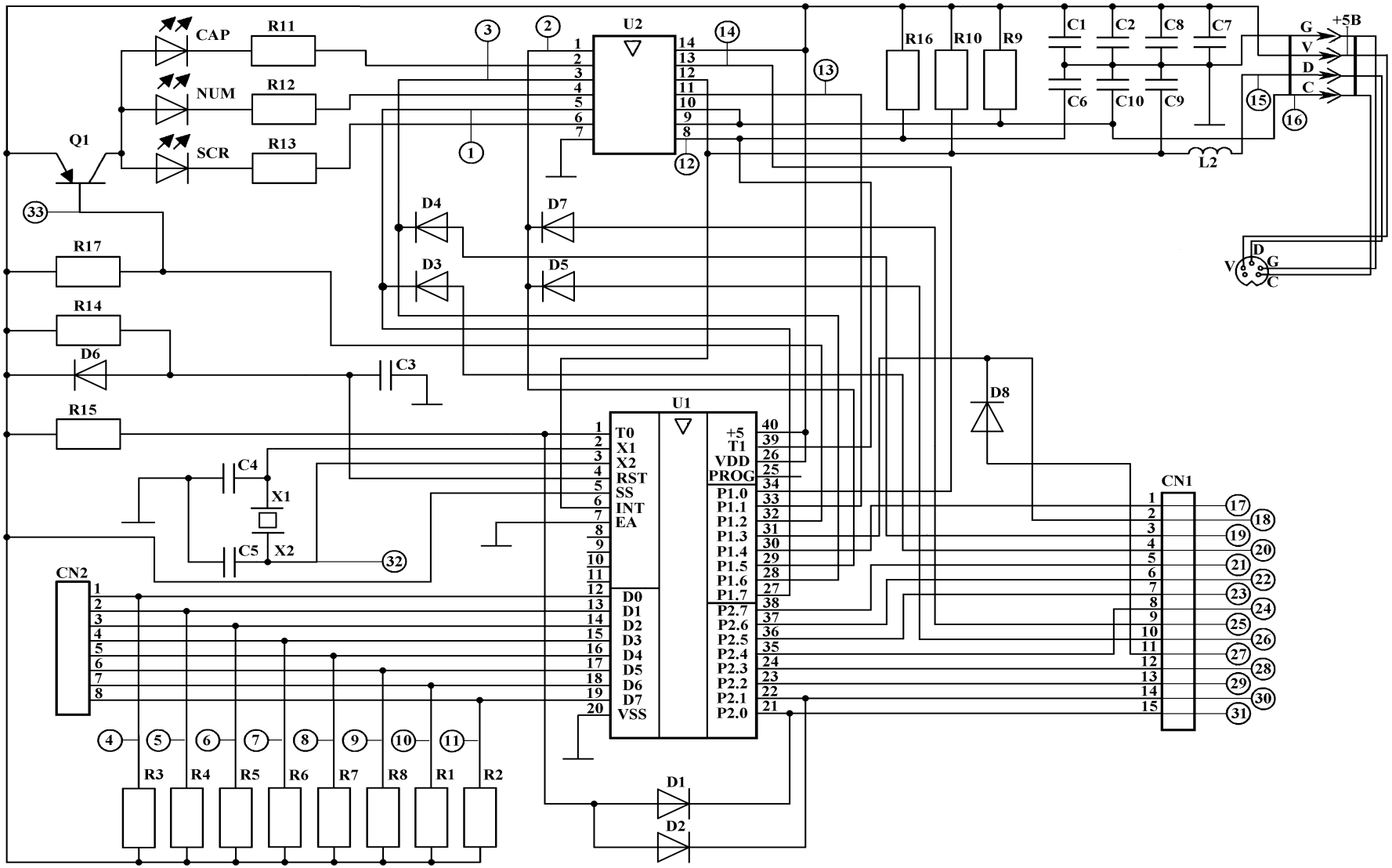

Рис. 3 Общая электрическая схема клавиатуры

В табл. 1 представлен список контрольных точек электрической схемы электронной 101-клавишной клавиатуры X-RING и соответствующие контакты разъема ЗУДКТ. В лабораторной работе используются клавиатуры с двумя вариантами соединения контактов разъема ЗУДКТ и контрольных точек электрической схемы - вариант 1 и вариант 2.

Таблица 1.

Контрольные точки		Контакты разъема ЗУДКТ	
		Вариант 1	Вариант 2
1	Сигналы включения светодиода Scroll Lock и импульсы опроса столбцов матрицы датчиков клавиш	Б21	А1
2	Сигналы включения светодиода Caps Lock и импульсы опроса столбцов матрицы датчиков клавиш.	Б23	А2
3	Сигналы включения светодиода Num Lock и импульсы опроса столбцов матрицы датчиков клавиш.	Б22	А3
4 - 11	Контакты на разъеме CN2, - 8 битный двунаправленный порт ввода-вывода информации (выводы D0-D7 микроконтроллера U1) используются для приема сигналов со строк матрицы датчиков клавиш.	Б20 - Б13	А4 -А11
12	Инвертированный входной сигнал линии синхронизации интерфейса (вывод 8 микросхемы U2)	Б12	А12
13	Инвертированный выходной сигнал синхронизации интерфейса (вывод 33 микросхемы U1)	Б11	А13
14	Инвертированный выходной сигнал данных интерфейса (вывод 34 микросхемы U1)	Б10	А14
15	Сигнал на линии данных интерфейса (KB-Data)	Б9	А15
16	Сигнал на линии синхронизации интерфейса (KB-Clock)	Б8	А16
17, 18, 21 - 24, 28 - 31	Сигналы на контактах разъема CN1, непосредственно подключенные к квази двунаправленным портам ввода-вывода микроконтроллера U1, и используемые для опроса столбцов матрицы датчиков клавиш (10 столбцов)	А22, А21, А18 - А15, А11 - А8	Б1, Б2, Б5 - Б8, Б12 - Б15
19, 20, 26	Сигналы на контактах разъема CN1, подключенные к выводам (27 - 29) квази двунаправленного порта ввода-вывода P1 микроконтроллера U1 через разделительные диоды D3 - D5, и используемые для опроса столбцов матрицы датчиков клавиш (3 столбца). Эти выводы порта P1 управляют также через U2 светодиодами SCR, CAP и NUM.	А20, А19, А13	Б3, Б4, Б10
25	Сигнал опроса клавиши правый Enter, подключаемый диодом D7 с выхода 29 U1 к контакту 9 разъема CN1 при нажатие этой клавиши.	А14	Б9
27	Сигнал опроса клавиши Page Down, подключаемый диодом D8 с выхода 31 U1 к контакту 11 разъема CN1 при нажатие этой клавиши.	А12	Б11
32	Сигнал на кварцевом резонаторе	А7	Б16
33	Сигнал управления транзистором Q1 цепи питания светодиодов.	А6	Б17

3.3. Интерфейс клавиатуры

Для подключения клавиатуры к системе используется последовательный двунаправленный интерфейс, включающий в себя две обязательные линии сигналов KB-Data и KB-Clock. Обе линии на системной плате подтягиваются резисторами к шине +5 В. На обеих сторонах интерфейса выходные сигналы низкого уровня формируются выходами элементов с открытым коллектором (стоком), а состояние линий может быть прочитано через входные линии адаптеров. Вид разъемов подключения клавиатуры к системному блоку (со стороны задней панели) и назначение их контактов приведены на рис. 4.

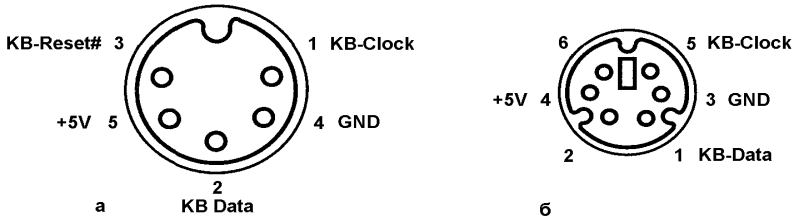

Рис.4. Разъемы (вид со стороны контактов) подключения клавиатур: *а* - AT и *б* - PS/2

Процессор общается с клавиатурой через *адаптер клавиатуры* — микроконтроллер i8042 или программно-совместимый с ним, установленный на системной плате. Для обмена данными используется порт 60h адаптера клавиатуры, через который принимаются данные из клавиатуры и посылаются в клавиатуру команды и их атрибуты. О необходимости чтения байта данных, поступившего в выходной буфер порта 60h, адаптер сигнализирует процессору через аппаратное прерывание IRQ1. Сигнал прерывания вырабатывается по каждому событию клавиатуры, сопровождаемому посылкой байта в адаптер клавиатуры (нажатию и отпусканию клавиши, поступлению команды или ее атрибута из адаптера и т. п.). Задание параметров автоповтора, выбор таблиц скан-кодов, управление светодиодными индикаторами, а также управление режимом сканирования матрицы клавиш и запуск диагностического теста осуществляется программно командами, посылаемыми клавиатуре через порт 60h. Адаптер транслирует команды в посылки, направляемые к клавиатуре через последовательный двунаправленный интерфейс.

При включении питания контроллер, находящийся на клавиатуре генерирует сигнал "сброс по включению питания" (POR) на время от 150 мс до 2,0 с, после чего переходит к выполнению базового теста. Базовый тест (BAT) состоит из теста процессора клавиатуры, контрольной суммы ПЗУ (ROM) и теста ЗУПВ (RAM). Во время процедуры BAT сигналы на линиях KB-Clock и KB-Data игнорируются. Светодиоды включаются в начале BAT и выключаются в конце BAT. BAT занимает от 300 до 500 мс (не считая времени, требуемого POR).

После успешного завершения BAT в систему передается код завершения (AAh) и начинается сканирование клавиатуры. При возникновении ошибки BAT клавиатура передает системе код ошибки (FDh). После этого клавиатура прекращает сканирование и ожидает ввода команды. Коды завершения теста посылаются через 0,45-2,0 с после POR и через 300-500 мс после подтверждения команды RESET.

Немедленно после POR клавиатура регистрирует сигналы на линиях KB-Clock и KB-Data и устанавливает протокол линии. Схема связей клавиатуры представлена на рис. 5.

Рис. 5. Схема связей клавиатуры и системы

Клавиатура и система общаются между собой посредством линий синхросигналов KB-Clock и данных KB-Data. Источниками сигналов на каждой из этих линий служат устройства с открытым коллектором, которые позволяют клавиатуре или системе установить линии на неактивный (низкий) уровень. Когда никакого обмена данными не происходит, линии синхросигнала и данных находятся на активном (высоком) уровне.

Когда адаптер системы посылает данные на клавиатуру, он устанавливает линию данных на неактивный (низкий) уровень и позволяет линии синхросигнала быть на активном уровне. Передатчик линии синхронизации клавиатуры обеспечивает синхросигналы, которые используются адаптером системы для синхронизации передачи данных на клавиатуру.

Когда клавиатура передает данные системе она сама генерирует сигналы данных и синхронизации. Система может запретить клавиатуре передавать данные, установив линию синхросигнала на неактивный (низкий) уровень. При этом линия данных может быть активной или не активной.

Передача данных на клавиатуру и с клавиатуры осуществляется потоком 11-разрядных данных (режим 2 и 3), которые передаются последовательно по линии данных. При активном (высоком) уровне пересылается логическая 1. Назначение разрядов показано в табл. 2.

Разряд паритета может содержать 1 или 0, так что 8 разрядов данных вместе с разрядом паритета всегда содержат нечетное число единиц.

В режиме 1 передача данных осуществляется потоком 9-разрядных данных, в которых отсутствуют разряд паритета и стоповый разряд.

Протокол ввода данных с клавиатуры. Когда клавиатура готова к передаче данных, она сначала проверяет состояние линий синхросигнала и данных, чтобы определить, нет ли запрета клавиатуры, или, запрашивает ли система передачу.

Если линия синхросигнала не активна (низкий уровень), данные сохраняются в буфере клавиатуры.

Если линия синхросигнала активна (высокий уровень), а линия данных не активна (системой сформирован запрос на передачу и стартовый разряд START BITc), данные также сохраняются в буфере клавиатуры, а клавиатура переходит в режим приема данных от системы, осуществляя синхронизацию этого процесса.

Таблица. 2. Назначение разрядов

Разряд	Назначение
1	Стартовый (всегда 0)
2	Разряд данных 0 (младший)
3	Разряд данных 1
4	Разряд данных 2
5	Разряд данных 3
6	Разряд данных 4
7	Разряд данных 5
8	Разряд данных 6
9	Разряд данных 7 (старший)
10	Разряд паритета
11	Стоповый (всегда 1)

Если линии синхронизации и данных активны (высокий уровень), клавиатура начинает генерировать синхросигналы и передает нулевой стартовый разряд (START BITk), 8 разрядов данных, разряд паритета PARITY и стоповый разряд STOP (см. диаграмму на рис. 6б). Данные будут действительны от заднего фронта и до переднего фронта синхроимпульса. Период Tcy синхроимпульса не менее 60 мкс, отрицательный полу период Tcl не менее 5 мкс. Клавиатура изменяет состояние линии данных за время Thk (минимальное значение 0 мкс) до формирования заднего фронта синхросигнала. Во время передачи клавиатура проверяет линию синхросигнала по меньшей мере через каждые 60 мкс.

Если после начала передачи данных с клавиатуры система переводит линию синхросигнала на неактивный уровень, имеет место состояние, которое называется конфликтной ситуацией на линии, и клавиатура прекращает передачу данных. Если конфликтная ситуация возникает до переднего фронта 10-го синхроимпульса (разряд паритета), буфер клавиатуры возвращает линии синхросигнала и данных на активный уровень.

Если до 10-го синхроимпульса конфликтной ситуации не возникает, клавиатура завершает передачу. После конфликтной ситуации система может запросить у клавиатуры повторную посылку данных.

После приема данных от клавиатуры система может заблокировать клавиатуру на время обработки поступивших данных или пока система не запросит ответ от клавиатуры (участок [6] временной диаграммы). При работе в MS DOS после ввода очередного кода с клавиатуры клавиатура блокируется системой путем удержания линии синхронизации в пассивном состоянии примерно на 400 мкс. Этот сигнал может рассматриваться интерфейсом клавиатуры как сигнал подтверждения приема системой информации от клавиатуры (ACKc).

В PC AT блокировка линии синхронизации может осуществляться программно командой адаптера клавиатуры ADh "Запретить клавиатуру", а разблокирование - командой AEh "Разрешить клавиатуру". В PC XT эти операции осуществляются установкой и сбросом разряда 7 порта 61h.

Временная диаграмма передачи данных с клавиатуры в систему показана на рис. 6б.

Протокол вывода данных в клавиатуру. Когда система готова к передаче данных в клавиатуру, она сначала проверяет, не передает ли клавиатура данные. Если клавиатура передает данные, но еще не передала 10-й синхросигнал, система может заблокировать вывод данных с клавиатуры путем установки линии синхросигнала клавиатуры на неактивный (низкий) уровень. Если клавиатура передала 10-й синхроимпульс, система должна принять передаваемые данные.

Если клавиатура не передает данные или если клавиатура передает данные, но еще не передала 10-й синхросигнал, система устанавливает линию синхросигнала клавиатуры на неактивный уровень на время более 60 микросекунд (мкс) и готовится в это время к передаче данных. Когда система готова к передаче (время T1 на рис. 6а), она устанавливает стартовый разряд START BITc (линия данных не активна) а затем через время T2 разрешает линии синхросигнала подняться на активный уровень.

В этом режиме клавиатура проверяет состояние линии синхросигнала с интервалами не более 10 мкс. Если обнаруживается состояние запроса передачи (RTS), клавиатура переходит в состояние считывания разрядов, передаваемых системой по линии данных. Считывание начинается после завершения состояния RTS формированием отрицательного фронта сигнала CLOCK (время T3). Клавиатура считывает 10 разрядов (START BITc, 8 битов данных и бит паритета PARITY). После этого клавиатура проверяет уровень линии данных и, если эта линия активна (системой сформирован стоповый разряд STOP), переводит ее на неактивный уровень (формирует сигал ACKk) и считывает еще один разряд. Это указывает системе, что клавиатура приняла переданные данные. После этого *адаптер* (система) формирует импульс KB-Clock длительностью не менее 60 мкс, который является запросом на прием ответа от клавиатуры (состояние [4] на временной диаграмме рис 6а). Адаптер ожидает окончания ответа на этот запрос не долее 20 мс и, если ответ не придет за это время, сформирует ошибку тайм-аута. Ошибка будет также в случае, если клавиатура не введет первый синхроимпульс за 15 мс от начала запроса при

передаче, или адаптер не примет данные, включая стоп-бит, за 2 мс с момента появления синхроимпульса бита 0 при приеме.

Если после чтения 10-го разряда обнаруживается неактивное состояние линии данных, это значит, что возникла ошибка кодирования, и клавиатура продолжает попытки чтения до тех пор, пока линия данных не станет активной. После этого клавиатура понижает уровень линии данных и передает системе команду RESEND (FEh).

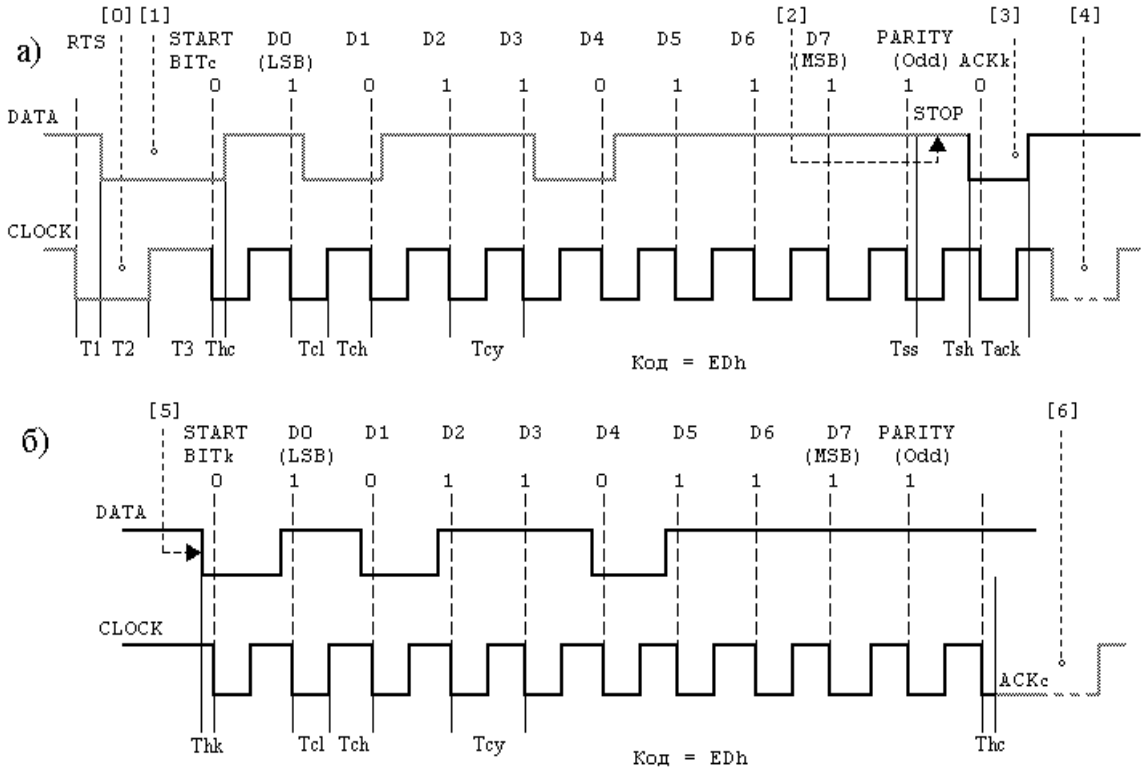

Рис. 6. Временные диаграммы связи клавиатуры и системы: а) - передача данных системой в клавиатуру; б) - передача данных клавиатурой в систему (серым цветом обозначены сигналы, формируемые системным адаптером, черным - клавиатурой).

[0] - CLOCK - совместно с низким уровнем линии DATA рассматривается клавиатурой как сигнал RTS - запрос на передачу данных от системы в клавиатуру;

[1] - DATA - сигнал низкого уровня, формируемый системой как стартовый разряд передаваемых в клавиатуру данных;

[2] - DATA - сигнал высокого уровня, формируемый системой как стоповый разряд STOP;

[3] - DATA - сигнал низкого уровня, устанавливаемый клавиатурой как сигнал ACKk подтверждения приема данных от системы;

[4] - CLOCK - сигнал низкого уровня, формируемый системой как сигнал запроса на передачу клавиатурой байта подтверждения приема байта команды или байта ее атрибута;

[5] - DATA - сигнал низкого уровня, формируемый клавиатурой как стартовый разряд START BITk при передаче данных с клавиатуры в систему;

[6] - CLOCK - сигнал низкого уровня, формируемый системой после приема очередного байта данных с клавиатуры; может использоваться клавиатурой, как сигнал подтверждения приема системой очередного байта данных с клавиатуры (ACKc).

При каждой передаче команды или данных от системы на клавиатуру система требует подтверждения от клавиатуры (код ответа FAh - логическое ACKk). Если только система не запретит вывод данных с клавиатуры, последняя дает этот ответ в течение 20 мс.

Если ответ клавиатуры недействителен или содержит ошибку паритета, система повторно посылает ей команду. Однако двухбайтовые команды требуют особой обработки. Если были переданы и подтверждены команды F3h (установить скорость/задержку повторения), F0h (выбрать альтернативные коды сканирования) или EDh (установить/сбросить индикаторы режима), а также передан байт данных (атрибут команды), но ответ оказался недействительным или с ошибкой паритета, то система передаст повторно и команду и ее атрибут.

4. Указания по подготовке к лабораторной работе.

При подготовке к лабораторной работе, ее выполнении и оформлении отчета необходимо следовать рекомендациям раздела 4.1."Общие указания по выполнению и оформлению лабораторных работ".

Подготовка к данной лабораторной работе заключается в изучении описания лабораторной работы "Электронная клавиатура и ее интерфейс" и разделов 1.2. - 1.2.5 учебного пособия. Необходимо также ознакомиться с основными командами отладчика AFD и с командами процессора i8086.

5. Порядок выполнения лабораторной работы.

1. Ознакомиться с описанием электрической схемы клавиатуры и описанием ее контрольных точек.
2. Загрузить операционную систему Windows 95/98/Mel. в режиме Command Prompt Only, нажимая клавишу F8 перед началом загрузки ОС.
3. С помощью электронного осциллографа определить электрические параметры сигналов в контрольных точках электрической схемы клавиатуры (состояние контрольных точек - амплитуда, длительность и период сигналов или напряжение постоянного уровня):
   - 3.1. Определить состояние контрольных точек 1, 2, и 3 при включенных и выключенных соответствующих светодиодах SCR, CAP и NUM.
   - 3.2. Определить состояние одной из контрольных точек 4 - 11, указанной преподавателем, при нажатие последовательно всех клавиш клавиатуры при выключенных светодиодах SCR, CAP и NUM, а затем при включенных светодиодах. Зафиксировать в протоколе работы те клавиши, при нажатие которых в контрольной точке появлялись последовательности импульсов (с указанием параметров импульсов). Эта информация будет необходима для выполнения пункта 2.11.задания.
   - 3.3. Определить состояние контрольных точек 12 - 16 при нажатие и отпускание клавиши, указанной преподавателем и определить 16-ричное значение ее скан кода.
   - 3.4. В контрольных точках 17, 18, 21 - 24, 28 - 31 определить параметры сигналов опроса 10 столбцов матрицы датчиков клавиш
   - 3.5. Определить, при нажатие каких клавиш появляются сигналы в контрольных точках 19, 20 и 26.
   - 3.6. Определить, у каких клавиш, подключенных к контрольным точкам 19, 20 и 26 (задание 2.5) и как изменяется форма сигналов опроса их датчиков при изменение состояния режимов Caps Lock, Num Lock и Scroll Lock. Эта информация будет необходима для выполнения пункта 2.11.задания.
   - 3.7. Определить, как влияют режимы Caps Lock, Num Lock и Scroll Lock на форму сигналов в контрольных точках 25 и 27 при нажатие клавиш "Правый Enter" и "Page Down" соответственно.
   - 3.8. Определить параметры сигнала кварцевого резонатора в контрольной точке 32.
   - 3.9. Определить параметры сигналов в контрольной точке 33 при активных и пассивных режимах Caps Lock, Num Lock и Scroll Lock.
   - 3.10. Подключить контакт контрольной точки 31 к входу "Внешняя синхронизация" осциллографа, переключить осциллограф в режим внешней синхронизации и определить задержки импульсов опроса в контрольных точках 17, 18, 21 - 24, 28 - 30 по отношению к импульсу в контрольной точке 31.
   - 3.11. Пользуясь результатами пункта 2.10. определить порядок опроса клавиш, установленных в результате выполнения пункта 2.2. Для этого использовать правила пункта 2.10. подключения осциллографа, а сигналы фиксировать в контрольной точке, указанной в пункте 2.2. Сигнал с более отрицательной задержкой будет идентифицировать клавишу, опрашиваемую первой в данном цикле опроса группы клавиш, определенной в пункте 2.2., а с максимальной положительной задержкой - последней. Для идентификации клавиш, сигналы датчиков которых не синхронизируются необходимо переключать режимы Caps Lock, Num Lock и Scroll Lock в соответствие с информацией, полученной в пункте

3.12. Получить дополнительное задание от преподавателя.

4. В окне дизассемблера отладчика AFD набрать программу 1, представленную в разделе 6 описания лабораторной работы, запустить ее командой "G" отладчика и с помощью осциллографа определить параметры сигналов в контрольных точках 15 и 16 (линии KB-Data и KB-Clock соответственно). На основание полученных данных и описания программы 1 определить параметры протокола обмена по интерфейсу клавиатуры (см. рис.6., описание протокола в разделе 3 описания лабораторной работы и в разделе 1.2.5.4. учебного пособия).
5. В окне дизассемблера отладчика AFD набрать программу 2, представленную в разделе 6 описания лабораторной работы, запустить ее командой "G" отладчика и с помощью осциллографа определить параметры сигналов в контрольных точках 15 и 16 (линии KB-Data и KB-Clock соответственно). На основание полученных данных и описания программы 2 определить дополнительные параметры протокола обмена по интерфейсу клавиатуры.

6. Примеры программ.

Программа KLEE1.TXT. Для определения конкретных временных параметров сигнального протокола интерфейса клавиатуры можно воспользоваться простой программой (см. листинг 1) (программа KLEE1.TXT), выполняемой в среде отладчика DOS (Debug или AFD).

Листинг 1. Программа KLEE1.TXT. - циклическое выполнение команды клавиатуры "Эхо" (код EEh).

```
2321:0100  FA        CLI           ;Запрет аппаратных прерываний.
2321:0101  BB0001    MOV    BX,0100;Число циклов выполнения программы.
2321:0104  E464      IN     AL,[64];Чтение рег. состояния адаптера и
2321:0106  A802      TEST   AL,02  ;проверка: Вх. буфер пуст? Если нет,
2321:0108  75FA      JNZ    0104   ;то новый цикл проверки.
2321:010A  B0EE      MOV    AL,EE  ;Передача кода EEh в клавиатуру
2321:010C  E660      OUT    [60],AL;через порт 60h.
2321:010E  E464      IN     AL,[64];Чтение рег. состояния адаптера и
2321:0110  A801      TEST   AL,01  ;проверка признака "вых. буфер полон":
2321:0112  74FA      JZ     010E   ;если пуст, то на новый цикл.
2321:0114  E460      IN     AL,[60];Чтение кода ответа клавиатуры из вых.
2321:0116  88C4      MOV    AH,AL  ;буфера и запись его в регистр AH.
2321:0118  B0AE      MOV    AL,AE  ;Команда "разрешить интерфейс клавиат."
2321:011A  E664      OUT    [64],AL;передается в адаптер (код AEh).
2321:011C  B90F00    MOV    CX,000F;Организация
2321:011F  51        PUSH   CX     ;вложенного
2321:0120  B9FFFF    MOV    CX,FFFF;цикла
2321:0123  E2FE      LOOP   0123   ;программной
2321:0125  59        POP    CX     ;задержки
2321:0126  E2F7      LOOP   011F
2321:0128  4B        DEC    BX     ;Уменьшение счетчика циклов BX.
2321:0129  75D9      JNZ    0104   ;Если BX не 0, то на новый цикл прогр.
2321:012B  FB        STI           ;Разрешение маскируемых прерываний.
2321:012C  CC        INT3          ;Остановка и передача упр. отладчику.
```

В программе реализовано циклическое выполнение команды клавиатуры "Эхо" (код EEh, см. раздел 1.2.6.2.2). Код команды EEh передается через интерфейс в последовательном формате в клавиатуру, которая в ответ передает адаптеру клавиатуры код ответа EEh в том же формате. Подключившись осциллографом к линиям KB-Clock и KB-Data интерфейса клавиатуры можно фиксировать электрические сигналы, реализующие передачу этих кодов на физическом уровне в соответствие с сигнальным протоколом аппаратного интерфейса клавиатуры. На рис. 7 представлены осциллограммы этих сигналов интерфейса клавиатуры. Сигналы с более отрицательной амплитудой формируются системным адаптером клавиатуры, а с менее отрицательной - клавиатурой.

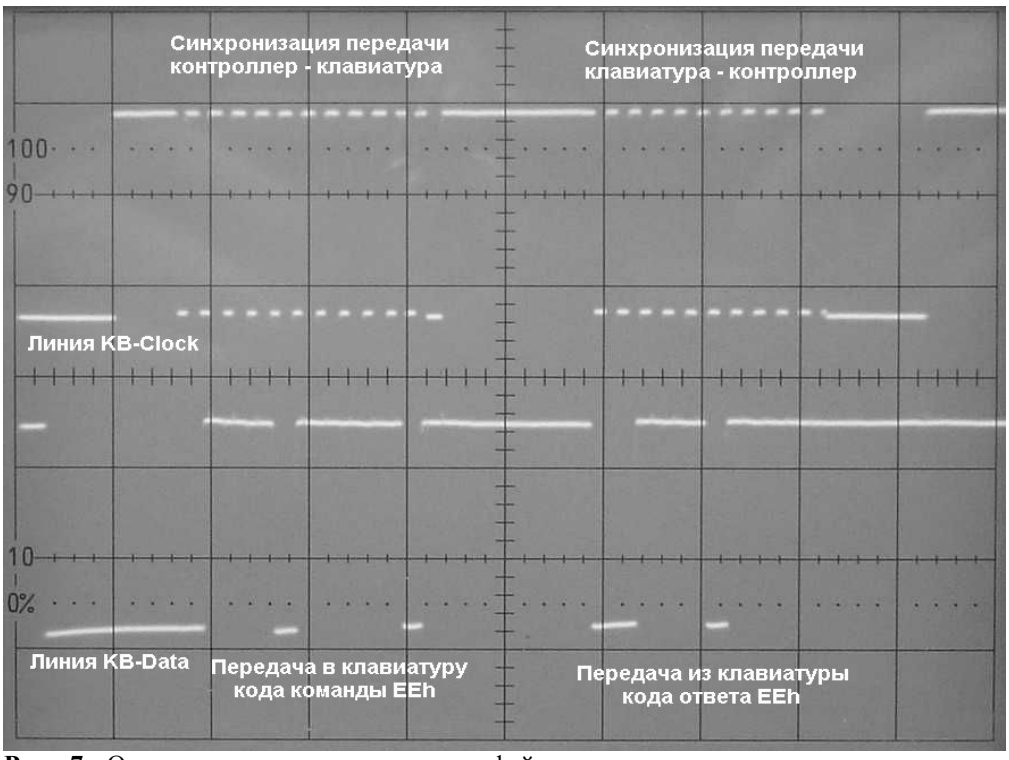

Рис. 7. Осциллограммы сигналов интерфейса клавиатуры, зарегистрированные при выполнении программы KLEE1.BIN: развертка - 0,36 мс/дел., чувствительность по напряжению - 2 В/дел.; развертка запускается отрицательным фронтом сигнала линии KB-Clock.

Программа Klf2.TXT. Для определения времени пауз между байтами, посылаемыми клавиатурой при выполнении много байтовых операций можно воспользоваться программой KLF2.TXT, представленной в листинге 2.

Листинг 2 Программа KLF2.TXT. - циклическое выполнение команды клавиатуры "Идентификатор расширенной клавиатуры" (код F2h).

```
2BF9:0100  FA      CLI             ;Запрет прерываний.
2BF9:0101  BB0002  MOV    BX,0200  ;Счетчик циклов программы.
2BF9:0104  E464    IN     AL,[64]  ;Проверка признака "входной буфер
2BF9:0106  A802    TEST   AL,02    ;полон"; если 1, то повторная
2BF9:0108  75FA    JNZ    0104     ;проверка, если 0, далее.
2BF9:010A  B0F2    MOV    AL,F2    ;Команда F2h "Идентификатор расширенной
2BF9:010C  E660    OUT    [60],AL  ;клавиатуры" передается в клавиатуру.
2BF9:010E  E464    IN     AL,[64]  ;Ожидание поступления от клавиатуры
2BF9:0110  A801    TEST   AL,01    ;в выходной буфер кода
2BF9:0112  74FA    JZ     010E     ;подтверждения FAh.
2BF9:0114  E460    IN     AL,[60]  ;Чтение кода FAh и помещение его
2BF9:0116  88C4    MOV    AH,AL    ;в регистр AH.
2BF9:0118  E464    IN     AL,[64]  ;Ожидание поступления от клавиатуры
2BF9:011A  A801    TEST   AL,01    ;первого кода идентификатора
2BF9:011C  74FA    JZ     0118     ;расширенной клавиатуры.
2BF9:011E  E460    IN     AL,[60]  ;Чтение первого кода и помещение его
2BF9:0120  88C2    MOV    DL,AL    ;в регистр DL.
2BF9:0122  E464    IN     AL,[64]  ;Ожидание поступления от клавиатуры
2BF9:0124  A801    TEST   AL,01    ;второго кода идентификатора
2BF9:0126  74FA    JZ     0122     ;расширенной клавиатуры.
2BF9:0128  E460    IN     AL,[60]  ;Чтение второго кода и помещение его
2BF9:012A  88C6    MOV    DH,AL    ;в регистр DH.
2BF9:012C  B9FF00  MOV    CX,00FF  ;Программная задержка перед
2BF9:012F  51      PUSH   CX       ;новым циклом
2BF9:0130  B9FFFF  MOV    CX,FFFF  ;выполнения программы,
2BF9:0133  E2FE    LOOP   0133     ;реализующей выполнение команды
2BF9:0135  59      POP    CX       ;"Идентификатор расширенной
2BF9:0136  E2F7    LOOP   012F     ;клавиатуры".
2BF9:0138  4B      DEC    BX       ;Счетчик циклов уменьшается на 1.
2BF9:0139  75C9    JNZ    0104     ;Если BX не 0, то новый цикл.
```

```
2BF9:013B  FB      STI             ;Разрешение прерываний.
2BF9:013C  CC      INT3            ;Возврат в отладчик.
```

Программа циклически посылает в клавиатуру код команды F2h и принимает от клавиатуры ответ на эту команду: код подтверждения FAh, младший байт идентификатора расширенной клавиатуры и затем старший байт идентификатора. Программа выполняется в среде отладчика AFD. Результаты выполнения программы заносятся в регистры AH, DL и DH, а также наблюдаются на экране осциллографа, подключенного к линиям KB-Data и KB-Clock интерфейса клавиатуры. Пример наблюдаемых осциллограмм представлен на рис.8.

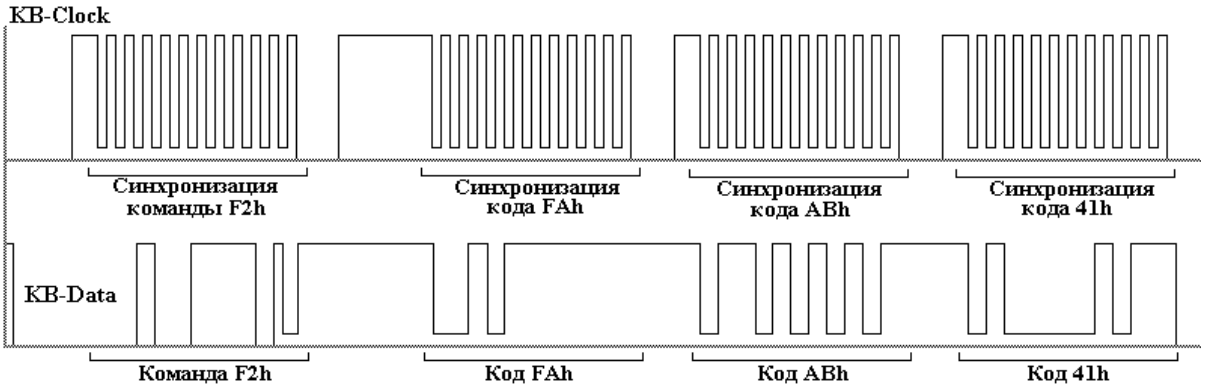

Рис. 8. Осциллограммы, наблюдаемые на линиях KB-Clock и KB-Data при выполнении программы KLF2.TXT.

Сигналы с более отрицательной амплитудой формируются системным адаптером клавиатуры, а с менее отрицательной - клавиатурой. После выполнения программы в регистрах были зафиксированы следующие коды: в AH - FAh, в DL - ABh, в DH - 41h.

7. Рекомендуемая литература

При подготовке к лабораторной работе и к ее защите рекомендуется пользоваться информацией, представленной в описании лабораторной работы, в разделе 1.2. настоящего учебно-методического пособия и в дополнительной литературе [5, 6, 8].

8. Контрольные вопросы

1. Охарактеризуйте клавиатуру и принципы ее взаимодействия с системой.
2. Охарактеризуйте три основных типа клавиатур ПК семейства PC.
3. Охарактеризуйте возможности 101/102-клавишной клавиатуры и ее развитие.
4. Охарактеризуйте основные группы клавиш 101/102-клавишной клавиатуры.
5. Что входит в состав модуля управления клавиатурой?
6. Охарактеризуйте конструктивные компоненты 101/102-клавишной клавиатуры.
7. Дайте описание блок-схемы клавиатуры (Рис. 1.14).
8. Какие типы датчиков клавиш могут применяться в клавиатурах?
9. Каковы функции устройства управления клавиатурой (контроллера)?
10. Какими типами данных обмениваются клавиатура и система?
11. Охарактеризуйте буфер клавиатуры, организованный в устройстве управления клавиатурой (в контроллере).
12. Как устройство управления клавиатурой (контроллер) обрабатывает состояние клавиш?
13. Дайте краткую характеристику компонентам электрической схемы 101-клавишной клавиатуры фирмы Xring.
14. Охарактеризуйте контрольные точки электрической схемы клавиатуры.
15. Какой тип клавиатуры может принимать команды от системы?
16. Можно ли подключить USB-клавиатуру к разъему PS/2-клавиатуры?
17. Кратко охарактеризуйте интерфейс 101-клавишной клавиатуры.
18. Кратко опишите взаимодействие 101-клавишной клавиатуры с системой.
19. Как взаимодействовала XT-клавиатура с системой?
20. Как выполняется базовый тест клавиатуры (BAT)?
21. Охарактеризуйте использование линий синхронизации и данных интерфейса клавиатуры.
22. Кратко охарактеризуйте потоки данных между клавиатурой и системой.
23. Опишите особенности ввода данных с клавиатуры.
24. Опишите особенности передачи данных в клавиатуру.
25. Охарактеризуйте осциллограммы, представленные на рис. 1.19.
26. Охарактеризуйте программу, представленную в листинге KLEE1.TXT.

4. Лабораторная работа 3 (описание) "Программное взаимодействие с клавиатурой"

1. Цель работы

Целью лабораторной работы является ознакомление студентов с принципами взаимодействия системы с клавиатурой на уровне команд, функций BIOS и функций DOS, изучение набора команд, выполняемых клавиатурой и принципов программного управления режимами работы клавиатуры на уровне команд и функций BIOS.

2. Технические и программные средства , используемые в работе.

В работе используется персональный компьютер типа Pentium с 101-клавишной клавиатурой. ПК работает под управлением ОС Windows 95/98/Mellenium. Операционная система загружается в режиме DOS (Command prompt only) с оболочкой типа Norton Commander. Непосредственное программное взаимодействие с клавиатурой осуществляется в среде отладчика типа AFD.

3. Краткое описание средств и правил программного взаимодействия с клавиатурой.

Программное взаимодействие с универсальной клавиатурой возможно через порты 60h и 64h, через обработчик аппаратного прерывания INT 09h; посредством функций программных прерываний INT 15h и INT 16h, обрабатываемых BIOS, посредством функций программного прерывания INT 21h, обрабатываемого DOS и средствами стандартных прикладных программных интерфейсов, например Win32 API в среде современных ОС.

3.1. Работа с клавиатурой через порты ввода-вывода

Работа с клавиатурой через порты ввода-вывода является самым низкоуровневым методом доступа. Такой метод используется обычно только обработчиками прерываний INT 09h, INT 16h и драйверами клавиатуры. Обмен данными с клавиатурой осуществляется через порт 60h адаптера клавиатуры командами IN и OUT центрального процессора ПК. Команда IN используется для чтения поступающих с клавиатуры скан-кодов и кодов ответов клавиатуры, команда OUT - для передачи в клавиатуру команд и их атрибутов (параметров). При организации обмена порт 64h адаптера клавиатуры используется для синхронизации обмена и управления потом данных. Так, готовность адаптера к передаче байта в клавиатуру или к передаче полученного от клавиатуры байта в систему определяется состоянием разрядов 1 и 0 регистра состояния адаптера, доступного по чтению через порт 64h в любое время (см. раздел 1.1.4.1.и 1.1.8.). Запрет или разрешение работы интерфейса клавиатуры осуществляется командами ADh и AEh адаптера, передающимися в адаптер также через порт 64h (см. раздел 1.1.5.). Синхронизацию приема данных с клавиатуры можно осуществлять по прерыванию IRQ1 (INT 09h), генерируемому адаптером при установке в 1 разряда 0 регистра состояния адаптера (см. разделы 1.1.6. и 1.1.7.).

Ниже в листинге 1 приведен пример программы чтения скан-кода с использованием портов 60h и 64h:

Листинг 1. Чтение скан-кода на регистровом уровне.

```
2C47:0100  FA       CLI           ;запрет обработки маскируемых аппаратных
                                  ;прерываний
2C47:0101  E464     IN    AL,[64] ;проверка наличия данных в выходном буфере
2C47:0103  A801     TEST  AL,01   ;адаптера клавиатуры
2C47:0105  74FA     JZ    0101    ;
2C47:0107  E460     IN    AL,[60] ;чтение скан-кода из выходного буфера и
2C47:0109  88C3     MOV   BL,AL   ;пересылка его в регистр BLh
2C47:010B  E464     IN    AL,[64] ;отработка сигнала ACKk подтверждения приема
2C47:010D  A802     TEST  AL,02   ;скан-кода системой (см. предыдущий пример)
2C47:010F  75FA     JNZ   010B    ;Как правило эту операцию
2C47:0111  B0AD     MOV   AL,AD   ;адаптер клавиатуры
2C47:0113  E664     OUT   [64],AL ;выполняет автоматически
2C47:0115  E464     IN    AL,[64] ;своими программными
2C47:0117  A802     TEST  AL,02   ;средствами, удерживая линию синхронизации
2C47:0119  75FA     JNZ   0115    ;на низком уровне пока процессор не прочтет
2C47:011B  B0AE     MOV   AL,AE   ;содержимое порта 60
2C47:011D  E664     OUT   [64],AL ;
2C47:011F  FB       STI           ;разрешение обработки маскируемых аппаратных
                                  ;прерываний
2C47:0120  CC       INT3          ;возвращение управления отладчику
```

В данном фрагменте программы команда CLI блокирует работу обработчика прерываний клавиатуры INT 09h на время выполнения программы. Команда STI разрешает работу обработчика INT 09h. При запуске этого фрагмента программы в среде отладчика AFD в результате ее работы в регистр BL занесется код 9Ch - скан код отпускания клавиши "Enter", нажатием которой инициируется команда G запуска программы в среде AFD.

Как правило, блокировка интерфейса клавиатуры после приема очередного байта с клавиатуры удерживанием линии синхронизации на низком уровне осуществляется адаптером автоматически на время, пока система не прочитает поступивший байт из выходного буфера адаптера через порт 60h. Поэтому в выше проведенной программе можно исключить команды, реализующие формирование сигнала подтверждения приема скан-кода системой (команды с адреса 2C47:010B по 2C47:011D). Управление активностью интерфейса клавиатуры командами ADh и AEh может применяться в программах, которые должны блокировать клавиатуру на время выполнения определенного набора операций и т. п.

В PC/AT запись в порт 60h позволяет программировать клавиатуру, т.е. все, что помещается в порт 60h передается в клавиатуру (исключение составляют атрибуты команд адаптера клавиатуры, которые также записываются по адресу 60h). С помощью порта 64h можно читать состояние линий интерфейса клавиатуры, состояние клавиатуры и адаптера клавиатуры, состояние его портов, осуществлять его программирование (используются команды взаимодействия с адаптером клавиатуры, см. раздел 1.1.).

При программирование клавиатуры необходимо первоначально в порт 60h записать код команды, сделать задержку и затем записать байт данных в тот же порт 60h (список команд приведен ниже):

```
          MOV   AL,код команды  ;См. ниже
          OUT   [60],AL         ;Запись кода команды в порт
                                ;Если нужно посылать данные,
          MOV   CX, 2000h       ;то необходима задержка
 pause:   LOOP  pause           ;Задержка
          MOV   AL, байт данных ;Запись параметра (атрибута)
          OUT   [60], AL        ;команды
```

Приведенный фрагмент программы может работать только на ПК с тактовыми частотами процессора меньше 30 мГц. На современных ПК вместо программной задержки нужно контролировать поступление от клавиатуры кода подтверждения приема (код FAh) или результатов выполнения команды. Этот код и результаты доступны по чтению через порт 60h при появление 1 в нулевом разряде регистра состояния адаптера клавиатуры. Соответствующий фрагмент программы представлен в листинге 2.

Листинг 2. Установка индикаторов состояния клавиатуры.

```
2C47:0100  FA      CLI             ;Запрет маскируемых прерываний
2C47:0101  E464    IN     AL,[64]  ;Определение состояния входного
2C47:0103  A802    TEST   AL,02    ;буфера: 0 - свободен, 1 - занят
2C47:0105  75FA    JNZ    0101     ;
2C47:0107  B0ED    MOV    AL,ED    ;Пересылка в клавиатуру кода команды
2C47:0109  E660    OUT    [60],AL  ;"Установить индикаторы состояния" (код EDh)
2C47:010B  E464    IN     AL,[64]  ;Определение состояния выходного буфера
2C47:010D  A801    TEST   AL,01    ;адаптера: 0 - пуст, 1 - принят код с
2C47:010F  74FA    JZ     010B     ;клавиатуры
2C47:0111  E460    IN     AL,[60]  ;Чтение кода из выходного буфера и сравнение
2C47:0113  3CFA    CMP    AL,FA    ;его с кодом FAh - подтверждение ACK
2C47:0115  7512    JNZ    0129     ;если не FAh - выход по ошибке
2C47:0117  B007    MOV    AL,07    ;Код конфигурации - включить все индикаторы
2C47:0119  E660    OUT    [60],AL  ;пересылается в клавиатуру (код 07h)
2C47:011B  E464    IN     AL,[64]  ;Определение состояния выходного буфера
2C47:011D  A801    TEST   AL,01    ;адаптера: 0 - пуст, 1 - принят код с
2C47:011F  74FA    JZ     011B     ;клавиатуры
2C47:0121  E460    IN     AL,[60]  ;Чтение кода из выходного буфера и сравнение
2C47:0123  3CFA    CMP    AL,FA    ;его с кодом FAh - подтверждение ACK
2C47:0125  7502    JNZ    0129     ;если не FAh - выход по ошибке
2C47:0127  FB      STI             ;Разрешение маскируемых прерываний
2C47:0128  CC      INT3            ;Передача управления отладчику
2C47:0129  88C4    MOV    AH,AL    ;Пересылка ошибочного кода в регистр AH
2C47:012B  EBFA    JMP    0127     ;Переход на окончание программы
```

При выполнение этого фрагмента программы в среде отладчика на клавиатуре включаются все три индикатора состояния: Num Lock, Caps Lock и Scroll Lock.

Команды клавиатуры и их описание.

Команды передаются клавиатуре через входной буфер адаптера клавиатуры, записью их кода по адресу 60h. Ответные коды клавиатуры на команды помещаются адаптером в выходной буфер и доступны для

чтения по адресу 60h. При программирование клавиатуры на уровне ее команд необходимо соблюдать ряд правил:

- передавать код команды через входной буфер адаптера с адресом 60h можно только при наличие признака "Входной буфер пуст" (разряд D1 регистра состояния адаптера клавиатуры установлен в 0);
- ответные коды клавиатуры на команды можно читать из выходного буфера по адресу 60h только после установки адаптером признака "Выходной буфер полон" (разряд D0 регистра состояния адаптера клавиатуры установлен в 1);
- перед программированием клавиатуры необходимо запретить, а при завершение - разрешить обработку прерывания клавиатуры.

Список команд клавиатуры приведен в табл. 1., а ниже - их описание.

Таблица. 1. Команды клавиатуры

Код	Функция
EDh	Установка и сброс индикаторов состояния
EEh	Эхо (средство диагностирования)
EFh, F1h	Резерв холостые команды (NOP)
F0h	Выбрать альтернативный режим
F2h	Идентификатор расширенной клавиатуры
F3h	Установить частоту повторения клавиши
F4h	Разрешить клавиатуре передачу данных
F5h	Запрет по умолчанию
F6h	Установить условия по умолчанию
F7h - FAh	Установить все клавиши
FBh - FDh	Установить отдельные клавиши
FEh	Послать повторно
FFh	Сброс

"Установить индикаторы состояния" (EDh) - это двухбайтовая команда, которая изменяет состояние светодиодных индикаторов клавиатуры. После получения этой команды клавиатура прекращает сканирование, возвращает системе код ACK (FAh - код подтверждения) и ожидает от системы байта конфигурации (атрибута команды). Байт конфигурации определяет установку светодиодных индикаторов и имеет следующий формат:

7	6	5	4	3	2	1	0
Резерв					CL	NL	SL

Бит CL задаст значение индикатора CAPS LOCK (0 - выключен, 1 - включен).
Бит NL управляет значением индикатора NUM LOCK (0 -выключен, 1 - включен).
Бит SL задает значение индикатора SCROLL LOCK (0 - выключен, 1 - включен).
Зарезервированные биты 7-3 должны быть установлены в 0.

После получения байта конфигурации клавиатура устанавливает индикатор состояния, возвращает код ACK (FAh) и возобновляет сканирование. Если в то время, когда клавиатура ожидает байт конфигурации, поступает другая команда, выполнение команды "Установить индикаторы состояния" прекращается и выполняется новая команда.

"Эхо" (EEh) используется для диагностирования. После получения этой команды клавиатура возвращает ответ - "Эхо" (EEh). Обычно команда EEh используется при возникновении ошибок в работе клавиатуры. Посылка команды EEh после выполнения сброса клавиатуры позволяет определить, восстановился ли нормальный режим функционирования. Неправильный ответ на эхо-команду означает, что процессор клавиатуры "зациклился" и привести систему в нормальное состояние можно только отключением питания.

Выбрать альтернативный режим (F0h) : клавиатура подтверждает эту двухбайтовую команду, сбрасывает выходной буфер и повторение клавиши. После этого она готовится к приему байта конфигурации, который задает рабочий режим. Клавиатура подтверждает байт конфигурации, устанавливает новый рабочий режим и возобновляет сканирование.

Байт конфигурации должен содержать номер устанавливаемого режима клавиатуры (01h - режим 1; 02h - режим 2 и 03h - режим 3).

Когда система посылает байт конфигурации 00h, клавиатура отвечает кодом подтверждения и посылает байт конфигурации, указывающий системе текущий режим.

"Холостые операции" (EFh, F1h): в ответ на такую команду клавиатура посылает код повторной посылки (FEh), не предпринимая никаких других действий.

"Идентификатор расширенной клавиатуры" (F2h) - это команда, передаваемая клавиатуре при включении питания. Клавиатура отвечает кодом ACK, прекращает сканирование, передает два байта идентификатора расширенной клавиатуры: сначала код ABh, а затем код 83h (или другой, например 41h) и возобновляет сканирование клавиш.

"Установить частоту повторения клавиши" (F3h) - это двухбайтовая команда, которая изменяет значения задержки и частоты повторения клавиши. Схема работы этой команды аналогична схеме работы команды "Установить индикаторы состояния" (EDh).

Как только клавиатура получает команду "Установить частоту повторения клавиши", она прекращает сканирование и возвращает код ACK. После этого система посылает клавиатуре байт, в котором содержатся значения частоты повторения и времени задержки. Частота повторения показывает, сколько раз в секунду повторяется данная клавиша. Время задержки - это время, в течение которого клавиша должна удерживаться нажатой, прежде чем начнется функция повторения. Клавиатура возвращает код ACK и остается в предшествующем состоянии сканирования - разрешенном или запрещенном.

Для возобновления сканирования должна быть передана команда "Разрешить клавиатуру" (F4h).

Байт частоты повторения и времени задержки имеет следующий формат:

- частота повторения (число повторений в секунду) - разряды (4 - 0):

00000=30,0,	01000=15,0,	10000=7,5,	11000=3,7,
00001=26,6,	01001=13,3,	10001=6,7,	11001=3,3,
00010=24,0,	01010=12,0,	10010=6,0,	11010=3,0,
00011=21,8,	01011=10,9,	10011=5,5,	11011=2,7,
00100=20,0,	01100=10,0,	10100=5,0,	11100=2,5,
00101=18,4,	01101=9,2,	10101=4,6,	11101=2,3,
00110=17,1,	01110=8,6,	10110=4,3,	11110=2,1,
00111=16,0,	01111=8,0,	10111=4,0,	11111=2,0;

- время задержки (в миллисекундах) - разряды (6 - 5):

00=250,	01=500,	10=750,	11=1000.

Седьмой разряд всегда должен содержать нуль.

Команда ***"Разрешить клавиатуру" (F4h)*** разрешает клавиатуре начать сканирование и передачу данных. После получения этой команды клавиатура возвращает код ACK, сбрасывает выходной буфер и начинает сканирование.

"Запрет по умолчанию" (F5h) останавливает сканирование клавиатуры. После получения этой команды клавиатура возвращает системе код ACK (подтверждение) и:

- устанавливает частоту повторения клавиши и время задержки, принятые по умолчанию;
- прекращает сканирование;
- очищает буфер клавиатуры;
- устанавливает типы клавиш по умолчанию (только для режима 3);
- сбрасывает последнюю повторяемую клавишу;
- ожидает дальнейших команд.

По команде ***"Установить условия по умолчанию" (F6h)*** клавиатура посылает код ACK и:

- устанавливает частоту повторения клавиши и время задержки, принятые по умолчанию;
- очищает буфер клавиатуры;
- устанавливает типы клавиш по умолчанию (только для режима 3);
- сбрасывает последнюю повторяемую клавишу;
- продолжает сканирование.

По команде ***"Установить все клавиши" (F7h, F8h, F9h, FAh)*** клавиатура посылает подтверждение получения команды, очищает выходной буфер, устанавливает все клавиши так, как это указано командой (табл. 1.11), и продолжает сканирование, если оно до этого было разрешено. Эти команды используются только **в режиме 3.**

Таблица 2. Назначение команд установки клавиш

Команда	Действие всех клавиш
F7h	Только Повторение
F8h	Только Нажатие/Отпускание
F9h	Только Нажатие
FAh	Повторение/Нажатие/Отпускание

По команде ***"Установить отдельные клавиши" (FBh, FCh, FDh)*** клавиатура подтверждает получение этой двухбайтовой команды, устанавливает команду кода отдельной клавиши, очищает выходной буфер, прекращает сканирование и ожидает байт идентификации клавиши. Идентификация клавиши выполняется с помощью кода сканирования позиции клавиши, приведенного в табл. 3. Клавиатура подтверждает байт идентификации клавиши, устанавливает новый тип клавиши и ожидает следующей команды. Клавиатура не возобновляет сканирование, пока система не позволит сделать это посредством

команды "Разрешить клавиатуру" (F4h), что дает возможность одновременной установки нескольких клавиш. Эти команды используются только в **режиме 3.**

Таблица 3. Назначение команд установки отдельных клавиш

Команда	Действие всех клавиш
FBh FCh FDh	Только Повторение Только Нажатие/Отпускание Только Нажатие

Команда ***"Послать повторно" (FEh)*** выдается только после передачи данных с клавиатуры. Клавиатура отвечает повторной передачей последнего байта, переданного системе. Если последний переданный байт был RESEND (FEh), клавиатура передает предшествующий ему байт.

По команде *"Сброс" (FFh)* клавиатура возвращает системе код ACK. Для того чтобы принять этот код, система возбуждает линии CLOCK и DATA по меньшей мере на 500 мкс. После передачи кода ACK клавиатура выполняет внутренний тест (BAT) и посылает в систему код завершения теста - значение AAh, если тест завершился удачно, и FDh, если произошла ошибка.

Коды ответов клавиатуры

Клавиатура отвечает системе кодами, описание которых приведено ниже.

Переполнение/ошибка клавиши (00h или FFh). Для указания переполнения буфера клавиатура помещает 00h в буфер в качестве последнего символа для кодов сканирования в режимах 2 и 3. Для кодов сканирования в режиме 1 используется FFh.

Клавиатура передает код ошибки клавиши, если не удается идентифицировать замыкание контактов на клавиатуре. Для кодов сканирования в режиме 1 передается FFh, а в режимах 2 и 3 - 00h.

Подтверждение (FAh). Клавиатура выдает подтверждение (ACK) в ответ на любую правильную команду, кроме команд "Эхо" и RESEND. Если клавиатура прерывается во время передачи ACK, то ACK отменяется и клавиатура начинает обработку новой команды.

Код завершения включения питания (AAh). После успешного завершения включения питания клавиатура передает системе код AAh. Передача любого другого кода после включения указывает на неисправность клавиатуры.

Эхо (EEh). Клавиатура передает этот код в ответ на команду "Эхо".

Идентификатор клавиатуры (83ABh). Идентификатор клавиатуры состоит из двух байтов - 83ABh. Клавиатура отвечает на команду "Идентификатор расширенной клавиатуры" подтверждением, прерывает сканирование и передает два байта идентификатора. Первым передается младший байт. После вывода идентификатора клавиатура начинает сканирование.

Повторная посылка (FEh). Клавиатура передает этот код после получения неправильных входных данных или данных с ошибкой паритета.

3.2. Работа с клавиатурой через прерывание INT 09h

Прерывание INT 09h является аппаратным прерыванием (IRQ1). Обработчик этого прерывания обрабатывает нажатия и отпускания клавиш клавиатуры на уровне BIOS. Принятое с клавиатуры значение скан-кода и соответствующее ему значение ASCII кода обработчик прерывания INT 09 сохраняет в буфере клавиатуры. Буфер расположен в памяти рабочей области BIOS по адресам с 0040:001A по 0040:003C. Обработчик также устанавливает флажки состояния статуса клавиш-переключателей (Shift, Ctrl, Caps Lock и т. д.) в ячейках 0040:0017 и 0040:0018. Ячейка 0:0419 используется как аккумулятор кода Alt-набора.

Буфер клавиатуры может накапливать до 15 нажатий на клавишу независимо от того, являются они однобайтовыми кодами ASCII или двухбайтовыми расширенными кодами. Таким образом в буфере отводится два байта памяти для каждого нажатия на клавишу. Для однобайтовых кодов первый байт содержит код ASCII, а второй - скан-код клавиши. Для расширенных кодов первый байт содержит код ASCII 0, а второй - номер расширенного кода. Этот код обычно совпадает со скан-кодом клавиши, но не всегда, поскольку некоторые клавиши можно комбинировать с клавишами сдвига для генерации различных кодов.

Буфер устроен, как циклическая очередь, которую называют также буфером FIFO (первый вошел - первый ушел). Он занимает непрерывную область адресов памяти. Однако определенной ячейки памяти, которая хранит "начало строки" в буфере, нет. Вместо этого два указателя хранят позиции головы (0040:001A) и хвоста (0040:001C) строки символов, находящейся в буфере в текущий момент. Новые нажатия клавиш сохраняются в позициях, следующих за хвостом (в более старших адресах памяти), и соответственно обновляется указатель хвоста буфера. После того как израсходовано все буферное пространство, новые символы продолжают вставляться, начиная с самого начала буферной области (0040:001E); поэтому возможны ситуации, когда голова строки в буфере имеет больший адрес, чем хвост. Когда буфер заполнен, новые вводимые символы игнорируются, при этом обработчик прерывания клавиатуры выдаст гудок через динамик.

В то время как указатель на голову строки установлен на первый введенный символ, указатель на хвост

установлен на позицию за последним введенным символом. Когда оба указателя равны, буфер пуст. Чтобы разрешить ввод 15 символов, требуется 16-я пустая позиция, 2 байта которой всегда содержат код возврата каретки (ASCII 13 (0Dh)) и скан-код клавиши «Enter», равный 28 (1Ch). Эта пустая позиция непосредственно предшествует голове строки символов. 32 байта буфера начинаются с адреса 0040:001E. Указатели на голову и хвост расположены по адресам 0040:001A и 0040:001C соответственно. Хотя под указатели отведено 2 байта, используется только младший байт. Значения указателей меняются от 30 (1Eh) до 60 (3Ch), что соответствует позициям буфера клавиатуры в области данных BIOS. Для очистки буфера надо просто установить значение ячейки 0040:001A равным значению ячейки 0040:001C.

Отметим, что программа может вставлять символы в буфер, завершая строку символом возврата каретки и меняя значения указателей. Если это сделано перед завершением программы, то при возврате управления в MS-DOS эти символы будут считаны и может быть автоматически загружена другая программа.

В связи с такой организацией буфера не рекомендуется вызывать прерывание INT 09h непосредственно из пользовательских программ, поскольку это может привести к непредсказуемым последствиям.

Назначение флажковых разрядов, определяющих статус клавиш-переключателей и устанавливаемых обработчиком INT 09h в ячейках памяти 0040:0017 и 0040:0018, приведено в таблице 4.

Таблица 4. Назначение флажковых разрядов

Адрес	Бит	Клавиша	Значение, когда бит = 1
0040:0017	0	Правый Shift	Клавиша нажата
	1	Левый Shift	Клавиша нажата
	2	Любой Ctrl	Клавиша нажата
	3	Любой Alt	Клавиша нажата
	4	Scroll Lock	Режим Scroll Lock включен
	5	Num Lock	Режим Num Lock включен
	6	Caps Lock	Режим Caps Lock включен
	7	Insert	Режим вставки включен
0040:0018	0	Левый Ctrl	Клавиша нажата
	1	Левый Alt	Клавиша нажата
	2	Правый Ctrl	Клавиша нажата
	3	Правый Alt	Клавиша нажата
	4	Scroll Lock	Клавиша нажата
	5	Num Lock	Клавиша нажата
	6	Caps Lock	Клавиша нажата
	7	SysReq	Клавиша нажата

Перехват прерывания INT 09h можно использовать в резидентных программах для работы с кодами национальных шрифтов или в программах, построенных на интенсивной обработке данных с клавиатуры (компьютерные игры). Как правило, обработка клавиш в INT 09h выполняется с помощью портов 60h/61h/64h, (порт 61h используется в PC/XT) как рассмотрено выше.

Стандартный INT 09h вызывает следующие прерывания:

- INT 05h после нажатия клавиши Prt Sc для печати копии экрана;
- INT 15h (подфункция 4Fh) для альтернативной обработки нажатия (отпускания) клавиши. Эта подфункция вызывается (для 83- и 101/102-клавишных клавиатур) перед обработкой прочитанного кода клавиши. При вызове INT 15h/4Fh в AL передается скан-код клавиши и устанавливается CF; если клавиша обработана альтернативным обработчиком, то после завершения подфункции должен быть сброшен CF (иначе значения регистров не должны измениться);
- INT 15h (подфункция 85h) для обработки клавиши Sys Req с входными параметрами: AH = 85h (номер подфункции), AL = 00h (если Sys Req нажата) или 01h (если Sys Req отжата). Обычно эта подфункция никак не задействована в BIOS и используется прикладными программами;
- INT 15h (подфункция 91h - "прерывание завершено") со значением AL=02h (клавиатура).
- и ряд других.

3.3. Работа с клавиатурой через прерывание INT 16h (драйвер клавиатуры)

Программно вызываемое прерывание BIOS INT 16h представляет интерфейс прикладного уровня для клавиатуры. Его основное назначение — извлечение слов из клавиатурного буфера. Функция задается в регистре AH при вызове:

- AH = 00h — чтение (с ожиданием готовности) и выборка слова из буфера;
- AH = 01h — проверка готовности, чтение без выборки;
- AH = 02h — чтение состояния флагов;

- AH = 03h — установка задержки и скорости автоповтора;
- AH = 05h — запись слова в буфер клавиатуры.

Функции *с* AH = 10h, 11h, 12h аналогичны 00h, 0lh, 02h, но предназначены специально для 101/102-клавишных клавиатур. Для ряда клавиш, отсутствующих в клавиатуре AT-84, эти функции дадут результаты, отличающиеся от результатов вызовов 00h, 0lh и 02h.

Функции с AH = 20h, 21h и 22h используются для дополнительных клавиш 122-клавишной клавиатуры.

Ниже приводится описание функций 00 - 05 и 10 - 12.

Чтение данных с клавиатуры с ожиданием нажатия Реально эта функция берет данные из буфера клавиатуры, поэтому с ее помощью нельзя определить нажатие таких клавиш, как Alt, Shift и т.п.

Вход: AH = 00h.
Выход: AL - ASCII-код клавиши,
AH - скан-код клавиши.

Проверка клавиатуры. Эта подфункция проверяет, есть ли в буфере символ (код клавиши). При наличии символа возвращает его значение, но не очищает буфер клавиатуры.

Вход: AH = 01h.
Выход: ZF=ZR=1, если в буфере нет символа;
ZF=ZR=0, если в буфере есть символ, при этом
AL - ASCII-код клавиши,
AH - скан-код клавиши.

Прочитать статус сдвигов. Возвращает значение байта с адресом 0040:0017h.

Вход: AH = 02h.
Выход: AL - статус сдвигов:
разряд 0 - 1, если правая клавиша Shift нажата,
разряд 1 - 1, если левая клавиша Shift нажата,
разряд 2 - 1, если клавиша Ctrl (с любой стороны) нажата,
разряд 3 - 1, если клавиша Alt (с любой стороны) нажата,
разряд 4 - 1, если активен режим Scroll Lock,
разряд 5 - 1, если активен режим Num Lock,
разряд 6 - 1, если активен режим Caps Lock,
разряд 7 - 1, если активен режим Insert.

Установка скорости генерации символов повтора и паузы. Не используется с клавиатурой PC XT.

Вход: AH = 03h,
AL = 05h,
BL - скорость генерации символов (0 - быстрее, чем 1Fh),
BH - задержка (0-0,25 с, 1-0,5 с, 2-0,75 с, 3-1 с).
Выход: нет.

Записать символ в буфер клавиатуры. Не используется с клавиатурой PC XT.

Вход: AH = 05h,
CL - ASCII-код символа,
CH - скан-код или нуль, если скан-код безразличен.
Выход: AL = 0, если операция записи завершилась успешно.

Чтение данных с клавиатуры с ожиданием нажатия (для 101-клавишной клавиатуры). Эта подфункция является как бы расширением подфункции 00h.

Вход: AH = 10h.
Выход: AL - ASCII-код клавиши,
AH - скан-код клавиши.

Проверка клавиатуры (для 101-клавишной клавиатуры) Эта подфункция является расширением подфункции 01h.

Вход: AH = 11h.
Выход: ZF=ZR=1, если в буфере нет символа;
ZF=ZR=0, если в буфере есть символ, при этом
AL - ASCII-код клавиши,
AH - скан-код клавиши.

Прочитать статус сдвигов 101-клавишной клавиатуры

Вход: AH = 12h.
Выход: AL - статусы, совпадающие с результатами работы функции 02h
AH - статус сдвигов дополнительных клавиш 101-клавишной клавиатуры:
разряд 0 - 1, если левая клавиша "Ctrl" нажата,
разряд 1 - 1, если левая клавиша "Alt" нажата,

разряд 2 - 1, если правая клавиша "Ctrl" нажата,
разряд 3 - 1, если правая клавиша "Alt" нажата,
разряд 4 - 1, если клавиша "Scroll Lock" нажата,
разряд 5 - 1, если клавиша "Num Lock" нажата,
разряд 6 - 1, если клавиша "Caps Lock" нажата,
разряд 7 - 1, если клавиша "Sys Req" нажата.

Младший байт считанного функциями 00h, 10h и 20h слова содержит *ASCII-символ,* старший — *скан-код*. Символы, полученные нестандартным способом (в русском регистре или Alt-набором), сопровождаются нулевым скан-кодом. При нулевом младшем байте старший содержит *расширенный ASCII-код* (Extended ASCII Keystroke). Дополнительные клавиши 101/102 клавиатур при использование функций 10h-11h генерируют код E0h в младшем байте и скан-код, соответствующий аналогичным управляющим клавишам 83/84-клавишных клавиатур.

Функция записи (05h), несколько неожиданная для клавиатуры, позволяет легко имитировать работу оператора для различных демонстрационных программ. Если прикладная программа не перехватывает обслуживание клавиатуры на уровне аппаратного прерывания (INT 9h), то резидентная программа может ей «подбрасывать» слова в буфер, которые будут восприниматься как нажатие клавиш.

3.4. Функции DOS обработки клавиатуры

Эти функции (подфункции INT 21h) обычно используются программами, которые должны обеспечивать максимальную совместимость со всеми типами ПК, самой операционной системой и ее утилитами. В DOS клавиатура обычно назначается стандартным устройством ввода и ей присваивается указатель 0. Если средствами DOS стандартное устройство ввода переназначено (на файл или последовательный порт), клавиатура становится недоступной для работы через INT 21h.

Ожидание поступления и чтение символа со стандартного устройства ввода с эхом на стандартное устройство вывода и обработкой Ctrl Break.

Вход: AH = 01h.
Выход: AL - код символа, клавиши типа F11 необходимо обрабатывать за два вызова.

Чтение со стандартного устройства ввода (без ожидания) или запись на стандартное устройство вывода.

Нажатие Ctrl Break не обрабатывается.
Вход: AH = 06h;
DL = FFh, если необходимо прочитать символ,
DL = 0-FEh, если необходимо вывести этот код.
Выход: если ZF сброшен, то символ был прочитан и его код хранится в AL (при вызове с DL=FFh).

Ожидание поступления и чтение символа со стандартного устройства ввода без эха и обработки Ctrl Break

Вход: AH = 07h.
Выход: AL - код символа, клавиши типа F11 необходимо обрабатывать за два вызова.

Ожидание поступления и чтение символа со стандартного устройства ввода без эха, но с обработкой Ctrl Break

Вход: AH = 08h.
Выход: AL - код символа, клавиши типа F11 необходимо обрабатывать за два вызова.

Буферизированный ввод со стандартного устройства ввода

Эта подфункция обеспечивает чтение строки символов с несложным редактированием вводимой строки и обработкой нажатия Ctrl Break.

Вход: AH = 0Ah;
DS:DX - адрес для записи вводимой строки (в первом байте - максимальная длина строки).
Выход: в буфер занесена строка (во втором - байте длина прочитанной строки).

Проверить стандартное устройство ввода

Обрабатывается нажатие Ctrl Break.
Вход: AH = 0Bh.
Выход: AH = 00h, если есть символ,
AH = FFh, если символа нет.

Чистка буфера стандартного устройства ввода и чтение символа с помощью указанной функции

Если указан не допустимый номер функции, то буфер будет очищен, но функция не вызывается.
Вход: AH = 0Ch;
AL - номер функции DOS (01h, 06h, 07h, 08h, 0Ah).
Выход: в соответствии с вызываемой подфункцией.

Прочитать данные с клавиатуры можно также посредством функций DOS 3Fh и 40h.

Информацию о скан-кодах и ASCII-кодов, используемых в клавиатурах можно найти в разделах 1.2.5.6. и 1.2.6.4.9.настоящего учебного пособия. Примеры программирования клавиатуры представлены в разделе 1.2.7.

4. Указания по подготовке к лабораторной работе.

При подготовке к лабораторной работе, ее выполнении и оформлении отчета необходимо следовать рекомендациям раздела 4.1."Общие указания по выполнению и оформлению лабораторных работ".

Подготовка к данной лабораторной работе заключается в изучении описания лабораторной работы "Программное взаимодействие с клавиатурой", разделов 1.2.5.5. - 1.2.6.5 и раздела 1.2.7 учебного пособия. Необходимо также ознакомиться с основными командами отладчика AFD и с командами процессора i8086.

5. Порядок выполнения лабораторной работы.

1. Ознакомиться с ниже приведенными программами.
2. В окне дизассемблера отладчика AFD набрать программу 1, приведенную в разделе 6 описания, проверить и запустить ее командой "G" отладчика.
3. Проанализировать результат, а затем модифицировать программу так, чтобы она включила только светодиоды указанные преподавателем.
4. Взяв за основу программу 1 из раздела 6 описания установить сначала максимальную скорость повторения нажатой клавиши с задержкой 1с, а затем минимальную с задержкой 0.5с. Результаты проверять нажатием клавиш клавиатуры. Установить скорость и задержку указанные преподавателем.
5. Взяв за основу программу 2 раздела 6 описания работы установить указанные в пункте 4 параметры скорости повторения нажатой клавиши и задержки с помощью функций драйвера клавиатуры INT 16h.
6. Используя программу 3 раздела 6 определить системные коды клавиш основной, функциональной и цифровой клавиатур указанных преподавателем (по 5 клавиш каждой из клавиатур). Сравнить эти коды с кодами, представленными в табл. 1.4 и 1.5 раздела 1.2.5.6. учебного пособия.
7. Используя программу 4 раздела 6 получить скан- и ASCII-коды, клавиш, указанных преподавателем и сравнить их с кодами, представленными в табл. 1.4 и 1.5 раздела 1.2.5.6. и в табл. 1.14 - 1.20 раздела 1.2.6.4.9. учебного пособия.
8. Используя программу 5 раздела 6 описания получить значения скан-кодов указанных преподавателем клавиш, при работе клавиатуры в режимах 1, 2 и 3 и сравнить их с представленными в разделах 1.2.5.6 и 1.3.6.4.9 учебного пособия.
9. Получить у преподавателя контрольное задание и выполнить его.

6. Примеры программ

Программа 1, представленная в листинге 3, аналогична программе, приведенной в разделе 1.2.6.2.1. Она позволяет управлять индикаторами состояния клавиатуры Num Lock, Caps Lock и Scroll Lock посылая в клавиатуру через порт 60h соответствующую двухбайтовую команду клавиатуры (код EDh команды " Установка и сброс индикаторов состояния" и ее атрибут 07h). В результате выполнения программы в клавиатуре включаются все три индикатора (светодиода). Описание команды EDh представлено в разделе 1.2.6.2.2.пособия. В листинге все команды программы достаточно подробно прокомментированы. Программа выполняется в среде отладчика AFD.

Листинг 3. Программа 1 установки индикаторов клавиатуры (файл KL3.TXT).

```
2BF9:0100  FA      CLI             ;Запрет прерываний.
2BF9:0101  E464    IN      AL,[64] ;Ожидание освобождения
2BF9:0103  A802    TEST    AL,02   ;входного буфера
2BF9:0105  75FA    JNZ     0101    ;адаптера клавиатуры.
2BF9:0107  A801    TEST    AL,01   ;Проверка: пуст ли вых. буфер,
2BF9:0109  7402    JZ      010D    ;если пуст, то далее, иначе чтение
2BF9:010B  E460    IN      AL,[60] ;выходного буфера (очичтка).
2BF9:010D  B0ED    MOV     AL,ED   ;Команда EDh посылается в клавиатуру
2BF9:010F  E660    OUT     [60],AL ;через порт 60h.
2BF9:0111  E464    IN      AL,[64] ;Ожидание получения
2BF9:0113  A801    TEST    AL,01   ;кода ответа
2BF9:0115  74FA    JZ      0111    ;от клавиатуры.
2BF9:0117  E460    IN      AL,[60] ;Чтение кода ответа и
```

```
2BF9:0119  3CFA    CMP    AL,FA     ;сравнение его с FAh,
2BF9:011B  7512    JNZ    012F      ;если не FAh, то выход по ошибке.
2BF9:011D  B007    MOV    AL,07     ;Пересылка в клавиатуру атрибута
2BF9:011F  E660    OUT    [60],AL   ;команды EDh.
2BF9:0121  E464    IN     AL,[64]   ;Ожидание поступления
2BF9:0123  A801    TEST   AL,01     ;кода ответа
2BF9:0125  74FA    JZ     0121      ;от клавиатуры.
2BF9:0127  E460    IN     AL,[60]   ;Чтение кода ответа и
2BF9:0129  3CFA    CMP    AL,FA     ;сравнение его с FAh,
2BF9:012B  7502    JNZ    012F      ;если не FAh, то выход по ошибке.
2BF9:012D  FB      STI              ;Разрешение прерываний.
2BF9:012E  CC      INT3             ;Возврат в отладчик.
2BF9:012F  88C4    MOV    AH,AL     ;Ошибочный код в регистр AH и
2BF9:0131  EBFA    JMP    012D      ;выход из программы.
```

Программа 2, представленная в листинге 4, устанавливает скорость повторения нажатой клавиши 10 раз в секунду с задержкой 1с с помощью функции 03 прерывания INT 16h, обслуживаемого BIOS, то есть выполняет те же действия что и команда EDh клавиатуры. Следует отметить, что команда EDh может быть выполнена программой 1, если в ней код команды F2h кодом команды EDh, а атрибут 07h на атрибут 6Ch. . Описание функций драйвера INT 16h приведено в разделе 1.2.6.4. пособия.

Листинг 4. Программа 2, установки скорости повтора нажатой клавиши и задержки повтора с использованием INT 16h (файл KL4.TXT).

```
2BF9:0100  B403    MOV    AH,03     ;Номер функции
2BF9:0102  B005    MOV    AL,05     ;Подфункция.
2BF9:0104  B30C    MOV    BL,0C     ;Константа скорости.
2BF9:0106  B703    MOV    BH,03     ;Константа задержки.
2BF9:0108  CD16    INT    16        ;Вызов драйвера.
2BF9:010A  CC      INT3             ;Возврат в отладчик.
```

Сравнивая тексты программ 1 и 2 видно, что для выполнения схожих функций использование драйвера значительно сокращает число команд программы. Однако драйвер не реализует функции остальных команд клавиатуры, представленных в разделе 1.2.6.2.2. пособия, в частности функции команды EDh, реализованные в программе 1.

Программа 3, представленная в листинге 5 позволяет принимать с 101-клавишной клавиатуры через адаптер клавиатуры системные скан-коды клавиш основной, функциональной и цифровой клавиатур. Клавиатура работает в режиме 2 и в адаптере клавиатуры разрешены процедуры конвертирования скан-кодов режима 2 в системные скан-коды (коды режима 1). Принимаемые коды размещаются в буфере памяти начиная с адреса ES:0000h. Буфер рассчитан на 11 байт, что позволяет разместить в нем коды нажатия и отпускания пяти клавиш и код отпускания клавиши "Enter", которой завершается запуск программы в среде AFD командой "G" отладчика.

Листинг 5. Программа 3 приема системных скан-кодов от клавиатуры через порт 60h адаптера клавиатуры (файл KL2.TXT).

```
2BF9:0100  FA      CLI              ;Запрет прерываний.
2BF9:0101  BF0000  MOV    DI,0000   ;Счетчик-индекс = 0.
2BF9:0104  E464    IN     AL,[64]   ;Проверка: есть ли в вых. буфера
2BF9:0106  A801    TEST   AL,01     ;байт данных,
2BF9:0108  74FA    JZ     0104      ;если нет, то новый цикл проверки.
2BF9:010A  E460    IN     AL,[60]   ;Чтение байта и запоминание его
2BF9:010C  268805  MOV    ES:[DI],AL ;в буфере памяти.
2BF9:010F  47      INC    DI        ;Инкремент регистра DI.
2BF9:0110  83FF0B  CMP    DI,000B   ;Если DI не равно Bh, то на новый цикл
2BF9:0113  75EF    JNZ    0104      ;чтения кодов с клавиатуры.
2BF9:0115  FB      STI              ;Разрешение прерываний.
2BF9:0116  CC      INT3             ;Переход в AFD.
```

Программа 4, представленная в листинге 6, позволяет принимать из 101-клавишной клавиатуры конвертированные адаптером клавиатуры системные скан-коды и получать соответствующие им ASCII-коды символов, закрепленных за клавишами в соответствии с установленной кодовой таблицей (см. раздел 1.2.6.4.9 учебного пособия). Получаемые коды размещаются в буфере памяти, начиная с адреса ES:0000h. Размер буфера 32 байта. После выполнения последней команды Int3 содержимое буфера отображается в окне памяти отладчика.

Листинг 6. Программа 4 приема системных скан-кодов от клавиатуры и получения соответствующего ASCII-кода (файл KL5.TXT).

```
2528:0100  BF0000   MOV    DI,0000 ;Счетчик-индекс = 0.
2528:0103  B410     MOV    AH,10   ;Номер функции.
2528:0105  CD16     INT    16      ;Вызов драйвера клавиатуры.
2528:0107  268825   MOV    ES:[DI],AH ;Помещение в буфер
2528:010A  47       INC    DI         ;памяти скан-кода
2528:010B  268805   MOV    ES:[DI],AL ; и ASCII-кода.
2528:010E  47       INC    DI      ;Инкремент DI.
2528:010F  F7C72000 TEST   DI,0020 ;Проверка счетчика-индекса.
2528:0113  74EE     JZ     0103    ;Если DI не равно 20h, новый цикл.
2528:0115  CC       INT3           ;Переход в AFD.
```

В программе используется функция 10h драйвера клавиатуры INT 16h BIOS. Для получения ASCII-кодов дополнительных клавиш 122-клавишной клавиатуры необходимо пользоваться функцией 20h (см. описание функций драйвера INT 16h, приведенное в разделе 1.2.6.4. пособия). Драйвер INT 16h пользуется результатами работы обработчика прерываний клавиатуры, который помещает их в кольцевой буфер клавиатуры, расположенный в памяти рабочей области BIOS (см. раздел 1.2.6.3 учебного пособия). Для работы с символами кириллицы необходимо предварительно загрузить программу-русификатор, которая перехватывает прерывания клавиатуры и возвращает в буфер клавиатуры ASCII-коды, соответствующие дополнительной (русифицированной) кодовой таблице. При работе в Windows 95/98/Me в режиме загрузки Command Prompt Only для русификации видеосистемы и клавиатуры необходимо в файл Autoexec.bat внести следующие строчки:

```
MODE CON CODEPAGE PREPARE=((866) C:\WINDOWS\COMMAND\EGA3.CPI)
MODE CON CODEPAGE SELECT=866
KEYB RU,,C:\WINDOWS\COMMAND\KEYBRD3.SYS
```

а в файл Config.sys внести строчки:

```
device=C:\WINDOWS\COMMAND\display.sys con=(ega,,1)
Country=007,866,C:\WINDOWS\COMMAND\country.sys
```

Программа 5, читающая скан-коды из выходного буфера адаптера клавиатуры (адрес 60h) и помещающая их в буфер, начинающийся с адреса DS:0000 и имеющий размер в 16 байт представлена в листинге 7. На время работы программы запрещается обработка аппаратных прерываний.

Программа использует:

- подпрограмму P3, разрешающую или запрещающую конвертирование скан-кодов в адаптере;
- подпрограмму P4, реализующую переключение режимов работы клавиатур
- подпрограмму P2, реализующую ожидание заполнения выходного буфера адаптера

Листинг 7. Программа чтения скан-кодов клавиатуры, работающей в режиме 1, 2 или 3

```
232E:016C  FA         CLI            ;Запрет маскируемых прерываний.
232E:016D  B301       MOV    BL,01   ;Запрет преобразования скан-кодов
232E:016F  B7BF       MOV    BH,BF   ;адаптером (бит 6 командн. байта
232E:0171  E8A4FF     CALL   0118    ;уст. в 0) (выполняет П/П P3).
232E:0174  B303       MOV    BL,03   ;Переключение клавиатуры в режим 3
232E:0176  E8D1FF     CALL   014A    ;подпрограммой (П/П) P4.
232E:0179  BF0000     MOV    DI,0000 ;Установка на начало буфера памяти.
232E:017C  8CD8       MOV    AX,DS   ;Содержимое сегментного регистра DS
232E:017E  8EC0       MOV    ES,AX   ;переписывается в регистр ES.
232E:0180  E885FF     CALL   0108    ;Поступил скан-код в вых. буфер?
232E:0183  E460       IN     AL,[60] ;Чтение скан-кода из вых. буфера
232E:0185  268805     MOV    ES:[DI],AL   ;и записьего в буф. памяти.
232E:0188  47         INC    DI      ;Установка на след. байт буфера.
232E:0189  F7C71000   TEST   DI,0010 ;Буфер памяти заполнен?
232E:018D  74F1       JZ     0180    ;если нет - чтение следующего кода.
232E:018F  B302       MOV    BL,02   ;Переключение клавиатуры в реж.2
232E:0191  E8B6FF     CALL   014A    ;подпрограммой P4.
232E:0194  B302       MOV    BL,02   ;Разрешение преобразования скан-
232E:0196  B740       MOV    BH,40   ;кодов в системные
232E:0198  E87DFF     CALL   0118    ;подпрограммой P3.
232E:019B  FB         STI            ;разрешение маскируемых прерываний.
232E:019C  CC         INT3           ;Передача управления отладчику AFD.
```

Подпрограмма P1. Ожидание очистки входного буфера адаптера клавиатуры.

```
232E:0100  E464       IN     AL,[64] ;Чтение регистра состояния.
232E:0102  A802       TEST   AL,02   ;Разряд 1 равен 0?
232E:0104  75FA       JNZ    0100    ;если нет, то новый цикл чтения
232E:0106  C3         RET            ;если да, то возврат из П/П.
```

```
232E:0107  90        NOP
```

Подпрограмма P2. Ожидание заполнения выходного буфера адаптера клавиатуры.

```
232E:0108  E464      IN     AL,[64] ;Чтение регистра состояния.
232E:010A  A801      TEST   AL,01   ;Разряд 0 равен 0?
232E:010C  74FA      JZ     0108    ;если да, то новый цикл чтения
232E:010E  C3        RET            ;если нет, то возврат из П/П.
```

Подпрограмма P21. Проверка положительного кода подтверждения клавиатуры FAh (ACK).

```
232E:010F  E460      IN     AL,[60] ;Чтение кода ответа клавиатуры.
232E:0111  3CFA      CMP    AL,FA   ;Сравнение с кодом FAh.
232E:0113  7501      JNZ    0116    ;Коды совпали?
232E:0115  C3        RET            ;если совпали - возврат из П/П.
232E:0116  CC        INT3           ;если не совпали - выход в AFD.
232E:0117  90        NOP
```

Подпрограмма P3. Изменение командного байта адаптера клавиатуры.

Перед вызовом подпрограммы необходимо задать входные параметры:
В младший байт регистра BX (BL) необходимо занести:

- 01h - если с текущим командным байтом выполняется логическая операция AND;
- 02h - логическая операция OR.

В старший байт регистра BX (BH) необходимо занести:

- для операции AND - байт с нулями в разрядах, которые необходимо обнулить в командном байте;
- для операции OR - ,байт с единицами в разрядах, которые необходимо установить в командном байте.

В подпрограмме используются вспомогательные подпрограммы:

- подпрограмма P1, реализующая процедуру ожидания очистки входного буфера адаптера;
- подпрограмма P2, реализующая процедуру ожидания заполнения выходного буфера адаптера.

```
232E:0118  E8E5FF    CALL   0100    ;Ожидание очистки вх. буфера.
232E:011B  B020      MOV    AL,20   ;Команда чтения командного байта (20h)
232E:011D  E664      OUT    [64],AL ;передается в адаптер.
232E:011F  E8E6FF    CALL   0108    ;Ожидание заполнения вых. буфера.
232E:0122  E460      IN     AL,[60] ;Чтение командного байта.
232E:0124  80FB01    CMP    BL,01   ;Операция "END"?
232E:0127  7407      JZ     0130    ;если "да", то переход на ее выполнение.
232E:0129  80FB02    CMP    BL,02   ;Операция "OR"?
232E:012C  7408      JZ     0136    ;если "да", то переход на ее выполнение,
232E:012E  EB18      JMP    0148    ;если "нет", то выход из П/П.
232E:0130  20F8      AND    AL,BH   ;Выполнение операции "END" и занесение
232E:0132  88C3      MOV    BL,AL   ;результата в регистр BL.
232E:0134  EB04      JMP    013A    ;Переход на запись нового командн. байта.
232E:0136  08F8      OR     AL,BH   ;Выполнение операции "OR" и занесение
232E:0138  88C3      MOV    BL,AL   ;результата в регистр BL.
232E:013A  B060      MOV    AL,60   ;Команда 60h записи командного байта
232E:013C  E664      OUT    [64],AL ;передается в адаптер через порт 64h.
232E:013E  E8BFFF    CALL   0100    ;Ожидание очистки входного буфера.
232E:0141  88D8      MOV    AL,BL   ;Значение нового командного байта
232E:0143  E660      OUT    [60],AL ;передается в адаптер через порт 60h.
232E:0145  E8B8FF    CALL   0100    ;Ожидание очистки входного буфера.
232E:0148  C3        RET            ;Выход из подпрограммы.
232E:0149  90        NOP
```

Подпрограмма P4 переключения режимов работы клавиатуры.

Перед вызовом подпрограммы необходимо задать входные параметры:
в регистр BL необходимо занести константу:

- 00h - чтение текущего режима работы клавиатуры;
- 01h - установить режим 1 (PC/XT);
- 02h - установить режим 2 (PC/AT-совместимые ПК);
- 03h - установить режим 3 (ПК типа PS/2).

```
        При BL=00h подпрограмма возвращает в регистре AL значение
        текущего режима работы клавиатуры.

        В подпрограмме используются вспомогательные подпрограммы:
                -  подпрограмма P1, реализующая процедуру ожидания очистки
                входного буфера адаптера;
                -  подпрограмма P2, реализующая процедуру ожидания заполнения
                выходного буфера адаптера;
                -  подпрограмма P21, реализующая процедуру проверки значения
                кода подтверждения клавиатуры FAh (ACK).

232E:014A  E8B3FF    CALL   0100    ;ожидание очистки входного буфера.
232E:014D  B0F0      MOV    AL,F0   ;Команда F0h установки режима клавиатуры
232E:014F  E660      OUT    [60],AL ;передается в клавиатуру через порт 60h.
232E:0151  E8B4FF    CALL   0108    ;Ожидание поступления кода в вых. буфер.
232E:0154  E8B8FF    CALL   010F    ;Чтение вых. буфера и сравнение с FAh.
232E:0157  88D8      MOV    AL,BL   ;Передача нового кода режима в клавиатуру
232E:0159  E660      OUT    [60],AL ;через порт 60h адаптера.
232E:015B  E8AAFF    CALL   0108    ;Ожидание поступления кода в вых. буфер.
232E:015E  E8AEFF    CALL   010F    ;Чтение вых. буфера и сравнение с FAh.
232E:0161  80FB00    CMP    BL,00   ;Код режима равен 0?
232E:0164  7505      JNZ    016B    ;если нет, то возврат из подпрограммы,
232E:0166  E89FFF    CALL   0108    ;если да, то чтение кода текущего режима
232E:0169  E460      IN     AL,[60] ;работы клавиатуры из вых. буфера.
232E:016B  C3        RET            ;Возврат из подпрограммы.
```

Программа запускается командой G 16C из командной строки отладчика AFD. Предварительно необходимо загрузить коды команд программы и подпрограмм по указанным в листинге адресам. В этих адресах содержимое регистра кодового сегмента (CS) определяет сам отладчик AFD при его запуске, а значения смещения должны быть такими, как указано в листинге 1.15.

7. Рекомендуемая литература.

При подготовке к лабораторной работе и к ее защите рекомендуется пользоваться информацией, представленной в описании лабораторной работы, в разделе 1.2. настоящего учебно-методического пособия и в дополнительной литературе [5, 6, 8, 13, 15].

8. Контрольные вопросы

1. Охарактеризуйте режим 1 101-клавишной клавиатуры.
2. Охарактеризуйте режим 2 101-клавишной клавиатуры.
3. Охарактеризуйте режим 3 101-клавишной клавиатуры.
4. Охарактеризуйте скан-коды 101-клавишной клавиатуры.
5. Скан-коды каких клавиш добавлены в 122-клавишной клавиатуре?
6. В чем заключается программирование клавиатуры PC/XT?
7. В чем заключается программирование 101-клавишной клавиатуры?
8. Как системой обрабатывается нажатие и отпускание клавиш?
9. Как обрабатываются специальные клавиши?
10. Охарактеризуйте особенности взаимодействия с клавиатурой PC/XT через порты 60h и 61h.
11. Охарактеризуйте особенности взаимодействия с 101-клавишной клавиатурой через порты 60h и 64h.
12. Какие правила необходимо соблюдать при программирование 101-клавишной клавиатуры на уровне ее команд?
13. Перечислите команды, поддерживаемые клавиатурой.
14. Охарактеризуйте команду клавиатуры EDh.
15. Охарактеризуйте команду клавиатуры EEh и F2h.
16. Охарактеризуйте команду клавиатуры F4h.
17. Охарактеризуйте коды ответов клавиатуры.
18. Где расположен буфер клавиатуры обработчика прерывания INT 9h?
19. Как организован буфер клавиатуры?
20. Охарактеризуйте назначение флаговых разрядов, устанавливаемых обработчиком INT 9h в ячейках памяти 0040:0017h и 0040:0018h.
21. Для чего используется перехват прерывания INT 9h?
22. Дайте общую характеристику программному прерыванию INT 16h.
23. Охарактеризуйте функции 00h, 10h и 20h прерывания INT 16h.
24. Охарактеризуйте функции 01h, 11h и 21h прерывания INT 16h.
25. Охарактеризуйте функции 02h, 12h и 22h прерывания INT 16h.
26. Охарактеризуйте функции 03h прерывания INT 16h.

27. Охарактеризуйте функции 05h прерывания INT 16h.
28. Охарактеризуйте ASCII-коды основной клавиатуры.
29. Охарактеризуйте ASCII-коды цифровой клавиатуры.
30. Охарактеризуйте ASCII-коды функциональных клавиш.
31. Охарактеризуйте ASCII-коды Alt-комбинаций символьных клавиш.
32. Охарактеризуйте ASCII-коды дополнительной клавиатуры.
33. Охарактеризуйте расширенные ASCII-коды цифровой клавиатуры.
34. Охарактеризуйте расширенные ASCII-коды комбинаций клавиш.
35. Охарактеризуйте подфункции обработки клавиатуры прерыванием INT 21h DOS.
36. Охарактеризуйте программу, представленную в листинге 1.8.
37. Охарактеризуйте программу, представленную в листинге 1.9.
38. Что выполняет процедура GetChar входящая в состав модулей, представленных в листинге 1.9?
39. Охарактеризуйте информацию, представленную в листинге 1.10.
40. Охарактеризуйте макросы, представленные в листинге 1.11.
41. Охарактеризуйте программу, представленную в листинге 1.12.
42. Охарактеризуйте программный модуль, представленный в разделе 1.2.7.1.5.
43. Когда возникает реальная необходимость в непосредственной работе с клавиатурой через порты 60h и 64h?
44. Охарактеризуйте программу, представленную в листинге 1.13.
45. Охарактеризуйте программу, представленную в листинге 1.14.
46. Охарактеризуйте программу, представленную в разделе 1.2.7.2.3.

5. Лабораторная работа 4 (описание) "Манипулятор PS/2 Mouse".

1. Цель работы.

Целью лабораторной работы является ознакомление студентов с принципами работы манипулятора PS/2 Mouse; изучение функционирования электрической схемы манипулятора с регистрвцией сигналов в контрольных точках схемы с помощью электронного осциллографа; изучение сигнального протокола интерфейса манипулятора а также используемых PS/2 Mouse форматов передачи данных.

2. Технические и программные средства , используемые в работе.

В работе используется персональный компьютер с оптическим манипулятором 3D USB/PS/2 Mouse (M/N:M-BJ69 или M/N:MBJ58 фирмы Logitech), подключенным к порту PS/2 через переходник USB - PS/2 и имеющим зондовое устройство доступа к контрольным точкам его электрической схемы (ЗУДКТ) (см. рис. 1 и рис. 2). Могут использоваться и другие оптические или обычные манипуляторы PS/2 Mouse с соответствующими ЗУДКТ и его описанием. ПК работает под управлением ОС Windows 95/98/Mellenium. Операционная система загружается в режиме DOS (Command prompt only) с оболочкой типа Norton Commander. Непосредственное низкоуровневое программное взаимодействие с манипулятором осуществляется в среде отладчика типа AFD.

В работе используется также электронный осциллограф с полосой регистрации 0 - 10 МГц и выше.

3. Краткое описание оптического манипулятора 3D USB/PS/2 Mouse и его интерфейса.

На рис.1 представлен внешний вид используемого в лабораторной работе оптического манипулятора 3D USB/PS/2 Mouse (M/N:M-BJ69) фирмы Logitech (1). Манипулятор подключается к системному блоку ПК интерфейсным кабелем. В манипулятор вмонтировано зондовое устройство доступа к контрольным точкам его электрической схемы (ЗУДКТ) (3). В состав ЗУДКТ входит контактное поле доступа к контрольным точкам, представленное разъемом (4) типа ГРПМ1-31ГО2-В, матрицы (5) токоограничивающих резисторов номиналом 680 Ом и плоский кабель (2), подключенный своими проводниками-зондами к контрольным точкам электрической схемы с одной стороны, и к матрицам токоограничивающих резисторов - с другой (см. рис 2).

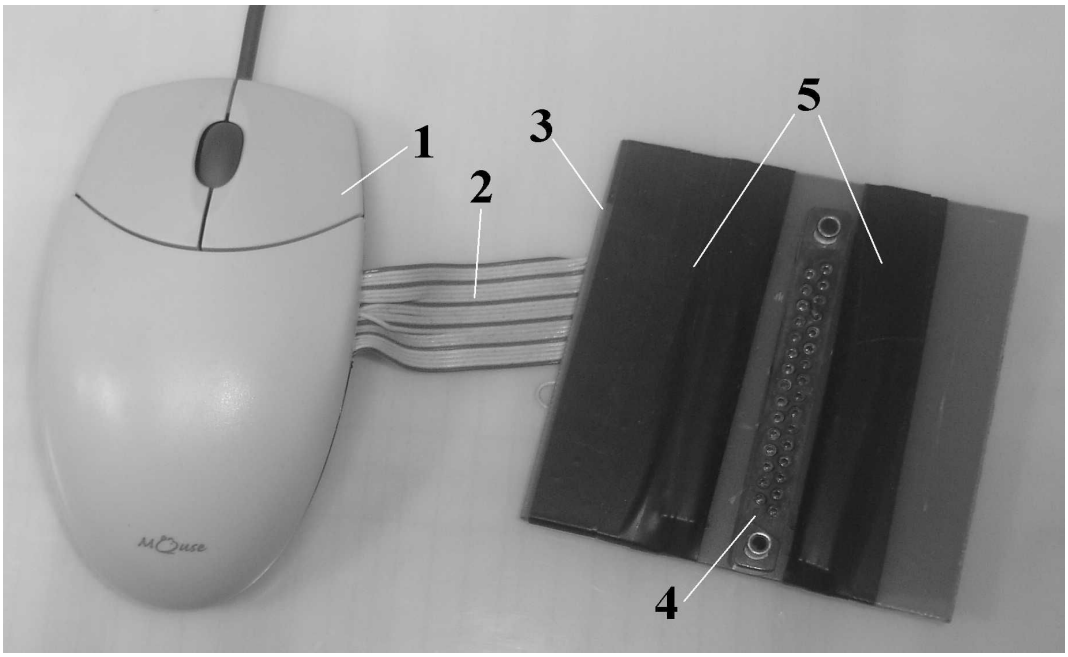

Рис. 1. Оптическая 3D USB/PS/2 Mouse с ЗУДКТ.

На рис.2 представлена внутренняя компоновка манипулятора 3D USB-PS/2 Mouse с проводниками-зондами кабеля (2) ЗУДКТ. На рис. 2 обозначены следующие компоненты:

- 1 - интерфейсный кабель манипулятора, посредством которого он подключается к разъему PS/2 порта системного блока ПК;
- 2 - кабель ЗУДКТ;
- 3 - ЗУДКТ;
- 4 - кнопочный датчик левой клавиши манипулятора;
- 5 - кнопочный датчик правой клавиши манипулятора;
- 6 - ролик подсистемы третьей координаты манипулятора;
- 7 - микросхема процессора видеокамеры манипулятора;
- 8 - микросхема контроллера манипулятора;
- 9 - оптическая система подсветки и видеокамеры.

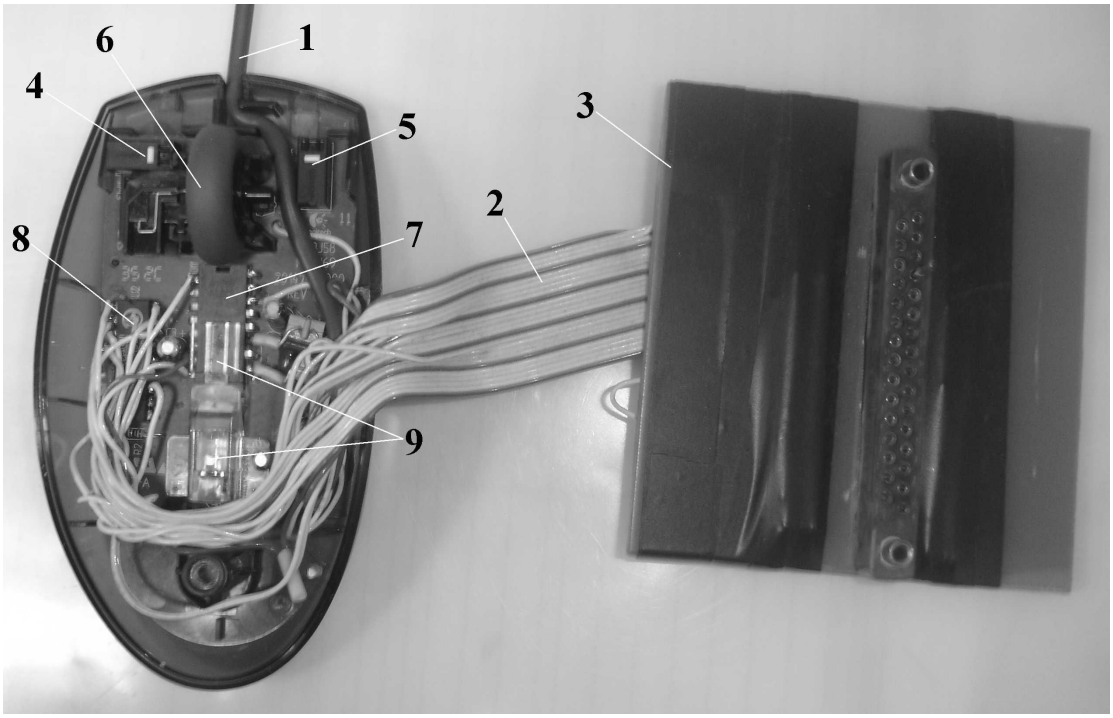

Рис 2. Пример подключения ЗУДКТ к электрической схеме оптической PS/2 Mouse.

3.1. Схемотехника 3D USB/PS/2 мыши

На рис. 3 представлена ***общая электрическая схема*** оптической 3D USB/PS/2 мыши, использующей в качестве датчиков перемещения по освещаемой поверхности встроенную видеокамеру с процессором, а в качестве датчика перемещения по третьей координате - сдвоенную оптопару.

Компоненты электрической схемы. В состав электрической схемы манипулятора 3D USB/PS/2 Mouse входят следующие компоненты:

- **Интерфейсный кабель**, содержащий четыре линии и защитный экран, подключенные к контактам разъема **J1**:
  - контакт 5 - MS-Clock - линия синхронизации;
  - контакт 1 - Корпус - оплетка защитного экрана;
  - контакт 4 - Data - линия передачи данных;
  - контакт 3 - +5V - линия питания;
  - контакт 2 - GND - сигнальная земля.
- **R14** - ограничительное сопротивление цепи питания +5V.
- **R12** - сопротивление "подтягивания" линии данных к потенциалу земли.
- **R13** - сопротивление "подтягивания" линии синхронизации к потенциалу земли.
- **C9** - конденсатор защиты линии синхронизации от высокочастотных помех.
- **C8** - конденсатор защиты линии данных от высокочастотных помех.
- **R2** - сопротивление, через которое вход 8 контроллера U2 подключается к линии данных интерфейса.
- **LD1** - светодиод подсветки поверхности перемещения.
- **U2 -** основной контроллер мыши, поддерживающий протоколы обмена шины USB и интерфейса PS/2-mouse.
- **U1** - процессор видеокамеры манипулятора.
- **C4 -** конденсатор развязки цепи питания по высоким частотам.

- **C5 -** конденсатор фильтра развязки цепи питания по низким частотам.

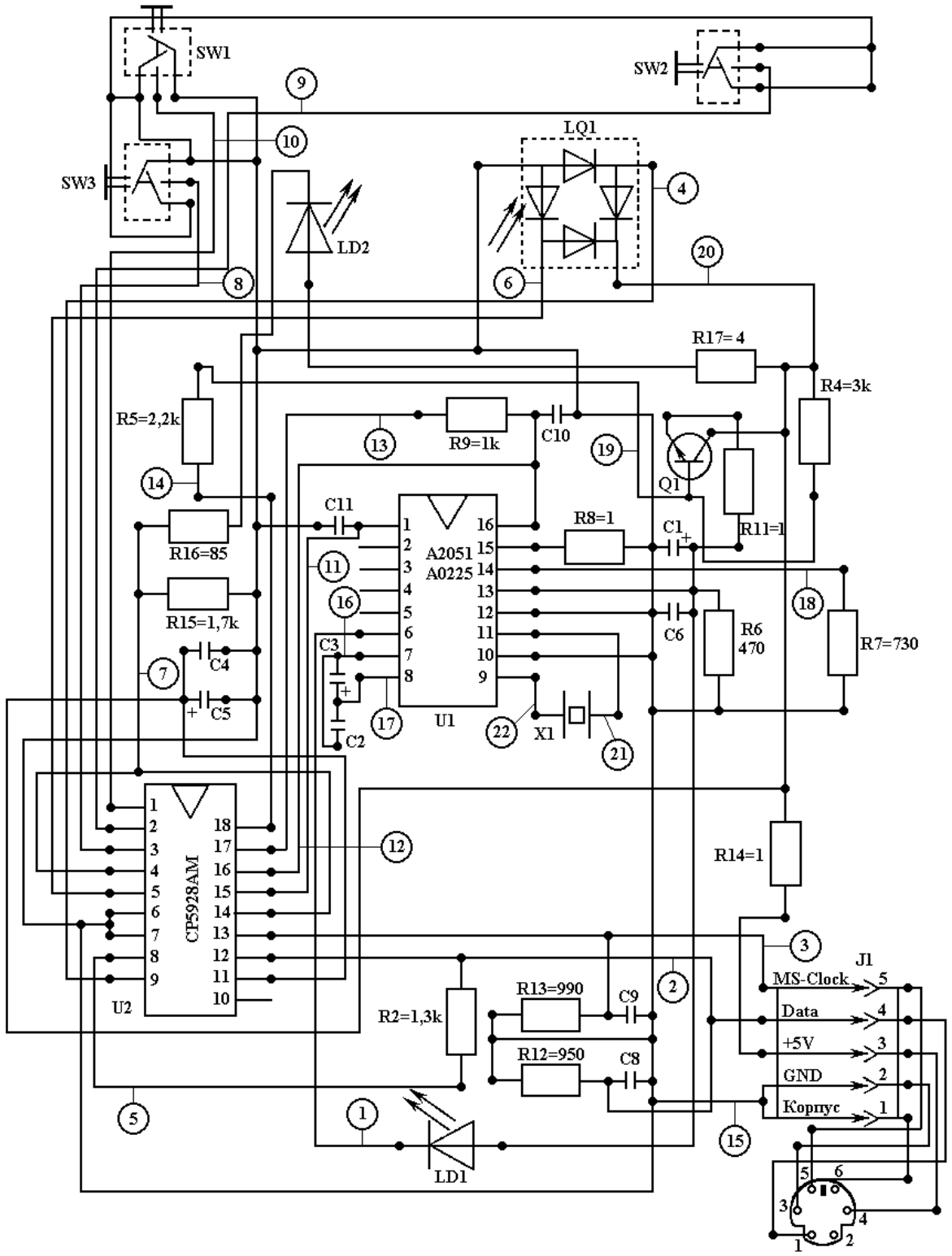

Рис. 3. Электрическая схема оптической 3D USB/PS/2 Mouse.

- **C2 и C3 -** конденсаторы высокочастотной и низкочастотной развязки внутренних цепей процессора видеокамеры.
- **R17 -** низкоомный резистор-перемычка подключения светодиода LD2 оптопары ролика прокрутки к цепи питания +5V.
- **R16 и R15 -** резисторы управляемой контроллером U2 цепи задания тока через светодиод LD2.
- **C10 и C11 -** конденсаторы высокочастотной развязки цепей взаимодействия процессора видеокамеры U1 и основного контроллера мыши U2.
- **R4 и R5 -** резисторы делителя, включенного в цепь базы транзистора Q1, задающего напряжение в цепи питания процессора U1 (контакт 13) и светодиода подсветки LD1. Делитель подключен к

управляющему контакту 18 контроллера U2.

- **C1 и C6** - конденсаторы низкочастотной и высокочастотной развязки цепи питания процессора U1.
- **R6** - балластное сопротивление цепи питания процессора U1.
- **R7** - резистор подключения контакта 14 процессора U1 к шине сигнальной земли.
- **R8** - токоограничивающее сопротивление в цепи контакта 15 процессора U1.
- **X1** - кварцевый резонатор.
- **SW1** - кнопочный датчик левой клавиши мыши.
- **SW2** - кнопочный датчик правой клавиши мыши.
- **SW3** - кнопочный датчик блока ролика прокрутки.

Контрольные точки электрической схемы. Контрольными точками обозначены цепи электрической схемы оптической мыши, в которых можно контролировать электрические сигналы, характеризующие работу мыши. Контроль сигналов осуществляется электронным осциллографом. В таблице 1 представлено описание контрольных точек и дано соответствие этих контрольных точек контактам разъема ЗУДКТ.

Таблица 1. Контрольные точки

Контрольные точки		Контакты разъема ЗУДКТ
1	Сигнал управления светодиодом LD1 подсветки, поступающий с контакта 6 процессора видеокамеры U1.	Б1
2	Сигналы на линии данных интерфейса мыши MS-Data.	Б2
3	Сигналы на линии синхронизации мыши MS-Clock.	Б3
4	Сигнал с правого плеча фотодиодного моста LQ1 датчика ролика прокрутки.	Б4
5	Сигналы, поступающие на контакт 8 контроллера U2 с линии данных интерфейса мыши.	Б5
6	Сигнал с левого плеча фотодиодного моста LQ1 датчика ролика прокрутки	Б6
7	Сигнал управления током через светодиод LD2 датчика ролика прокрутки; поступает с контактов 4 и 14 контроллера U2.	Б7
8	Сигнал с кнопочного датчика SW3 блока ролика прокрутки.	Б8
9	Сигнал с кнопочного датчика SW2 правой клавиши мыши.	Б9
10	Сигнал с кнопочного датчика SW1 левой клавиши мыши.	Б10
11	Сигналы цепи взаимодействия U1 и U2 через контакты U1-1 и U2-15.	А1
12	Сигналы цепи взаимодействия U1 и U2 через контакты U1-16 и U2-16.	А2
13	Сигналы, поступающие на контакт 17 контроллера U2 из цепи, обозначенной контрольной точкой 12.	А3
14	Сигнал управления транзистором Q1, поступающий с контакта 18 контроллера U2: он через делитель R5, R4 поступает на базу транзистора Q1.	А4
15	Сигнальная "Земля" (GND)	А5
16	Сигналы на контакте 7 процессора U1, соединенного через параллельно включенные конденсаторы C2 и C3 (C2 - электролитический, C3 - керамический) с его контактом 8.	А6
17	Сигнал контакта 8 процессора U1.	А7
18	Сигнал контакта 14 процессора U1.	А8
19	Сигнал на базе транзистора Q1.	А9
20	Потенциал на шине питания мыши.	А10
21	Сигнал кварцевого резонатора, поступающий на вывод 11 U1.	Б11
22	Сигнал кварцевого резонатора, поступающий на вывод 9 U1.	Б12

3.2. Интерфейс PS/2 Mouse.

PS/2-Mouse подключается к системе через двунаправленный последовательный интерфейс, аналогичный интерфейсу клавиатуры и обслуживается, как правило, тем же адаптером.

На физическом уровне адаптер обеспечивает передачу и прием по линиям данных и синхронизации интерфейса последовательностей электрических сигналов, с помощью которых реализуется передача и прием последовательностей битов информации, синхронизация передачи и приема, а также асинхронный протокол обмена, предусматривающий запросы, квитирование и управление потоком данных. Отличия от интерфейса клавиатуры выражаются только во временных параметрах используемых сигналов и в правилах подачи смещения на линии данных и синхронизации.

Для определения конкретных временных параметров сигнального протокола интерфейса PS/2 Mouse можно воспользоваться простой программой, представленной в листинге 1 (программа CPSD4E.TXT), выполняемой в среде отладчика DOS (Debug или AFD).

Листинг 1. Циклическое выполнение команды передачи в PS-Mouse, работающей в эхо режиме, кода EEh (программа CPSD4E.TXT).

```
2C47:0100  BB0011  MOV    BX,1100   ;Число циклов выполнения программы.
2C47:0103  FA      CLI              ;Запрет аппаратных прерываний.
2C47:0104  E464    IN     AL,[64]   ;Чтение рег. состояния адаптера и
2C47:0106  A802    TEST   AL,02     ;проверка: Вх. буфер пуст? Если нет,
2C47:0108  75FA    JNZ    0104      ;то новый цикл проверки.
2C47:010A  B0D4    MOV    AL,D4     ;Код команды D4h передается
2C47:010C  E664    OUT    [64],AL   ;в адаптер.
2C47:010E  E464    IN     AL,[64]   ;Чтение рег. состояния адаптера и
2C47:0110  A802    TEST   AL,02     ;проверка: Вх. Буфер пуст? Если нет,
2C47:0112  75FA    JNZ    010E      ;то новый цикл проверки.
2C47:0114  B0ED    MOV    AL,EE     ;Код команды EEh "Установить эхо-режим"
2C47:0116  E660    OUT    [60],AL   ;передается в PS/2 мышь.
2C47:0118  E464    IN     AL,[64]   ;Чтение рег. состояния адаптера и
2C47:011A  A801    TEST   AL,01     ;проверка признака "вых. буфер полон":
2C47:011C  74FA    JZ     0118      ;если пуст, то на новый цикл.
2C47:011E  E460    IN     AL,[60]   ;Чтение кода подтверждения мыши
2C47:0120  88C4    MOV    AH,AL     ;и пересылка его в регистр AH.
2C47:0122  E464    IN     AL,[64]   ;Чтение рег. состояния адаптера и
2C47:0124  A802    TEST   AL,02     ;проверка: Вх. буфер пуст? Если нет,
2C47:0126  75FA    JNZ    0122      ;то новый цикл проверки.
2C47:0128  E464    IN     AL,[64]   ;Чтение рег. состояния адаптера и
2C47:012A  A801    TEST   AL,01     ;проверка признака "вых. буфер полон":
2C47:012C  7406    JZ     0134      ;если пуст, то продолжение программы,
2C47:012E  E460    IN     AL,[60]   ;иначе чтение вых. буфера,
2C47:0130  89C6    MOV    SI,AX     ;запоминание его в регистре SI
2C47:0132  EBF4    JMP    0128      ;и переход на новый цикл проверки.
2C47:0134  B0D4    MOV    AL,D4     ;Код команды D4h передается
2C47:0136  E664    OUT    [64],AL   ;в адаптер.
2C47:0138  E464    IN     AL,[64]   ;Чтение рег. состояния адаптера и
2C47:013A  A802    TEST   AL,02     ;проверка: Вх. буфер пуст? Если нет,
2C47:013C  75FA    JNZ    0138      ;то новый цикл проверки.
2C47:013E  B0EE    MOV    AL,EE     ;Код EEh передается
2C47:0140  E660    OUT    [60],AL   ;в PS/2 мышь.
2C47:0142  E464    IN     AL,[64]   ;Чтение рег. состояния адаптера и
2C47:0144  A801    TEST   AL,01     ;проверка признака "вых. буфер полон":
2C47:0146  74FA    JZ     0142      ;если пуст, то на новый цикл проверки.
2C47:0148  E460    IN     AL,[60]   ;Чтение кода ответа мыши
2C47:014A  88C4    MOV    DH,AL     ;и запоминание его в регистре DH.
2C47:014C  B90F00  MOV    CX,000F   ;Вложенные
2C47:014F  51      PUSH   CX        ;циклы
2C47:0150  B9FFFF  MOV    CX,FFFF   ;программной
2C47:0153  E2FE    LOOP   0153      ;задержки, позволяющие выделить на
2C47:0155  59      POP    CX        ;на экране осциллографа циклы взаимо-
2C47:0156  E2F7    LOOP   014F      ;действия адаптера и мыши.
2C47:0158  4B      DEC    BX        ;Декремент счетчика циклов BX.
2C47:0159  75C7    JNZ    0122      ;Если BX не 0, то новый цикл программы.
2C47:015B  B0D4    MOV    AL,D4     ;Передача в
2C47:015D  E664    OUT    [64],AL   ;PS/2 Mouse
2C47:015F  E464    IN     AL,[64]   ;команды
2C47:0161  A802    TEST   AL,02     ;"Отменить
2C47:0163  75FA    JNZ    015F      ; эхо-режим"
2C47:0165  B0EC    MOV    AL,EC     ;(код
2C47:0167  E660    OUT    [60],AL   ;команды ECh).
2C47:0169  E464    IN     AL,[64]   ;Ожидание от мыши
2C47:016B  A801    TEST   AL,01     ;кода подтверждения
2C47:016D  74FA    JZ     0169      ;принятия команды ECh.
2C47:016F  E460    IN     AL,[60]   ;Чтение кода подтверждения
2C47:0171  88C2    MOV    DL,AL     ;и пересылка его в регистр DL.
2C47:0173  B020    MOV    AL,20     ;Сброс разряда D5 регистра состояния
2C47:0175  E664    OUT    [64],AL   ;адаптера 8042 командой 20h.
2C47:0177  E464    IN     AL,[64]   ;Ожидание завершения выполнения команды
```

```
2C47:0179  A801    TEST   AL,01     ;20h и чтение
2C47:017B  74FA    JZ     0177      ;результата выполнения команды
2C47:017D  E460    IN     AL,[60]   ;в регистр AL.
2C47:017F  FB      STI              ;Разрешение прерываний.
2C47:0180  CC      INT3             ;Остановка и передача упр. отладчику.
```

В программе реализовано циклическое выполнение команды D4h передачи байта данных в PS/2 Mouse, работающей в режиме "Эхо". В качестве байта данных выбран код EEh для того, чтобы можно было определить отличия сигнального протокола интерфейса PS/2 Mouse от сигнального протокола интерфейса клавиатуры, представленного в разделе 1.2.5.4.2 (осциллограммы (рис. 1.19) передачи в клавиатуру команды EEh "Эхо"). Программой CPSD4E.BIN код данных EEh передается через адаптер и интерфейс в PS/2 мышь, которая в ответ передает через интерфейс и адаптер код ответа EEh. Подключившись осциллографом к линиям MS-Clock и MS-Data интерфейса мыши можно фиксировать электрические сигналы, реализующие передачу этих кодов на физическом уровне в соответствие с сигнальным протоколом аппаратного интерфейса мыши. На рис. 4 представлены осциллограммы соответствующих сигналов интерфейса мыши.

После окончания циклов передачи кода EEh программа отменяет режим эхо и сбрасывает в регистре состояния адаптера 8042 флаговый разряд D5 "выходной буфер мыши заполнен" (Mouse_OBF), разрешает прерывания и передает управление отладчику. В регистрах AX и DX возвращается следующая информация:

- AL - результат выполнения команды 20h (например 47h - значение командного байта);
- AH - код подтверждения на команду EEh (код FAh - положительное подтверждение ACK);
- DL - код подтверждения на команду ECh (код FAh - положительное подтверждение ACK);
- DH - эхо код EEh.

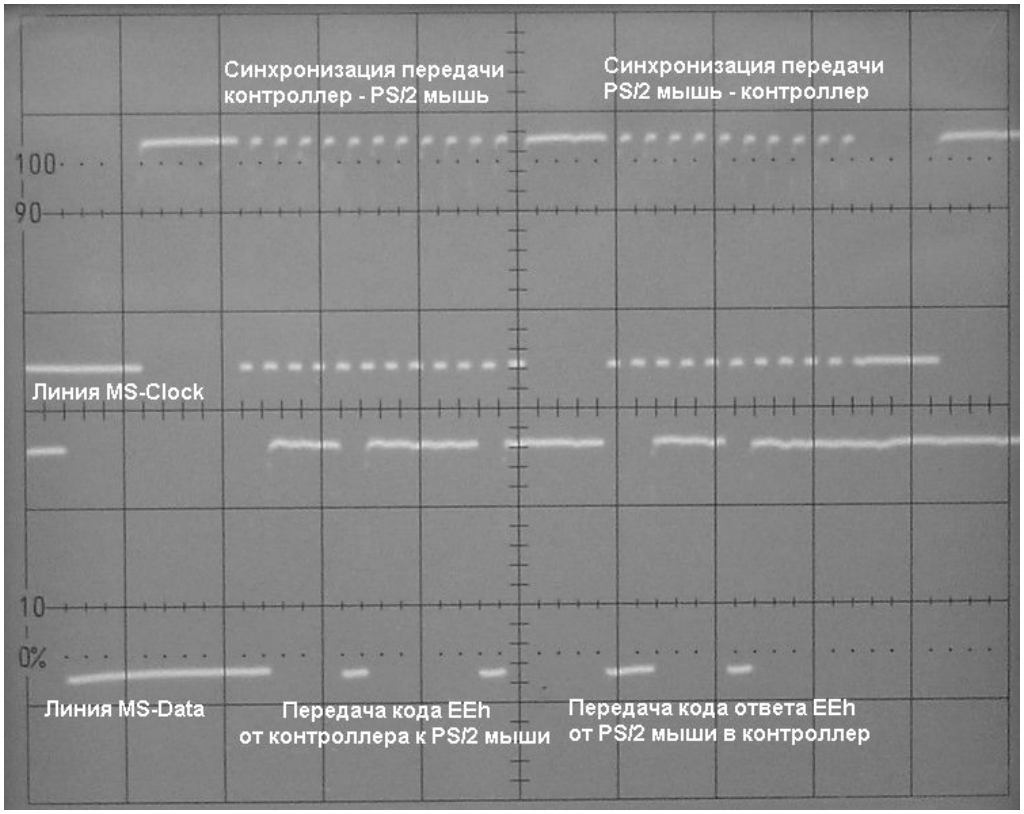

Рис. 4. Осциллограммы сигналов интерфейса PS/2 мыши, зарегистрированные при выполнении программы CPSD4E.BIN: развертка - 0,32 мс/дел., чувствительность по напряжению - 2 В/дел.; развертка запускается отрицательным фронтом сигнала линии MS-Clock.

На канальном уровне адаптер обеспечивает обмен байтами команд и данных между системой и мышью, используя те же порты ввода-вывода, что и клавиатура (60h и 64h). Отличие заключается в том, что:

- при передаче команд и данных в мышь перед записью каждого байта (и кода команды, и ее параметра) в порт 60h необходимо предварительно в порт 64h записывать команду адаптера D4h - передать мыши байт данных;
- при приеме байта данных от мыши адаптер устанавливает в регистре состояния флаг "выходной буфер мыши полон" (разряд D5) после чего примерно через 15 мкс передает полученный байт в выходной буфер адаптера и устанавливает флаг "выходной буфер полон" (разряд D0). При установке разряда D5 вырабатывается прерывание IRQ12, если оно не запрещено командным байтом 8242 (i8042B), а при установке разряда D0 - прерывание IRQ1, если оно не запрещено командным байтом. Прерывание IRQ12 необходимо закрепить за мышью с помощью процедуры BIOS SETUP, иначе оно

может быть захвачено каким-либо другим устройством и станет для мыши недоступным.

Более подробная информация о взаимодействии PS/2 Mouse и системы на канальном уровне приводится в разделе 1.3.4 учебного пособия.

Интерфейсный разъем PS/2 Mouse аналогичен разъему PS/2 клавиатуры (6-pin mini-DIN). Вид разъема (со стороны задней панели системного блока) и назначение контактов приведены на рис. 5.

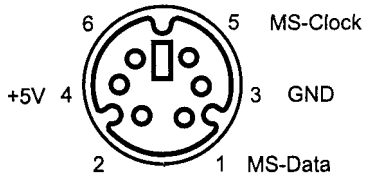

Рис. 5 Разъем PS/2-Mouse

3.3. Функционирование PS/2 Mouse.

Мышь PS/2 может находиться в одном из указанных ниже режимов работы.

- Потоковый режим - мышь выдает 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, то есть 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 осуществляет самотестирование и примерно через 0,5 секунд посылает компьютеру последовательность байтов: AAh, 00h - в случае нормального завершения теста, или FCh, 00h - в случае обнаружения ошибки. В обоих случаях второй байт 00h является идентификатором устройства типа PS/2 Mouse. 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" (F4h), которая активизирует мышь.

Управление режимами работы мыши и обменом байтами данных с ней может осуществляется непосредственно через порты 60h и 64h адаптера, функциями прерывание INT 15h, обрабатываемые BIOS, функциями программного прерывания INT 33h, обрабатываемые DOS и средствами стандартных прикладных программных интерфейсов, например Win32 API в среде современных ОС.

Группа форматов PS/2 Mouse.

При работе PS/2 мыши в потоковом режиме или в режиме дистанционного управления она передает информацию о своем перемещении, о прокрутке ролика третьей координаты и об изменение состояния своих кнопочных датчиков байтовыми пакетами, имеющими определенный формат.

Стандартный формат передачи данных для мыши PS/2 типа, разработанный фирмой IBM, показан в таблице 2.

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).

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, то есть противоположно оси Y дисплея. Преимуществом формата PS2 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).

Таблица 2. 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[1]	R	L
2	X7	X6	X5	X4	X3	X2	X1	X0
3	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

[1] У стандартных двухкнопочных манипуляторов этот бит всегда равен нулю.

В течение длительного времени область применения устройств PS/2-типа была ограниченной, однако с внедрением стандарта ATX началось их интенсивное распространение и развитие. В частности, появились новые, четырехбайтные протоколы передачи данных для трехкоординатных устройств, показанные в таблицах 3 и 4. При включение питания такие устройства начинают работать в стандартном режиме PS/2 со стандартным трехбайтным протоколом, а переключение в режим 3D с четырехбайтным протоколом происходит по специальной команде от драйвера мыши.

Таблица 3. Формат 3D PS/2 Mouse

Номер байта в посылке	Номер бита							
	7	6	5	4	3	2	1	0
1	0	0	YS	XS	1	M	R	L
2	X7	X6	X5	X4	X3	X2	X1	X0
3	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
4	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0

Таблица 4. Формат данных 5-кнопочной мыши Wheel Mouse

Номер байта в посылке	Номер бита							
	7	6	5	4	3	2	1	0
1	0	0	YS	XS	1	M	R	L
2	X7	X6	X5	X4	X3	X2	X1	X0
3	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
4	0	0	B5	B4	Z3	Z2	Z1	Z0

Формат 3D PS/2 Mouse предусматривает передачу координаты Z в четвертом байте пакета - в виде восьмиразрядного двоичного числа (старший разряд числа содержит знак). Формат пятикнопочной мыши Wheel Mouse отличается от предыдущего тем, что для передачи перемещения по Z применяются только четыре разряда, а еще два кодируют состояние специальных кнопок B4 и B5.

4. Указания по подготовке к лабораторной работе.

При подготовке к лабораторной работе, ее выполнении и оформлении отчета необходимо следовать рекомендациям раздела 4.1."Общие указания по выполнению и оформлению лабораторных работ".

Подготовка к данной лабораторной работе заключается в изучении описания лабораторной работы "Манипулятор PS/2 Mouse" и разделов 1.3. - 1.3.4.2 учебного пособия. Необходимо также ознакомиться с основными командами отладчика AFD и с командами процессора i8086.

5. Порядок выполнения лабораторной работы.

1. Ознакомиться с описанием электрической схемы 3D USB/PS/2 мыши и описанием ее контрольных точек.
2. Загрузить операционную систему Windows 95/98/Mel. в режиме Command Prompt Only, нажимая клавишу F8 перед началом загрузки ОС. При этом драйвер PS/2 мыши не загружается и манипулятор остается в состоянии остановки передачи данных (Disable).
3. С помощью электронного осциллографа определить электрические параметры сигналов в контрольных точках электрической схемы манипулятора (состояние контрольных точек), находящегося в состоянии Disable (амплитуда, длительность и период сигналов или напряжение постоянного уровня):
   - 3.1. Определить состояние контрольной точки 1 при нахождение PS/2 Mouse в состоянии покоя, при передвижении мыши, нажатии клавиш, прокрутке и нажатии ролика.
   - 3.2. Определить состояние контрольных точек 2, 3 и 5 при нахождение PS/2 Mouse в состоянии покоя, при передвижении мыши, нажатии клавиш, прокрутке и нажатии ролика.
   - 3.3. Определить состояние контрольных точек 4 и 6 при нахождение PS/2 Mouse в состоянии покоя и при прокрутке ролика.
   - 3.4. Определить состояние контрольной точки 7 при нахождение PS/2 Mouse в состоянии покоя и при прокрутке ролика.

3.5. Определить состояние контрольных точек 8, 9 и 10 при нахождение PS/2 Mouse в состоянии покоя и при нажатие соответствующей клавиши манипулятора и ролика.
3.6. Определить состояние контрольных точек 11 и 12 при нахождение PS/2 Mouse в состоянии покоя, при передвижении мыши, нажатии клавиш, прокрутке и нажатии ролика.
3.7. Определить состояние контрольных точек 13, 14 и 16 при нахождение PS/2 Mouse в состоянии покоя, при передвижении мыши, нажатии клавиш, прокрутке и нажатии ролика.
3.8. Определить состояние контрольных точек 17, 18 и 19 при нахождение PS/2 Mouse в состоянии покоя, при передвижении мыши, нажатии клавиш, прокрутке и нажатии ролика.
3.9. Определить состояние контрольных точек 21и 22.

4. В среде AFD набрать программу CPSD4E.TXT, представленную в листинге 1 раздела 3. Запустить программу командой G отладчика и определить состояние контрольных точек 2, 3 и 5:
4.1. Определить параметры синхроимпульсов (контрольная точка 3), их количество и общую длительность синхронизирующей последовательности передачи и приема одного байта. Определить время реакции мыши на принимаемый эхо-код.
4.2. Определить параметры сигналов передачи эхо-кода (контрольная точка 2). Выделить временные интервалы, соответствующие каждому биту передаваемого и принимаемого эхо-кода. Выделить временные интервалы стартового и стопового битов и бита контроля четности.
5. В среде AFD набрать программу 1, представленную в листинге 2 раздела 6. Приподняв мышь с поверхности стола запустить программу 1 командой G отладчика. Определить состояние контрольных точек 2, 3 и 5 при нажатие и отпускание левой клавиши мыши.
6. Приподняв мышь с поверхности стола запустить программу 1 командой G отладчика, нажать и отпустить левую клавишу мыши, затем правую и затем ролик. После этого прокрутить ролик на 2 щелчка вперед и на 2 щелчка назад, опустить мышь на стол и сдвинуть ее до окончания работы программы. Пользуясь данными таблицы 2 раздела 3 (табл. 1.22 раздела 1.3.4.1 пособия) идентифицировать кадры байтов, переданных мышью и представленных в окне памяти отладчика
7. Получить дополнительное задание у преподавателя.

6. Пример программы.

Программа 1 (см. листинг 2) командой CLI запрещает прерывания, командой адаптера ADh запрещает интерфейс клавиатуры, командой адаптера D4h разрешает интерфейс мыши и командой мыши F4h разрешает передачу данных от PS/2 мыши. После этого программа реализует 66 (42h) циклов приема байтов данных от мыши и размещение их в буфере памяти, начиная с адреса ES:0000h (22 трехбайтовых кадра). После заполнения буфера программа командой адаптера AEh разрешает работу интерфейса клавиатуры, командой мыши F5h останавливает передачу данных от мыши и командой адаптера A7h запрещает интерфейс мыши. После этого программа командой адаптера 20h ("Прочитать текущий командный байт") сбрасывает флаговый разряд D5 ("выходной буфер мыши заполнен") регистра состояния адаптера, разрешает прерывания и передает управление отладчику.

Листинг 2.Программа 1 (PSENDS.BIN), реализующая разрешение передачи данных от мыши и их прием в буфер памяти.

```
2C47:0100  FA      CLI             ;Запрет прерываний.
2C47:0101  E464    IN      AL,[64] ;Проверка водного буфера,
2C47:0103  A802    TEST    AL,02   ;если буфер заполнен, то
2C47:0105  75FA    JNZ     0101    ;новый цикл проверки.
2C47:0107  B0AD    MOV     AL,AD   ;Запрещение интерфейса клавиатуры
2C47:0109  E664    OUT     [64],AL ;командой ADh.
2C47:010B  E464    IN      AL,[64] ;Ожидание освобождения
2C47:010D  A802    TEST    AL,02   ;входного
```

```
2C47:010F  75FA    JNZ    010B      ;буфера.
2C47:0111  B0D4    MOV    AL,D4     ;Команда D4h "Передать
2C47:0113  E664    OUT    [64],AL   ;мыши байт данных".
2C47:0115  E464    IN     AL,[64]   ;Ожидание освобождения
2C47:0117  A802    TEST   AL,02     ;входного
2C47:0119  75FA    JNZ    0115      ;буфера.
2C47:011B  B0F4    MOV    AL,F4     ;Передача мыши команды F4h "Разрешить
2C47:011D  E660    OUT    [60],AL   ;передачу данных от мыши".
2C47:011F  E464    IN     AL,[64]   ;Ожидание поступления кода ответа мыши
2C47:0121  A801    TEST   AL,01     ;в выходной буфер
2C47:0123  74FA    JZ     011F      ;адаптера.
2C47:0125  E460    IN     AL,[60]   ;Чтение кода ответа и
2C47:0127  88C4    MOV    AH,AL     ;передача его в регистр AH.
2C47:0129  90      NOP              ;Пустые
2C47:012A  90      NOP              ;операции.
2C47:012B  BF0000  MOV    DI,0000   ;Счетчик-индекс DI обнуляется.
2C47:012E  E464    IN     AL,[64]   ;Ожидание поступления байта
2C47:0130  A801    TEST   AL,01     ;данных из мыши
2C47:0132  74FA    JZ     012E      ;в выходной буфер адаптера 8042.
2C47:0134  E460    IN     AL,[60]   ;Чтение поступившего байта и пересылка
2C47:0136  268805  MOV    ES:[DI],AL ;его в буфер памяти.
2C47:0139  47      INC    DI        ;Инкремент счетчика-индекса DI.
2C47:013A  83FF40  CMP    DI,0042   ;Если DI не равно 0, то переход ожидание
2C47:013D  75EF    JNZ    012E      ;поступления очередного байта от мыши.
2C47:013F  B0AE    MOV    AL,AE     ;Разрешение интерфейса клавиатуры
2C47:0141  E664    OUT    [64],AL   ;командой AE.
22D3:0143  E464    IN     AL,[64]   ;Ожидание освобождения
22D3:0145  A802    TEST   AL,02     ;входного
22D3:0147  75FA    JNZ    0143      ;буфера.
22D3:0149  B0D4    MOV    AL,D4     ;Команда D4h "Передать
22D3:014B  E664    OUT    [64],AL   ;мыши байт данных".
22D3:014D  E464    IN     AL,[64]   ;Ожидание освобождения
22D3:014F  A802    TEST   AL,02     ;входного
22D3:0151  75FA    JNZ    014D      ;буфера.
22D3:0153  B0F5    MOV    AL,F5     ;Передача мыши команды F5h "Остановить
22D3:0155  E660    OUT    [60],AL   ;передачу данных от мыши".
22D3:0157  E464    IN     AL,[64]   ;Ожидание поступления кода ответа мыши
22D3:0159  A801    TEST   AL,01     ;в выходной буфер
22D3:015B  74FA    JZ     0157      ;адаптера.
22D3:015D  E460    IN     AL,[60]   ;Чтение кода ответа и
22D3:015F  88C2    MOV    DL,AL     ;передача его в регистр DL.
22D3:0161  B0A7    MOV    AL,A7     ;Команда адаптеру "Запретить
22D3:0163  E664    OUT    [64],AL   ;работу с мышью.
22D3:0165  E464    IN     AL,[64]   ;Ожидание освобождения
22D3:0167  A802    TEST   AL,02     ;входного
22D3:0169  75FA    JNZ    0165      ;буфера.
22D3:016B  B020    MOV    AL,20     ;Команда адаптеру "Прочитать текущий командный
22D3:016D  E664    OUT    [64],AL   ;байт", сбрасывает D5 рег. состояния адаптера.
22D3:016F  E464    IN     AL,[64]   ;Ожидание поступления командного байта
22D3:0171  A801    TEST   AL,01     ;в выходной буфер
22D3:0173  74FA    JZ     016F      ;адаптера.
22D3:0175  E460    IN     AL,[60]   ;Чтение командного байта.
22D3:0177  FB      STI              ;Разрешение прерываний.
22D3:0178  CC      INT3             ;Возврат в отладчик.
```

7. Рекомендуемая литература

При подготовке к лабораторной работе и к ее защите рекомендуется пользоваться информацией, представленной в описании лабораторной работы, в разделе 1.3. настоящего учебно-методического пособия.

8. Контрольные вопросы

1. Дайте общую характеристику манипулятору PS/2 Mouse.
2. Охарактеризуйте два основных варианта PS/2 Mouse.
3. Охарактеризуйте компоненты PS/2 Mouse, представленные на рис. 1.21.

4. Дайте общую характеристику оптическому манипулятору мышь.
5. Охарактеризуйте основные компоненты оптической мыши, представленных на рис. 1.22 и 1.23.
6. Охарактеризуйте компоненты электрической схемы 3D PS/2 мыши, представленной на рис. 1.24.
7. Охарактеризуйте контрольные точки электрической схемы (рис. 1.24).
8. Охарактеризуйте компоненты электрической схемы 3D PS/2 мыши, представленной на рис. 1.25.
9. Охарактеризуйте контрольные точки электрической схемы (рис. 1.25).
10. Охарактеризуйте компоненты электрической схемы 3D USB/PS/2 оптической мыши, представленной на рис. 1.26.
11. Охарактеризуйте контрольные точки электрической схемы (рис. 1.26).
12. Кратко охарактеризуйте интерфейс PS/2 мыши.
13. Охарактеризуйте программный модуль, представленный в листинге 1.16.
14. Каковы отличия взаимодействия системы на канальном уровне с PS/2 мышью от ее взаимодействия на канальном уровне с клавиатурой?
15. Какие режимы работы поддерживает PS/2 мышь?
16. Какие операции выполняет PS/2 мышь после включения питания или получения команды Reset?
17. Как может осуществляться управление режимами работы PS/2 мыши?
18. Дайте описание стандартного формата передачи данных PS/2 мыши.
19. Чем отличается формат передачи 3D PS/2 мыши и 5-кнопочной 3D PS/2 мыши от стандартного формата PS/2 мыши?

6. Лабораторная работа 5 (описание) "Программное взаимодействие с PS/2 Mouse"

1. Цель работы

Целью лабораторной работы является ознакомление студентов с принципами взаимодействия системы с PS/2 Mouse на уровне команд, функций BIOS и функций DOS, изучение набора команд, выполняемых PS/2 Mouse и принципов программного управления режимами работы PS/2 Mouse на уровне команд и функций BIOS.

2. Технические и программные средства , используемые в работе.

В работе используется персональный компьютер типа Pentium с 101-клавишной клавиатурой. ПК работает под управлением ОС Windows 95/98/Mellenium. Операционная система загружается в режиме DOS (Command prompt only) с оболочкой типа Norton Commander. Непосредственное программное взаимодействие с PS/2 Mouse осуществляется в среде отладчика типа AFD.

3. Краткое описание средств и правил программного взаимодействия с PS/2 Mouse.

Программное взаимодействие с PS/2 мышью возможно через порты 60h и 64h, посредством функций программного прерывания INT 15h, обрабатываемых BIOS, посредством функций программного прерывания INT 33h, обрабатываемого DOS и средствами стандартных прикладных программных интерфейсов, например Win32 API в среде современных ОС.

3.1. Работа с PS/2 Mouse через порты ввода-вывода

Работа с PS/2 Mouse через порты ввода-вывода является самым низкоуровневым методом доступа. Такой метод используется обычно только обработчиками прерываний INT 74h, INT 15h и драйверами PS/2 Mouse. Обмен данными с PS/2 Mouse осуществляется через порт 60h адаптера клавиатуры командами IN и OUT центрального процессора ПК. Команда IN используется для чтения поступающих с PS/2 Mouse кадров и кодов ответов, команда OUT - для передачи в PS/2 Mouse команд и их атрибутов (параметров). При организации обмена порт 64h адаптера клавиатуры-мыши используется для синхронизации обмена и управления потом данных. Так, готовность адаптера к передаче байта в PS/2 Mouse или к передаче полученного от PS/2 Mouse байта в систему определяется состоянием разрядов 1, 5 и 0 регистра состояния адаптера, доступного по чтению через порт 64h в любое время (см. раздел 1.1.4.1.и 1.1.8.). Запрет или разрешение работы интерфейса PS/2 Mouse осуществляется командами A7h и A8h адаптера, передающимися в адаптер также через порт 64h (см. раздел 1.1.5.). Синхронизацию приема данных с клавиатуры можно осуществлять по прерыванию IRQ1 (INT 09h), генерируемому адаптером при установке в 1 разряда 0 регистра состояния адаптера и по прерыванию IRQ12 (INT 74h), генерируемого адаптером при установке в 1 разряда D5 регистра состояния адаптера (см. разделы 1.1.6. и 1.1.7.).

В отличие от клавиатуры, которая по умолчанию почти всегда может передавать свои скан-коды в систему, манипулятор PS-Mouse может передавать свои данные только по разрешению системы. Например, после включения питания и отработки начального теста PS/2 Mouse устанавливается по умолчанию в потоковый режим передачи с запрещением передачи данных. В этом случае PS/2 Mouse может начать передачу своих данных только после получения от системы команды F4h "Разрешить передачу данных от мыши". Если PS/2 Mouse командой EAh переключается в режим дистанционного управления, то она будет передавать свои данные только по команде EBh "Выдать пакет данных", поступающей от системы.

Ниже в листинге 1 раздела 6 приведен пример программы, разрешающей PS/2 Mouse передавать данные в потоковом режиме, а в листинге 2 - в режиме дистанционного управления с использованием портов 60h и 64h адаптера.

Отличия между клавиатурой и мышью присутствуют также в правилах передачи команд и их атрибутов системой в клавиатуру и в PS/2 мышь. При работе с клавиатурой любой байт данных, записанный во входной буфер адаптера по адресу 60h (если только это не атрибут команды адаптера) будет передан адаптером в клавиатуру. Для передачи же данных в PS/2 мышь необходимо предварительно подать адаптеру команду D4h "Передать манипулятору мышь байт данных". Только после отработки этой команды можно передать байт данных в PS/2 мышь через порт 60h адаптера.

Список команд, выполняемых мышью PS/2, приводится в таблице 1. Правила передачи команд в мышь и получения данных из мыши в общих чертах аналогичны правилам взаимодействия с клавиатурой. Как и при работе с клавиатурой каждая передача байта в PS/2 Mouse должна подтверждаться ей кодами положительного FAh (ACK), отрицательного FEh (NAC) подтверждения или другими байтовыми посылками. На взаимодей-

ствие системы с мышью через порт 60h распространяются также и правила взаимодействия системы с адаптером клавиатуры-мыши.

Таблица 1. Команды, выполняемые мышью PS/2

Код	Команда	Выполняемое действие
E6h	Reset Scaling	Отменить масштабирование перемещения
E7h	Set Scaling 2:1	Установить масштабирование перемещения для увеличения чувствительности мыши (см. табл. 1.26)
E8h	Set Resolution	Установить разрешение
E9h	Status Request	Получить информацию о состоянии устройства
EAh	Set Stream Mode	Установить потоковый режим
EBh	Read Data	Выдать пакет данных
ECh	Reset Wrap Mode	Отменить эхо-режим
EEh	Set Wrap Mode	Установить эхо-режим
F0h	Set Remote Mode	Установить режим дистанционного управления
F2h	Read Device Type	Определить тип устройства
F3h	Set Sampling Rate	Установить частоту дискретизации
F4h	Enable	Разрешить передачу данных от мыши
F5h	Disable	Остановить передачу данных от мыши
F6h	Set Default	Установить значение параметров работы мыши, используемое по умолчанию
FEh	Resend	Повторить передачу данных
FFh	Reset	Осуществить сброс и произвести самотестирование устройства

Достаточно подробное описание команд PS/2 Mouse с примерами программных модулей, реализующих выполнение этих команд, представлено в разделе 1.3.4.2 учебного пособия.

Программа, реализующая передачу байта в манипулятор PS/2 Mouse должна предусматривать выполнение следующих основных действий:

1. Запретить обработку аппаратных прерываний командой CLI.

2. Проверить состояние выходного буфера адаптера (выходного буфера клавиатуры) и, если он не пуст, очистить его командой чтения выходного буфера (IN AL, [60]).

3. Проверить состояние входного буфера и, если он занят, дождаться его освобождения адаптером. После освобождения входного буфера переслать по адресу 64h команду D4h -передать мыши байт данных.

4. После освобождения входного буфера от кода команды D4h по адресу 60h записать байт, предназначенный для передачи в мышь.

5. Затем организовать цикл ожидания активного признака "выходной буфер адаптера полон" (D0=1 в регистре состояния адаптера). Этот признак устанавливается, когда мышь передаст в адаптер ответный байт (код положительного подтверждения FAh, код отрицательного подтверждения FEh или какой то другой код, предусмотренный режимом работы мыши), который сначала фиксируется в выходном буфере мыши, устанавливая признак "выходной буфер мыши полон" (разряд D5 регистра состояния адаптера), а затем (примерно через 15 микросекунд) переписывается в выходной буфер адаптера, устанавливая признак "выходной буфер адаптера (клавиатуры) полон" (разряд D0 регистра состояния адаптера) и становится доступным для чтения через порт 60h.

6. После установки активного признака "выходной буфер адаптера полон" выполнить чтение из порта 60h, что сбрасывает признак "выходной буфер адаптера полон", но признак "выходной буфер мыши полон" не сбрасывается. На этом цикл передачи байта в мышь заканчивается.

7. Если дальнейших операций с мышью в программе не предусматривается, то необходимо сбросить признак "выходной буфер мыши полон" любой командой адаптера, предусматривающей возвращение в выходной буфер адаптера байта результата выполнения этой команды (см. раздел 1.1.8.3.5).

8. Если дальнейших операций в программе не предусмотрено, то необходимо командой STI разрешить обработку аппаратных прерываний и передать управление ведущей программе.

Чтение поступающих от PS/2 Mouse байтов данных почти ни чем не отличается от приема данных от клавиатуры. Отличить данные, поступающие с PS/2 Mouse, от данных клавиатуры можно по состоянию признака "выходной буфер мыши полон" (разряд D5 регистра состояния адаптера): если D5=1, то байт получен от мыши. Байт, поступающий с клавиатуры, сбрасывает разряд D5 и устанавливает в единицу разряд D0.

Если чтение данных от мыши и клавиатуры осуществляется обработчиками их прерываний, то синхронизация их работы также осуществляется однозначно.

Так, при поступление байта от мыши устанавливается признак "выходной буфер мыши полон", на основании которого формируется запрос на прерывание по линии IRQ 12 (номер вектора 74h - INT 74h) и активизируется обработчик прерывания мыши. Этот обработчик блокирует или перехватывает запрос прерывания по линии IRQ 1 (номер вектора 09h - INT 09h), который может активизироваться при установке в регистре состояния адаптера признака D0=1 "выходной буфер адаптера (клавиатуры) полон". Далее после поступления байта данных из выходного буфера мыши в выходной буфер адаптера осуществляет чтение

байта из выходного буфера адаптера по адресу 60h, сбрасывает признак "выходной буфер мыши полон" и разрешает или восстанавливает вектор прерывания по линии IRQ 1.

При поступление байта с клавиатуры устанавливается признак D0=1 в регистре состояния адаптера, генерируется запрос IRQ 1 и активизируется обработчик прерываний клавиатуры, который считывает байт из выходного буфера адаптера.

Одновременное поступление байтов от клавиатуры и мыши невозможно, поскольку при активизации передачи по одному из интерфейсов адаптер блокирует работу другого интерфейса.

3.2. Работа с PS/2 Mouse через функции BIOS прерывания INT 15h.

Для обслуживания мыши PS/2 имеется специальная функция C2h прерывания BIOS Int 15h - интерфейс координатного устройства. К сожалению, данный интерфейс носит вспомогательный характер - он служит только для управления параметрами мыши, а драйвер в его состав не входит.

После выполнения любой функции данной группы в случае успешного ее завершения флаг CF будет сброшен, в случае ошибки - установлен. Кроме того, в любом случае в регистр AH будет помещена дополнительная информация о результате операции:

- 00h - успешное завершение;
- 01h - недопустимый номер функции;
- 02h - недопустимое значение параметра;
- 03h - ошибка интерфейса;
- 04h - необходимо выполнить повторную передачу информации;
- 05h - драйвер устройства не установлен.

Рассмотрим набор подфункций интерфейса координатного устройства более подробно.

Прерывание INT 15h, функция C2h, подфункция 00h: разрешить или запретить работу мыши PS/2.

Подфункция позволяет разрешить или запретить работу мыши или другого координатного устройства, использующего интерфейс PS/2.

Перед вызовом прерывания INT 15h необходимо занести в регистры следующие значения:

- в AX - значение C200h;
- в BH - код команды (0 - запретить работу, 1 - разрешить работу).

После выполнения подфункции следует проверить флаг CF (флаг переноса). В случае возникновения ошибки флаг переноса будет установлен в 1 а в регистр AH будет помещен код ошибки. Ниже приведен пример использования подфункции 00h для запрещения работы мыши:

```
2C47:0100  B800C2    MOV    AX,C200 ;Помещение в AX номера функции.
2C47:0103  B700      MOV    BH,00   ;В AH заносим признак отключения мыши.
2C47:0105  CD15      INT    15      ;Вызов прерывания BIOS.
2C47:0107  CC        INT3           ;
```

При выполнении этого фрагмента программы без загруженного драйвера мыши приводит к ошибке с кодом 05h - драйвер устройства не установлен.

Прерывание INT 15h, функция C2h, подфункция 01h: произвести сброс устройства.

С помощью этой подфункции осуществляется перевод мыши или другого координатного устройства в исходное состояние.

Перед вызовом прерывания необходимо занести в регистр AX значение C201h.

Ниже приведен пример использования подфункции 01h для сброса PS/2 мыши.

```
2C47:0110  B801C2         MOV    AX,C201
2C47:0113  CD15           INT    15
2C47:0115  CC             INT3
```

После успешного выполнения этого примера в среде AFD параметры PS/2 мыши будут установлены в следующее состояние:

- частота передачи данных: 100 пакетов в секунду;
- разрешение: 4 отсчета на миллиметр;
- масштабирование 1:1;
- работа устройства запрещена.

Подфункция после завершения работы устанавливает флаг CF = 0, а регистр BX записывает следующие значения, переданные мышкой: старший байт BH - идентификатор устройства 00h, младший байт BL - значение AAh. Для выполнения этой подфункции не требуется загруженного драйвера.

Прерывание INT 15h, функция C2h, подфункция 02h: установить частоту дискретизации.

Подфункция позволяет изменить частоту дискретизации (частоту передачи пакетов данных от устройства к компьютеру).

Перед вызовом прерывания необходимо занести в регистры следующие значения:

- в AX - значение C202h;

- в BH - код частоты дискретизации (0 - 10 посылок/с, 1 - 20 посылок/с, 2 - 40 посылок/с, 3 - 60 посылок/с, 4 - 80 посылок/с, 5 - 100 посылок/с, 6 - 200 посылок/с).

Подфункция выполняеться и без загруженного драйвера мыши. Пример ее использования приведен ниже.

```
2C47:0120  B802C2          MOV     AX,C202
2C47:0123  B704            MOV     BH,04
2C47:0125  CD15            INT     15
2C47:0127  CC              INT3
```

После выполнения примера в мыши будет установлена частота дискретизации 80 посылок в секунду. Подфункция возвращает признак CF = 0 и код ошибки в регистре AH = 00h.

Прерывание INT 15h, функция C2h, подфункция 03h: установить разрешение.

Подфункция позволяет регулировать чувствительность мыши к перемещению.

Перед вызовом прерывания необходимо занести в регистры следующие значения:

- в AX - значение C203h;
- в BH - код устанавливаемого разрешения (0 - 1 отсчет на мм, 1 - 2 отсчета на мм, 2 - 4 отсчета на мм, 3 - 8 отсчетов на мм).

Подфункция может выполняться без загруженного драйвера мыши. Пример ее использования приведен ниже.

```
2C47:0130  B803C2          MOV     AX,C203
2C47:0133  B702            MOV     BH,02
2C47:0135  CD15            INT     15
2C47:0137  CC              INT3
```

После установки в мыши разрешения 4 отсчета на мм подфункция возвращает значение CF=0 и код ошибки в регистре AH=00h.

Прерывание INT 15h, функция C2h, подфункция 04h: получить идентификатор устройства.

Подфункция предназначена для определения типа подключенного к компьютеру координатного устройства.

Перед вызовом прерывания необходимо занести в регистр AX значение C204h.

В случае успешного выполнения функции в регистре BH будет возвращен идентификационный код устройства.

Подфункция может выполняться без загруженного драйвера мыши. Пример ее использования приведен ниже.

```
2C47:0140  B804C2          MOV     AX,C204
2C47:0143  CD15            INT     15
2C47:0145  CC              INT3
```

Подфункция возвращает значение CF=0, код ошибки в регистре AH=00h и код устройства 00h (манипулятор PS/2 Mouse) в регистре BH.

Прерывание INT 15h, функция C2h, подфункция 05h: инициализировать интерфейс координатного устройства

Подфункция инициализирует интерфейс координатного устройства. После ее выполнения устройство мыши будет иметь следующие параметры: выключено, разрешение - 4 отсчета на миллиметр, частота дискретизации - 100 выборок в секунду, масштабирование - 1 : 1.

Перед вызовом прерывания необходимо занести в регистры следующие значения:

- в AX - значение C205h;
- в BH - размер пакета данных в байтах (от 1 до 8).

Подфункция может выполняться без загруженного драйвера мыши. Пример ее использования приведен ниже. В нем помимо стандартных значений параметров мыши устанавливается длина пакета 6 байт

```
2C47:0150  B805C2          MOV     AX,C205
2C47:0153  B706            MOV     BH,06
2C47:0155  CD15            INT     15
2C47:0157  CC              INT3
```

После успешного выполнения подфункция 05h возвращает значение CF=0, код ошибки в регистре AH=00h и код устройства 00h (манипулятор PS/2 Mouse) в регистре BH.

Прерывание INT 15h, функция C2h, подфункция 06h: получить состояние или установить масштаб.

Подфункция позволяет определить состояние устройства или установить режим масштабирования.

Перед вызовом прерывания необходимо занести в регистры следующие значения:

- в AX - значение C206h;
- в BH - код выполняемой операции (0 - определить состояние устройства, 1 - установить масштаб 1:1, 2 - установить масштаб 2:1).

После завершения выполнения операций с кодами 1 или 2 функция возвращает только общий код состояния координатного устройства в регистре AH. В случае успешного завершения операции с кодом 0 в регистрах будут размещены следующие значения:

- в AH - общий код состояния;
- в BL - байт состояния устройства;
- в CL - код текущего разрешения устройства (0 - 1 отсчет на мм, 1 - 2 отсчета на мм, 2 - 4 отсчета на мм, 3 - 8 отсчетов на мм);
- в DL - частоту дискретизации (0Ah - 10 посылок/с, 14h - 20 посылок/с, 28h - 40 посылок/с, 3Ch - 60 посылок/с, 50h - 80 посылок/с, 64h - 100 посылок/с, C8h - 200 посылок/с).

Разряды байта состояния устройства, возвращаемого в BL, имеют следующее значение:

- бит 0 - состояние правой кнопки (0 - отпущена, 1 - нажата);
- бит 1 - состояние средней кнопки (0 - отпущена, 1 - нажата);
- бит 2 - состояние левой кнопки (0 - отпущена, 1 - нажата);
- бит 3 - зарезервирован (всегда 0);
- бит 4 - масштабирование (0 - режим 1:1, 1 - режим 2:1);
- бит 5 - передача данных от устройства к компьютеру (0 - запрещена, 1 - разрешена);
- бит 6 - режим (0 - потоковый, 1 - дистанционного управления);
- бит 7 - зарезервирован (всегда 0).

Подфункция может выполняться без загруженного драйвера мыши. Пример ее использования, с кодом выполняемой операции 00h, приведен ниже.

```
2C47:0160  B806C2          MOV     AX,C206
2C47:0163  B700            MOV     BH,00
2C47:0165  CD15            INT     15
2C47:0167  CC              INT3
```

После успешного выполнения подфункция 06h с кодом операции 00h она возвращает значение CF=0, код ошибки в регистре AH=00h и код устройства 00h (манипулятор PS/2 Mouse) в регистре BH. Помимо этого она возвращает в регистре BL байт состояния устройства 00h, в регистре CL значение 02h - 4 отсчета на мм и в регистре DL код 64h - 100 посылок в секунду.

Прерывание INT 15h, функция C2h, подфункция 07h: установить адрес обработчика прерываний.

Функция позволяет программисту установить свою собственную процедуру обработки прерываний от координатного устройства.

Перед вызовом прерывания необходимо занести в регистры следующие значения:

- в AX - значение C207h;
- в ES:BX - сегментный адрес и смещение пользовательского обработчика прерываний от координатного устройства. Для отмены пользовательского обработчика следует указать 0000h:0000h.

Подфункция может выполняться без загруженного драйвера мыши. Пример ее использования приведен ниже. В примере устанавливается адрес обработчика прерывания, расположенного в рабочей области отладчика начиная с адреса 2C47h:0200h.

```
2C47:0170  B8072C          MOV     AX,2C47
2C47:0173  8EC0            MOV     ES,AX
2C47:0175  BB0000          MOV     BX,0200
2C47:0178  B807C2          MOV     AX,C207
2C47:017B  CD15            INT     15
2C47:017D  CC              INT3
```

К сожалению, обработчик прерываний не входит в данный набор функций, и писать его должен сам программист.

4. Указания по подготовке к лабораторной работе.

При подготовке к лабораторной работе, ее выполнении и оформлении отчета необходимо следовать рекомендациям раздела 4.1."Общие указания по выполнению и оформлению лабораторных работ".

Подготовка к данной лабораторной работе заключается в изучении описания лабораторной работы "Программное взаимодействие с PS/2 Mouse", разделов 1.3.3. и 1.3.4 учебного пособия. Необходимо также ознакомиться с основными командами отладчика AFD и с командами процессора i8086.

5. Порядок выполнения лабораторной работы.

10. Ознакомиться с программой PSF4.BIN, представленной в листинге 1 раздела 6 описания лабораторной работы, выделить в ней участки, реализующие взаимодействие системы с мышью по программному каналу ввода-вывода и объяснить их назначение в организации программного канала ввода-вывода.

11. В окне дизассемблера отладчика AFD набрать программу PSF4.TXT, проверить и запустить ее командой "G" отладчика. В контрольных точках 2 и 3 проконтролировать сигналы при нажатие и отпускание клавиш клавиатуры и мыши (мышь должна находиться в приподнятом от поверхности стола положении). Определить параметры наблюдаемых сигналов и объяснить их назначение. (Клавиши нажимать пока программа не закончит свою работу).
12. Модифицируйте программу так, чтобы она запоминала поступающие от клавиатуры и PS/2 мыши байты данных в буфере памяти, начиная с адреса EX:0000h. Запустите программу командой "G" отладчика и проделайте те же манипуляции с клавишами клавиатуры и мыши, что и в пункте 2. В окне памяти отладчика идентифицируйте байты данных, принятых от клавиатуры и от PS/2 мыши (начало буфера памяти ES:0000h).
13. Ознакомиться с программой PSF0.BIN, представленной в листинге 2 раздела 6 описания лабораторной работы, выделить в ней участки, реализующие взаимодействие системы с мышью по программному каналу ввода-вывода и объяснить их назначение в организации программного канала ввода-вывода.
14. В окне дизассемблера отладчика AFD набрать программу PSF0.TXT, проверить и запустить ее командой "G" отладчика. В контрольных точках 2 и 3 проконтролировать сигналы при нажатие и отпускание клавиш клавиатуры и мыши (мышь должна находиться в приподнятом от поверхности стола положении). Определить параметры наблюдаемых сигналов и объяснить их назначение.
15. Модифицируйте программу так, чтобы она запоминала поступающие от клавиатуры и PS/2 мыши байты данных в буфере памяти, начиная с адреса EX:0200h. Запустите программу командой "G" отладчика и проделайте те же манипуляции с клавишами клавиатуры и мыши, что и в пункте 5. В окне памяти отладчика идентифицируйте байты данных, принятых от клавиатуры и от PS/2 мыши (начало буфера памяти ES:0200h).
16. Ознакомиться с программами, представленными в листингах 3 и 4 раздела 6 описания, выделить в них участки, реализующие взаимодействие системы с мышью по программному каналу ввода-вывода и объяснить их назначение в организации программного канала ввода-вывода, работа которого синхронизируется прерыванием IRQ 12 (INT 74h).
17. В окне дизассемблера отладчика AFD набрать программу IITPS.TXT, начиная с адреса CS:0100h и программу INTPS.TXT, начиная с адреса CS:0200, проверить их и запустить программу IITPS.TXT командой "G100" отладчика. Нажимайте 3 раза левую клавишу мыши, затем правую клавишу мыши (мышь должна находиться в приподнятом от поверхности стола положении). Опустить мышь на поверхность стола и сдвигать ее до окончания работы программы IITPS.TXT. В окне памяти отладчика идентифицировать байты данных, принятых обработчиком прерывания INT 74h (программа INTPS.TXT).
18. Пользуясь описанием команд PS/2 Mouse и программными модулями, реализующими их выполнение (раздел 1.3.4.2 учебного пособия) установить частоту дискретизации 40 пакетов/с и проверить установку параметра, используя команду E9h.
19. Выполнить задание пункта 9 с использованием функций программного прерывания INT15h (см. раздел 1.3.4.3 учебного пособия).
20. Получить дополнительное задание у преподавателя и выполнить его.

6. Примеры программ

Программа PSF4.TXT (листинг 1) реализует прием от клавиатуры и/или от PS/2 Mouse, работающей в потоковом режиме, 64 байт данных. Данные принимаются в регистр-аккумулятор AL и далее никуда не передаются, т.е. очередной принимаемый байт занимает место предыдущего. Предварительно программа командой F4h разрешает передачу данных от мыши, а после окончания циклов приема байтов данных командой F5h запрещает мыши передавать данные. В начале работы программы командой CLI запрещаются прерывания, а в конце программы командой STI они снова разрешаются.

Листинг 1. Прием от PS/2 Mouse и/или от клавиатуры 64 байт данных (программа PSF4.TXT).

```
2BF9:0100  BB4000  MOV    BX,0040   ;Счетчик принимаемых байтов данных.
2BF9:0103  FA      CLI              ;Запрет прерываний.
2BF9:0104  E464    IN     AL,[64]   ;Ожидание освобождения
2BF9:0106  A802    TEST   AL,02     ;входного
2BF9:0108  75FA    JNZ    0104      ;буфера.
2BF9:010A  B0D4    MOV    AL,D4     ;Команда адаптеру: "Передать манипу-
2BF9:010C  E664    OUT    [64],AL   ;лятору мышь байт данных".
2BF9:010E  E464    IN     AL,[64]   ;Ожидание освобождения
2BF9:0110  A802    TEST   AL,02     ;входного
2BF9:0112  75FA    JNZ    010E      ;буфера.
2BF9:0114  B0F4    MOV    AL,F4     ;Команда манипулятору: "Разрешить
2BF9:0116  E660    OUT    [60],AL   ;передачу данных от мыши".
2BF9:0118  E464    IN     AL,[64]   ;Ожидание поступления байтов данных
2BF9:011A  A801    TEST   AL,01     ;от манипулятора
2BF9:011C  74FA    JZ     0118      ;или от клавиатуры.
2BF9:011E  E460    IN     AL,[60]   ;Прием данных в регистр AL.
2BF9:0120  4B      DEC    BX        ;Декремент счетчика байт.
2BF9:0121  75F5    JNZ    0118      ;Если счетчик ≠ 0, то новый цикл приема.
2BF9:0123  B0D4    MOV    AL,D4     ;Команда адаптеру: "Передать манипу-
2BF9:0125  E664    OUT    [64],AL   ;лятору мышь байт данных".
2BF9:0127  E464    IN     AL,[64]   ;Ожидание освобождения
2BF9:0129  A802    TEST   AL,02     ;входного
2BF9:012B  75FA    JNZ    0127      ;буфера.
2BF9:012D  B0F5    MOV    AL,F5     ;Команда манипулятору: "Остановить
2BF9:012F  E660    OUT    [60],AL   ;передачу данных от мыши".
2BF9:0131  E464    IN     AL,[64]   ;Ожидание поступления от мыши
2BF9:0133  A801    TEST   AL,01     ;кода подтверждения
2BF9:0135  74FA    JZ     0131      ;приема команды.
2BF9:0137  E460    IN     AL,[60]   ;Чтение кода подтверждения.
2BF9:0139  FB      STI              ;Разрешение прерываний.
2BF9:013A  CC      INT3             ;Передача управления отладчику.
```

После завершения работы программы в регистре командного байта адаптера разряд D5 остается в состоянии 0, разрешая работу интерфейса PS/2 мыши. Если окончание работы программы произошло при получение данных от PS/2 мыши, то в регистре состояния адаптера бит D5 "Выходной буфер мыши заполнен" будет находиться в состояние 1, пока не будет нажата любая клавиша клавиатуры. В регистре-аккумуляторе AL будет находиться код подтверждения FAh (ACK), переданный PS/2 Mouse в ответ на команду F5h.

Программа PSF0.TXT (листинге 2), разрешает PS/2 мыши передавать данные в режиме дистанционного управления. Программа командой F0h переключает PS/2 Mouse в режим дистанционного управления. В этом режиме мышь по запросу системы передает ей 64 четырехбайтовых кадра. Первый байт в каждом кадре является кодом FAh подтверждения получения команды EBh "Выдать пакет данных", а последующие три байта соответствуют стандартному формату кадра PS/2 мыши (см. таблицу 1.22 в разделе 1.3.4.1 учебного пособия). В программе предусмотрена программная задержка после приема от мыши очередного кадра, что позволяет отделить кадры друг от друга по времени. Эта задержка позволяет однозначно отделять на экране осциллографы кадры друг от друга и дает студенту время изменять состояние мыши в процессе выполнения программы. После окончания циклов приема кадров программа командой F6h переключает мышь в потоковый режим с запрещением передачи данных и затем передает управление отладчику.

Листинг 2. Получение данных от PS/2 мыши, работающей в режиме дистанционного управления (программа PSF0.TXT).

```
2BF9:0100  FA      CLI              ;Запрет прерываний.
2BF9:0101  E464    IN     AL,[64]   ;Ожидание поступления
2BF9:0103  A801    TEST   AL,01     ;от клавиатуры
2BF9:0105  74FA    JZ     0101      ;кода отпускания
2BF9:0107  E460    IN     AL,[60]   ;клавиши Enter (9Ch), нажатием которой
2BF9:0109  3C9C    CMP    AL,9C     ;(код 1Ch)программа запускается
2BF9:010B  75F4    JNZ    0101      ;в среде отладчика AFD.
2BF9:010D  E464    IN     AL,[64]   ;Ожидание освобождения
2BF9:010F  A802    TEST   AL,02     ;входного буфера
2BF9:0111  75FA    JNZ    010D      ;адаптера.
2BF9:0113  B0D4    MOV    AL,D4     ;Команда "Передать манипулятору мышь
2BF9:0115  E664    OUT    [64],AL   ;байт данных" передается адаптеру.
```

```
2BF9:0117  E464    IN     AL,[64]   ;Ожидание освобождения
2BF9:0119  A802    TEST   AL,02     ;входного буфера
2BF9:011B  75FA    JNZ    0117      ;адаптера.
2BF9:011D  B0F0    MOV    AL,F0     ;Команда F0h "Set Remote Mode"
2BF9:011F  E660    OUT    [60],AL   ;передается PS/2 мыши.
2BF9:0121  E464    IN     AL,[64]   ;Ожидание поступления от мыши кода
2BF9:0123  A801    TEST   AL,01     ;подтверждения получения
2BF9:0125  74FA    JZ     0121      ;команды F0h.
2BF9:0127  E460    IN     AL,[60]   ;Чтение кода подтверждения и
2BF9:0129  3CFA    CMP    AL,FA     ;сравнение его с FAh, если не равен,
2BF9:012B  75F4    JNZ    0121      ;то новый цикл ожидания.
2BF9:012D  BB4000  MOV    BX,0040   ;Установка счетчика кадров.
2BF9:0130  B0D4    MOV    AL,D4     ;Команда "Передать манипулятору мышь
2BF9:0132  E664    OUT    [64],AL   ;байт данных" передается адаптеру.
2BF9:0134  E464    IN     AL,[64]   ;Ожидание освобождения
2BF9:0136  A802    TEST   AL,02     ;входного буфера
2BF9:0138  75FA    JNZ    0134      ;адаптера.
2BF9:013A  B0EB    MOV    AL,EB     ;Команда EBh "Выдать пакет данных"
2BF9:013C  E660    OUT    [60],AL   ;передается в PS/2 Mouse.
2BF9:013E  B90400  MOV    CX,0004   ;Установка счетчика байт в кадре.
2BF9:0141  E464    IN     AL,[64]   ;Ожидание поступления
2BF9:0143  A801    TEST   AL,01     ;байта данных
2BF9:0145  74FA    JZ     0141      ;от мыши.
2BF9:0147  E460    IN     AL,[60]   ;Чтение байта.
2BF9:0149  E2F6    LOOP   0141      ;Проверка конца кадра.
2BF9:014B  B9F400  MOV    CX,00F4   ;Вложенные
2BF9:014E  51      PUSH   CX        ;циклы
2BF9:014F  B9FFFF  MOV    CX,FFFF   ;программной
2BF9:0152  E2FE    LOOP   0152      ;задержки. Формируют временной
2BF9:0154  59      POP    CX        ;интервал между
2BF9:0155  E2F7    LOOP   014E      ;кадрами.
2BF9:0157  4B      DEC    BX        ;Декремент счетчика кадров.
2BF9:0158  75D6    JNZ    0130      ;Если не 0, то прием нового кадра.
2BF9:015A  B0D4    MOV    AL,D4     ;Команда "Передать манипулятору мышь
2BF9:015C  E664    OUT    [64],AL   ;байт данных" передается адаптеру.
2BF9:015E  E464    IN     AL,[64]   ;Ожидание освобождения
2BF9:0160  A802    TEST   AL,02     ;входного буфера
2BF9:0162  75FA    JNZ    015E      ;адаптера.
2BF9:0164  B0F6    MOV    AL,F6     ;Команда F6h "Set Default"
2BF9:0166  E660    OUT    [60],AL   ;передается в PS/2 Mouse.
2BF9:0168  E464    IN     AL,[64]   ;Ожидание поступления от мыши кода
2BF9:016A  A801    TEST   AL,01     ;подтверждения получения
2BF9:016C  74FA    JZ     0168      ;команды F6h.
2BF9:016E  E460    IN     AL,[60]   ;Чтение кода подтверждения.
2BF9:0170  FB      STI              ;Разрешение прерываний.
2BF9:0171  CC      INT3             ;Передача управления отладчику.
```

Как и в предыдущем примере после завершения работы программы в регистре командного байта адаптера разряд D5 остается в состоянии 0, разрешая работу интерфейса PS/2 мыши, а в регистре состояния адаптера бит D5 "Выходной буфер мыши заполнен" будет находиться в состояние 1, пока не будет нажата любая клавиша клавиатуры.

Программы IITPS.TXT и INTPS.TXT.

Ниже приводятся программные модули, реализующие прием информации от PS/2 мыши обработчиком прерывания INT 74h, по запросам IRQ 12, формируемым адаптером PS/2 Mouse при поступлении данных от PS/2 мыши.

Первый программный модуль IITPS.txt, представленный в листинге 3, выполняет основные действия по подготовке подсистемы PS/2 мыши к передаче данных мышью в потоковом режиме, к приему и размещению этих данных в буфере памяти обработчиком прерывания INT 74h, а также реализует циклы ожидания заполнения буфера данными. После заполнения буфера памяти данными работа PS/2 мыши запрещается, сбрасывается разряд D5 регистра состояния адаптера (признак: - выходной буфер мыши заполнен) и управление возвращается отладчику, который отображает в своем окне памяти содержимое буфера, заполненного обработчиком прерывания, начиная с адреса DS:0000h (128 баит).

В среде отладчика программный модуль IITPS.bin загружается с адреса DS:0100h. (В исходном состоянии в отладчике AFD в сегментных регистрах CS, DS, ES, SS, HS и FS содержатся одинаковые

значения, но загрузка программных модулей командой L ведется в сегмент данных DS). Запуск на выполнение модуля IITPS.bin осуществляется командой G 100 отладчика.

Листинг 3. Программный модуль IITPS.bin.

```
    ;Настройка ведомого контроллера прерываний.
2C47:0100  E4A1      IN     AL,[A1] ;Снятие маскирования запроса IRQ12,
2C47:0102  24EF      AND    AL,EF   ;поступающего в разряд D4
2C47:0104  EB00      JMP    0106    ;регистра запросов
2C47:0106  EB00      JMP    0108    ;ведомого контроллера
2C47:0108  E6A1      OUT    [A1],AL ;прерываний.
    ;Формирование вектора прерывания INT74h по адресу 0000:01D0h.
2C47:010A  B80000    MOV    AX,0000 ;В сегментный регистр ES
2C47:010D  8EC0      MOV    ES,AX   ;заносится код 0000h.
2C47:010F  B80002    MOV    AX,0200 ;В младшие два байта вектора заносится
2C47:0112  26A3D001  MOV    ES:[01D0],AX ;смещение 0200h
2C47:0116  8CC8      MOV    AX,CS   ;В старшие два байта вектора заносится
2C47:0118  26A3D201  MOV    ES:[01D2],AX ;содержимое сегментного регистра CS.
    ;Передача в PS/2 мышь команды f6h "Set Default" - установить параметры
    ;мыши, используемые по умолчанию (масштаб - 1:1, разрешение - 4 отсчета на
    ;миллиметр, частота дискретизации - 100 выборок и потоковый режим).
2C47:011C  FA        CLI            ;Запрет обработки прерываний.
2C47:011D  E464      IN     AL,[64] ;Проверка и
2C47:011F  A801      TEST   AL,01   ;очистка выходного
2C47:0121  7404      JZ     0127    ;буфера
2C47:0123  E460      IN     AL,[60] ;адаптера
2C47:0125  EBF6      JMP    011D    ;клавиатуры и PS/2 мыши.
2C47:0127  E464      IN     AL,[64] ;Проверка: входной буфер
2C47:0129  A802      TEST   AL,02   ;адаптера пуст?
2C47:012B  75FA      JNZ    0127    ;если нет, то новый цикл проверки.
2C47:012D  B0D4      MOV    AL,D4   ;Код команды "Передать мыши байт данных"
2C47:012F  E664      OUT    [64],AL ;передается в адаптер.
2C47:0131  E464      IN     AL,[64] ;Проверка: входной буфер
2C47:0133  A802      TEST   AL,02   ;адаптера пуст?
2C47:0135  75FA      JNZ    0131    ;если нет, то новый цикл проверки.
2C47:0137  B0F6      MOV    AL,F6   ;Код команды "Set Default"
2C47:0139  E660      OUT    [60],AL ;передается в PS/2 мышь.
2C47:013B  E464      IN     AL,[64] ;Проверка: выходной буфер
2C47:013D  A820      TEST   AL,20   ;мыши полон? Если пуст,
2C47:013F  74FA      JZ     013B    ;то на новый цикл проверки.
2C47:0141  E464      IN     AL,[64] ;Проверка: выходной буфер
2C47:0143  A801      TEST   AL,01   ;адаптера полон? Если пуст,
2C47:0145  74FA      JZ     0141    ;то на новый цикл проверки.
2C47:0147  E460      IN     AL,[60] ;Чтение кода ответа мыши и помещение
2C47:0149  88C4      MOV    AH,AL   ;его в регистр AH.
    Передача в PS/2 мышь команды F4h "Enable" - разрешить передачу данных от
    мыши, работающей в потоковом режиме.
2C47:014B  B0D4      MOV    AL,D4   ;Код команды "Передать мыши байт данных"
2C47:014D  E664      OUT    [64],AL ;передается в адаптер.
2C47:014F  E464      IN     AL,[64] ;Проверка: входной буфер
2C47:0151  A802      TEST   AL,02   ;адаптера пуст?
2C47:0153  75FA      JNZ    014F    ;если нет, то новый цикл проверки.
2C47:0155  B0F4      MOV    AL,F4   ;Код команды "Enable"
2C47:0157  E660      OUT    [60],AL ;передается в PS/2 мышь.
2C47:0159  E464      IN     AL,[64] ;Проверка: выходной буфер
2C47:015B  A820      TEST   AL,20   ;мыши полон? Если пуст,
2C47:015D  74FA      JZ     0159    ;то на новый цикл проверки.
2C47:015F  E464      IN     AL,[64] ;Проверка: выходной буфер
2C47:0161  A801      TEST   AL,01   ;адаптера полон? Если пуст,
2C47:0163  74FA      JZ     015F    ;то на новый цикл проверки.
2C47:0165  E460      IN     AL,[60] ;Чтение кода ответа мыши и помещение
2C47:0167  88C3      MOV    BL,AL   ;его в регистр BL.
    ;Установка в командном байте адаптера разрядов: D5 - разрешение работы ;с
    мышью и D1 - разрешение прерывания от мыши.
```

```
2C47:0169  B020      MOV    AL,20   ;Код команды "Прочитать текущий командный
2C47:016B  E664      OUT    [64],AL ;байт" передается в адаптер.
2C47:016D  E464      IN     AL,[64] ;Проверка: выходной буфер
2C47:016F  A801      TEST   AL,01   ;адаптера полон? Если пуст,
2C47:0171  74FA      JZ     016D    ;то на новый цикл проверки.
2C47:0173  E460      IN     AL,[60] ;Чтение командного байта, установка
2C47:0175  24DF      AND    AL,DF   ;в нем флага "Разрешение работы с мышью"
2C47:0177  0C02      OR     AL,02   ;и флага "Разрешение прерывания от мыши".
2C47:0179  88C7      MOV    BH,AL   ;и запоминание его в регистре BH.
2C47:017B  B060      MOV    AL,60   ;Код команды "Записать новый командный
2C47:017D  E664      OUT    [64],AL ;байт" передается в адаптер.
2C47:017F  E464      IN     AL,[64] ;Проверка: входной буфер
2C47:0181  A802      TEST   AL,02   ;адаптера пуст?
2C47:0183  75FA      JNZ    017F    ;если нет, то новый цикл проверки.
2C47:0185  88F8      MOV    AL,BH   ;Значение нового командного байта
2C47:0187  E660      OUT    [60],AL ;передается в адаптер.
   ;Организация циклов ожидания заполнения обработчиком прерывания INT74h
   ;буфера памяти размером 128 (80h) байт.
2C47:0189  B90000    MOV    CX,0000 ;Обнуление счетчика байт в буфере.
2C47:018C  FB        STI            ;Разрешение обработки прерываний.
2C47:018D  F6C180    TEST   CL,80   ;Проверка: буфер заполнен? (128 байт)
2C47:0190  74FB      JZ     018D    ;если нет, то на новый цикл проверки.
2C47:0192  90        NOP
2C47:0193  90        NOP
   ;Передача в PS/2 мышь команды F5h "Disable" - остановить передачу данных
   ;от мыши, работающей в потоковом режиме.
2C47:0194  50        PUSH   AX      ;Содержимое AX запоминается в стеке.
2C47:0195  B0D4      MOV    AL,D4   ;Код команды "Передать мыши байт данных"
2C47:0197  E664      OUT    [64],AL ;передается в адаптер.
2C47:0199  E464      IN     AL,[64] ;Проверка: входной буфер
2C47:019B  A802      TEST   AL,02   ;адаптера пуст?
2C47:019D  75FA      JNZ    0199    ;если нет, то новый цикл проверки.
2C47:019F  B0F5      MOV    AL,F5   ;Код команды "Disable"
2C47:01A1  E660      OUT    [60],AL ;передается в мышь.
2C47:01A3  E464      IN     AL,[64] ;Проверка: выходной буфер полон?
2C47:01A5  A801      TEST   AL,01   ;если нет, то
2C47:01A7  74FA      JZ     01A3    ;новый цикл проверки.
2C47:01A9  E460      IN     AL,[60] ;Чтение кода подтверждения мыши.
   ;Сброс в регистре состояния адаптера разряда D5 - выходной буфер мыши
   ;полон командой адаптера 20h - прочитать текущий командный байт.
2C47:01AB  B020      MOV    AL,20   ;Код команды 20h
2C47:01AD  E664      OUT    [64],AL ;передается в адаптер
2C47:01AF  E464      IN     AL,[64] ; Проверка: выходной буфер полон?
2C47:01B1  A801      TEST   AL,01   ;если нет, то
2C47:01B3  74FA      JZ     01AF    ;новый цикл проверки.
2C47:01B5  E460      IN     AL,[60] ;Чтение командного байта.
   ;Завершение работы программы.
2C47:01B7  58        POP    AX      ;Восстановление из стека AX.
2C47:01B8  CC        INT3           ;Остановка и выход в AFD.
```

Второй программный модуль (INTPS.txt), представленный в листинге 4, выполняет функции обработчика прерываний. Он формирует буфер памяти, осуществляет его заполнение данными, обрабатывая прерывание INT 74h, и формирует информацию о степени заполнения буфера. В среде отладчика обработчик INTPS.bin загружается с адреса DS:0200h.

Листинг 4. Обработчик прерывания INT 74h (программный модуль INTPS.bin).

```
2C47:0200  57        PUSH   DI      ;Сохранение в стеке
2C47:0201  06        PUSH   ES      ;значений регистров
2C47:0202  50        PUSH   AX      ;процессора:
2C47:0203  53        PUSH   BX      ;DI, ES,AX,BX.
2C47:0204  FA        CLI            ;Запрещение обработки прерываний.
2C47:0205  BB0000    MOV    BX,0000 ;Обнуление счетчика байтов пакета.
2C47:0208  8CD8      MOV    AX,DS   ;Установка сегмента
2C47:020A  8EC0      MOV    ES,AX   ;буфера памяти.
```

```
2C47:020C  E464      IN     AL,[64] ;Проверка наличия байта в выходном
2C47:020E  A801      TEST   AL,01   ;буфере адаптера: если буфер
2C47:0210  74FA      JZ     020C    ;пуст, то на новый цикл проверки.
2C47:0212  E460      IN     AL,[60] ;Чтение байта, поступившего из мыши.
2C47:0214  89CF      MOV    DI,CX   ;Позиционирование на адрес в буфере памяти
2C47:0216  268805    MOV    ES:[DI],AL    ;Запись принят. байта в буф. памяти.
2C47:0219  41        INC    CX      ;Инкремент счетчика записанных в буф.байт.
2C47:021A  FEC3      INC    BL      ;Инкремент счетчика байт пакета.
2C47:021C  80FB03    CMP    BL,03   ;Если меньше 03, то на чтение очередного
2C47:021F  75EB      JNZ    020C    ;байта.
2C47:0221  B020      MOV    AL,20   ;Сброс D5 регистра состояния
2C47:0223  E664      OUT    [64],AL ;передачей адаптеру команды "Прочитать
2C47:0225  E464      IN     AL,[64] ;текущий командный байт" и.
2C47:0227  A801      TEST   AL,01   ;ожидание поступления командного байта
2C47:0229  74FA      JZ     0225    ;в выходной буфер адаптера.
2C47:022B  E460      IN     AL,[60] ;Чтение командного байта.
2C47:022D  B020      MOV    AL,20   ;Пересылка команды EOI (завершение обра-
2C47:022F  E6A0      OUT    [A0],AL ;ботки прерывания) в ведомый и ведущий
2C47:0231  E620      OUT    [20],AL ;контроллеры прерываний.
2C47:0233  5B        POP    BX      ;Восстановление из стека
2C47:0234  58        POP    AX      ;значений регистров
2C47:0235  07        POP    ES      ;процессора:
2C47:0236  5F        POP    DI      ;BX, AX, ES,DI.
2C47:0237  FB        STI            ;Разрешение обработки прерываний.
2C47:0238  CF        IRET           ;Возврат в прерванную программу.
```

7. Рекомендуемая литература.

При подготовке к лабораторной работе и к ее защите рекомендуется пользоваться информацией, представленной в описании лабораторной работы, в разделе 1.3. учебно-методического пособия и в дополнительной литературе [5, 6, 8, 13, 15].

8. Контрольные вопросы

1. Кратко охарактеризуйте команды, выполняемые PS/2 мышью.
2. Дайте описание основных действий программы, реализующей передачу байта в PS/2 мышь.
3. Охарактеризуйте особенности чтения данных, поступающих от PS/2 мыши.
4. Охарактеризуйте программу, представленную в листинге 1.17.
5. Объясните действие команды E7h.
6. Охарактеризуйте программу, представленную в листинге 1.18.
7. Охарактеризуйте программу, представленную в листинге 1.19.
8. Объясните осциллограмму, представленную на рис. 1.29.
9. Охарактеризуйте команду EBh.
10. Охарактеризуйте команды ECh и EDh.
11. Охарактеризуйте команды F0h - F2h.
12. Охарактеризуйте команду F3h.
13. Охарактеризуйте команды F4h и F5h.
14. Охарактеризуйте команду F6h.
15. Охарактеризуйте команду FFh.
16. Охарактеризуйте функцию C3h прерывания BIOS INT 15h.
17. Охарактеризуйте подфункции 00h и 01h функции C2h.
18. Охарактеризуйте подфункции 02h и 03h функции C2h.
19. Охарактеризуйте подфункции 04h и 05h функции C2h.
20. Охарактеризуйте подфункции 06h функции C2h.
21. Охарактеризуйте подфункции 07h функции C2h.
22. Для чего используются процедуры, представленные в листинге 1.20?
23. Охарактеризуйте программу, представленную в листинге 1.21.
24. Охарактеризуйте программу, представленную в листинге 1.22.
25. Охарактеризуйте программный модуль, представленный в листинге 1.23.
26. Охарактеризуйте программный модуль, представленный в листинге 1.24.

7. Лабораторная работа 6 (описание) "Последовательный асинхронный порт ПК (COM-порт)"

1. Цель работы

Целью лабораторной работы является:

- ознакомление студентов с принципами работы контроллера COM-порта, реализующего дуплексный обмен с ПУ и объектами внешнего мира в соответствии с правилами последовательного асинхронного интерфейса RS 232C;
- изучение набора команд, выполняемых контроллером;
- изучение принципов программного управления режимами работы контроллера и обменом через него на уровне регистров контроллера и на уровне функций прерывания INT 14h, обрабатываемого BIOS.

2. Технические и программные средства, используемые в работе.

В работе используется персональный компьютер типа Pentium, имеющий COM-порты и работающий под управлением операционной системы Windows 95/98/Melenium. Операционная система загружается в режиме DOS (Command prompt only) с оболочкой типа Norton Commander. Программное взаимодействие с контроллером осуществляется в среде отладчика типа AFD.

В работе используется также электронный осциллограф с полосой регистрации 0 - 10 МГц и выше и кабель имеющий с одной стороны разъем *DB-9P* (розетка), подключаемый к разъему COM-порта компьютера, а с другой стороны проводники кабеля доступны для подключения щупов осциллографа. В выходные цепи COM-порта включены защитные резисторы величиной 2 КОм.

3. Краткое описание интерфейса RS-232C, контроллера COM-порта и функций драйвера COM-порта INT 14h.

3.1. Интерфейс RS-232C

Интерфейс RS-232C предназначен для подключения аппаратуры, передающей или принимающей данные *(ООД* — оконечное оборудование данных, или *АПД* — аппаратура передачи данных; *DTE— Data Terminal Equipment),* к оконечной аппаратуре каналов данных *(АКД; DCE — Data Communication Equipment).* В роли АПД может выступать компьютер, принтер, плоттер и другое периферийное оборудование. В роли АКД обычно выступает модем. Конечной целью подключения является соединение двух устройств АПД. Полная схема соединения приведена на рис. 1; интерфейс позволяет исключить канал удаленной связи вместе с парой устройств АКД, соединив устройства непосредственно с помощью нуль-модемного кабеля (рис. 2).

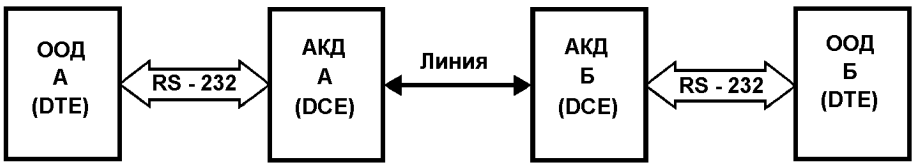

Рис. 1. Полная схема соединения по RS-232C

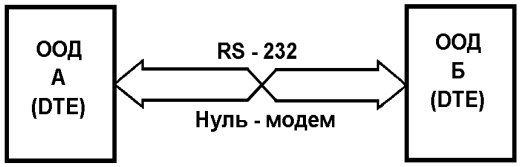

Рис. 2. Соединение по RS-232C нуль-модемным кабелем

Стандарт описывает управляющие сигналы интерфейса, пересылку данных, электрический интерфейс и типы разъемов. В стандарте предусмотрены асинхронный и синхронный режимы обмена, но COM-порты поддерживают только *асинхронный режим.* Функционально *RS-232C* эквивалентен стандарту МККТТ V.24/ V.28 и стыку С2, но они имеют различные названия сигналов.

Стандарт *RS-232C* описывает несимметричные передатчики и приемники — сигнал передается

относительно общего провода — схемной земли (симметричные дифференциальные сигналы используются в других интерфейсах — например, *RS-422).* Интерфейс *не обеспечивает гальванической развязки* устройств. Логической единице соответствует напряжение на *входе приемника* в диапазоне от -12 до -3 В. Для линий управляющих сигналов это состояние называется *ON* («включено»), для линий последовательных данных — *MARK*. Логическому нулю соответствует диапазон от +3 до +12 В. Для линий управляющих сигналов состояние называется OFF(«выключено»), а для линий последовательных данных — *SPACE*. Диапазон от -3 до +3 В — зона нечувствительности, обусловливающая гистерезис приемника: состояние линии будет считаться измененным только после пересечения порога (рис. 3). Уровни сигналов на выходах передатчиков должны быть в диапазонах от -12 до -5 В и от +5 до +12 В для представления единицы и нуля соответственно. Разность потенциалов между схемными землями (SG) соединяемых устройств должна быть менее 2 В, при более высокой разности потенциалов возможно неверное восприятие сигналов.

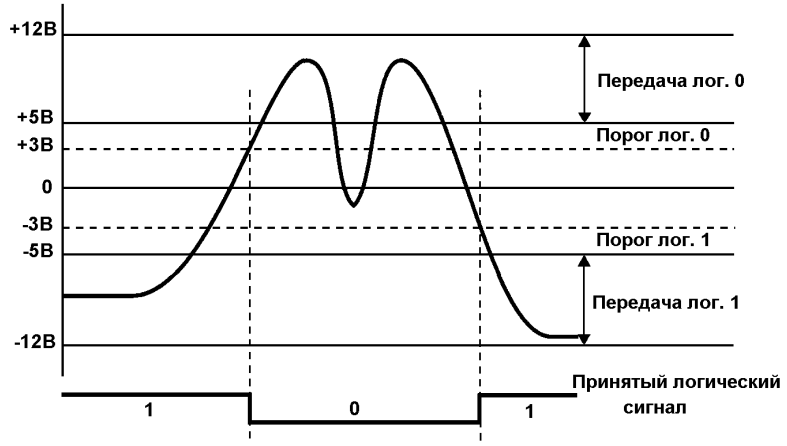

Рис. 3. Прием сигналов RS-232C

Интерфейс предполагает наличие *защитного заземления* для соединяемых устройств, если они оба питаются от сети переменного тока и имеют сетевые фильтры.

ВНИМАНИЕ -

Подключение и отключение интерфейсных кабелей устройств с автономным питанием должно производиться *при отключенном питании.* Иначе разность не выровненных потенциалов устройств в момент коммутации может оказаться приложенной к выходным или входным (что опаснее) цепям интерфейса и вывести из строя микросхемы.

Стандарт RS-232C регламентирует *типы применяемых разъемов.*

На аппаратуре АПД (в том числе на COM-портах) принято устанавливать *вилки DB-25P* - рис.4а или более компактный вариант — *DB-9P* (рис. 4б). Девятиштырьковые разъемы не имеют контактов для дополнительных сигналов, необходимых для синхронного режима (в большинстве 25-штырьковых разъемах эти контакты не используются). На аппаратуре *АКД* (модемах) устанавливают *розетки DB-25S* или *DB-9S.*

Это правило предполагает, что разъемы АКД могут подключаться к разъемам АПД непосредственно или через переходные «прямые» кабели с розеткой и вилкой, у которых контакты соединены «один в один». Переходные кабели могут являться и переходниками с 9 на 25-штырьковые разъемы (рис. 5).

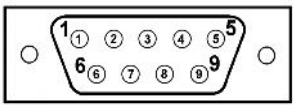

Рис. 4а. **9-контактный штырьковый разъем, вид со стороны задней стенки компьютера**

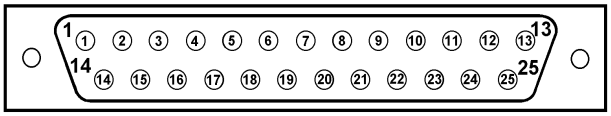

Рис. 4б. **25-контактный штырьковый разъем, вид со стороны задней стенки компьютера**

Если аппаратура АПД соединяется без модемов, то разъемы устройств (вилки) соединяются между собой *нуль-модемным кабелем* (Zero-modem, или Z-modem), имеющим на обоих концах розетки, контакты которых соединяются перекрестно по одной из схем, приведенных на рис. 6.

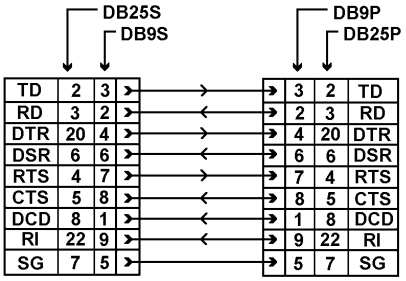

Рис. 5. Кабели подключения модемов

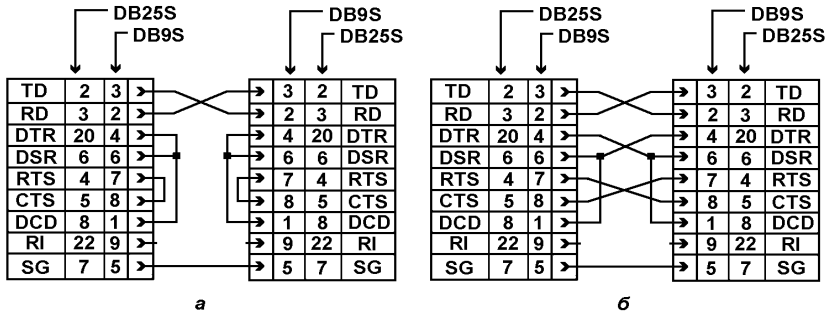

Рис. 6. Нуль-модемный кабель: а — минимальный, б — полный

Если на каком-либо устройстве АПД установлена розетка — это почти 100 % того, что к другому устройству оно должно подключаться прямым кабелем, аналогичным кабелю подключения модема. Розетка устанавливается обычно на тех устройствах, у которых удаленное подключение через модем не предусмотрено.

В табл. 1 приведено назначение контактов разъемов COM-портов (и любой другой аппаратуры передачи данных АПД). Контакты разъема DB-25S определены стандартом EIA/TIA-232-E, разъем DB-9S описан стандартом EIA/TIA-574. У модемов (АКД) название цепей и контактов такое же, но роли сигналов (вход-выход) меняются на противоположные.

Таблица 1. Разъемы и сигналы интерфейса RS-232C

Обозначение цепи			Контакт разъема		№ провода кабеля выносного разъема PC				Направление
COM-порт	RS-232	V.24 Стык 2	DB-25P	DB-9P	1[1]	2[2]	3[3]	4[4]	I/O
PG	AA	101	1	5	(10)	(10)	(10)	1	-
SG	AB	102	7	5	5	9	1	13	-
TD	BA	103	2	3	3	5	3	3	O
RD	BB	104	3	2	2	3	4	5	I
RTS	CA	105	4	7	7	4	8	7	O
CTS	CB	106	5	8	8	6	7	9	I
DSR	CC	107	6	6	6	2	9	11	I
DTR	CD	108/2	20	4	4	7	2	14	O
DCD	CF	109	8	1	1	1	5	15	I
RI	CE	125	22	9	9	8	6	18	I

[1] Ленточный кабель 8-битных мультиадаптеров.

[2] Ленточный кабель 16-битных мультиадаптеров и портов на системных платах.

[3] Вариант ленточного кабеля портов на системных платах.

[4] Широкий ленточный кабель к 25-контактному разъему.

Подмножество сигналов *RS-232C,* относящихся к асинхронному режиму, рассмотрим с точки зрения COM-порта PC. Для удобства будем пользоваться мнемоникой названий, принятой в описаниях COM-портов и большинства устройств (она отличается от безликих обозначений RS-232 и V.24). Напомним, что активному состоянию сигнала («включено») и *логической единице* передаваемых данных соответствует *отрицательный потенциал* (ниже -3 В) сигнала интерфейса, а состоянию «выключено» и *логическому нулю* — *положительный* (выше +3 В). Назначение сигналов интерфейса приведено в табл. 2. Нормальную последовательность управляющих сигналов для случая подключения модема к COM-порту иллюстрирует рис. 7.

Таблица 2. Назначение сигналов интерфейса RS-232C

Сигнал	Назначение
PG	*Protected Ground* — защитная земля, соединяется с корпусом устройства и экраном кабеля
SG	*Signal Ground* — сигнальная (схемная) земля, относительно которой действуют уровни сигналов
TD	*Transmit Data* — последовательные данные — выход передатчика
TD	*Receive Data* — последовательные данные — вход приемника
RTS	*Request To Send* — выход запроса передачи данных: состояние «включено» уведомляет модем о наличии у терминала данных для передачи. В полудуплексном режиме используется для управления направлением — состояние «включено» служит сигналом модему на переключение в режим передачи.
CTS	*Clear To Send* — вход разрешения терминалу передавать данные. Состояние «выключено» запрещает передачу данных. Сигнал используется для аппаратного управления потоками данных
DSR	*Data Set Ready*— вход сигнала готовности от аппаратуры передачи данных (модем в рабочем режиме подключен к каналу и закончил действия по согласованию с аппаратурой на противоположном конце канала)
DTR	*Data Terminal Ready* — выход сигнала готовности терминала к обмену данными. Состояние «включено» поддерживает коммутируемый канал в состоянии соединения.
DCD	*Data Carrier Detected* — вход сигнала обнаружения несущей удаленного модема.
RI	*Ring Indicator* — вход индикатора вызова (звонка). В коммутируемом канале этим сигналом модем сигнализирует о принятии вызова

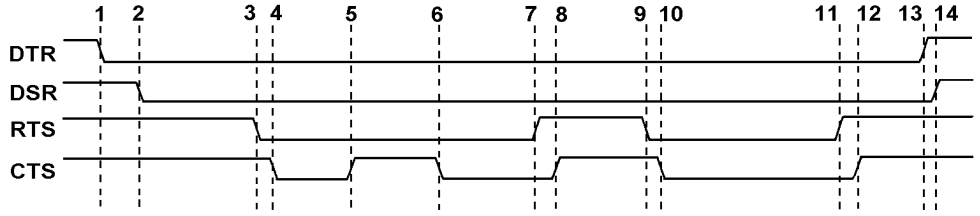

Рис. 7. Последовательность управляющих сигналов интерфейса

1. Установкой DTR компьютер указывает на желание использовать модем.
2. Установкой DSR модем сигнализирует о своей готовности и установлении соединения.
3. Сигналом RTS компьютер запрашивает разрешение на передачу и заявляет о своей готовности принимать данные от модема.
4. Сигналом CTS модем уведомляет о своей готовности к приему данных от компьютера и передаче их в линию.
5. Снятием CTS модем сигнализирует о невозможности дальнейшего приема (например, буфер заполнен) — компьютер должен приостановить передачу данных.
6. Сигналом CTS модем разрешает компьютеру продолжить передачу (в буфере появилось место).
7. Снятие RTS может означать как заполнение буфера компьютера (модем должен приостановить передачу данных в компьютер), так и отсутствие данных для передачи в модем. Обычно в этом случае модем прекращает пересылку данных в компьютер.
8. Модем подтверждает снятие RTS сбросом CTS.
9. Компьютер повторно устанавливает RTS для возобновления передачи.
10. Модем подтверждает готовность к этим действиям.
11. Компьютер указывает на завершение обмена.
12. Модем отвечает подтверждением.
13. Компьютер снимает DTR, что обычно является сигналом на разрыв соединения («повесить трубку»).
14. Модем сбросом DSR сигнализирует о разрыве соединения.

Из рассмотрения этой последовательности становятся понятными соединения DTR-DSR и RTS-CTS в нуль-модемных кабелях.

Асинхронный режим передачи

Асинхронный режим передачи является байт-ориентированным (символьно-ориентированным): минимальная пересылаемая единица информации — один байт (один символ). Формат посылки байта иллюстрирует рис. 8. Передача каждого байта начинается со *старт-бита,* сигнализирующего приемнику о начале посылки, за которым следуют *биты данных* и, возможно, *бит паритета* (контроля четности). Значение бита определяет состояние линии передачи в течение битового интервала, длительность которого определяется скоростью передачи. Завершает посылку *стоп-бит,* гарантирующий паузу между посылками. Старт-бит следующего байта посылается в любой момент после стоп-бита, то есть между передачами

возможны паузы произвольной длительности. Старт-бит, имеющий всегда строго определенное значение (логическая 1), обеспечивает простой механизм синхронизации приемника по сигналу от передатчика. Подразумевается, что приемник и передатчик работают на одной скорости обмена. Внутренний генератор синхронизации приемника использует счетчик-делитель опорной частоты, обнуляемый в момент приема начала старт-бита. Этот счетчик генерирует внутренние стробы, по которым приемник фиксирует последующие принимаемые биты. В идеале стробы располагаются в середине битовых интервалов, что позволяет принимать данные и при незначительном рассогласовании скоростей приемника и передатчика. Очевидно, что при передаче 8 бит данных, одного контрольного и одного стоп-бита предельно допустимое рассогласование скоростей, при котором данные будут распознаны верно, не может превышать 5 %. С учетом фазовых искажений и дискретности работы внутреннего счетчика синхронизации реально допустимо меньшее отклонение частот. Чем меньше коэффициент деления опорной частоты внутреннего генератора (чем выше частота передачи), тем больше погрешность привязки стробов к середине битового интервала, и требования к согласованности частот становятся более строгими. Чем выше частота передачи, тем больше влияние искажений фронтов на фазу принимаемого сигнала.

Взаимодействие этих факторов приводит к повышению требований к согласованности частот приемника и передатчика с ростом скорости обмена.

Рис. 8**. Формат асинхронной передачи RS-232C**

Формат асинхронной посылки позволяет выявлять возможные *ошибки передачи.*

- Если принят перепад, сигнализирующий о начале посылки, а по стробу старт-бита зафиксирован уровень логической единицы, старт-бит считается ложным и приемник снова переходит в состояние ожидания. Об этой ошибке приемник может не сообщать.
- Если во время, отведенное под стоп-бит, обнаружен уровень логического нуля, фиксируется ошибка стоп-бита.
- Если применяется контроль четности, то после посылки бит данных передается *контрольный бит.* Этот бит дополняет количество единичных бит данных до четного или нечетного в зависимости от принятого соглашения. Прием байта с неверным значением контрольного бита приводит к фиксации ошибки.
- Контроль формата позволяет обнаруживать обрыв линии: как правило, при обрыве приемник «видит» логический нуль, который сначала трактуется как старт-бит и нулевые биты данных, но потом срабатывает контроль стоп-бита.

Для асинхронного режима принят ряд *стандартных скоростей обмена*: 50, 75, 110, 150,300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600 и 115200 бит/с. Иногда вместо единицы измерения «бит/с» используют «бод» (baud). В бодах принято измерять частоту изменения состояния линии, что для COM-порта одно и тоже. Однако при недвоичном способе кодирования (широко применяемом в современных модемах) в канале связи скорости передачи бит (бит/с) и изменения сигнала (бод) могут отличаться в несколько раз.

Количество *бит данных* может составлять 5, 6, 7 или 8 (5- и 6-битные форматы распространены незначительно). Количество *стоп-бит* может быть 1, 1,5 или 2 («полтора бита» означает только длительность стопового интервала, измеряемая в битовых интервалах).

Управление потоком данных

Для управления потоком данных (Flow Control) могут использоваться два варианта протокола — аппаратный и программный. Иногда управление потоком путают с квитированием. *Квитирование* (handshaking) подразумевает посылку уведомления о получении элемента, в то время как *управление потоком* предполагает посылку уведомления о возможности или невозможности последующего приема данных. Зачастую управление потоком основано на механизме квитирования.

Аппаратный протокол управления потоком RTS/CTS (hardware flow control) -использует сигнал CTS, который позволяет остановить передачу данных, если приемник не готов к их приему (рис. 9). Передатчик «выпускает» очередной байт только при включенной линии CTS. Байт, который уже начал передаваться, задержать сигналом CTS невозможно (это гарантирует целостность посылки). Аппаратный протокол обеспечивает самую быструю реакцию передатчика на состояние приемника. Микросхемы асинхронных приемопередатчиков имеют не менее двух регистров в приемной части — сдвигающий, для приема очередной посылки, и хранящий, из которого считывается принятый байт. Это позволяет реализовать обмен по аппаратному протоколу без потери данных.

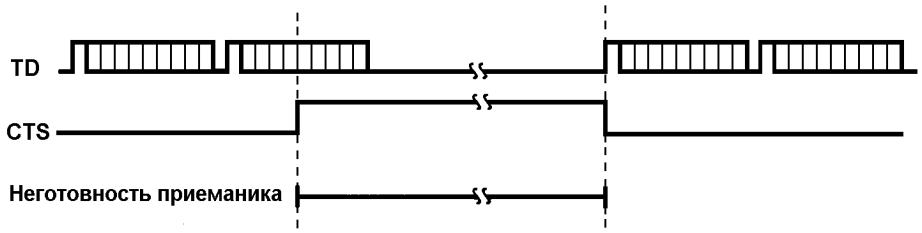

Рис. 9. Аппаратное управление потоком

Аппаратный протокол удобно использовать при подключении принтеров и плоттеров, если они его поддерживают. При непосредственном (без модемов) соединении двух компьютеров аппаратный протокол требует перекрестного соединения линий RTS — CTS. При непосредственном соединении по минимальному количеству линий у передающего терминала должно быть обеспечено состояние «включено» на линии CTS (соединением собственных линий RTS — CTS), в противном случае передатчик будет «молчать».

Применяемые в IBM PC приемопередатчики 8250/16450/16550 сигнал CTS аппаратно не обрабатывают, а только показывают его состояние в регистре MSR. Реализация протокола RTS/CTS возлагается на драйвер BIOS Int 14h, и называть его «аппаратным» не совсем корректно. Если же программа, пользующаяся COM-портом, взаимодействует с UART на уровне регистров (а не через BIOS), то обработкой сигнала CTS для поддержки данного протокола она занимается сама. Ряд коммуникационных программ позволяет игнорировать сигнал CTS (если не используется модем), и для них не требуется соединение входа CTS с выходом даже своего сигнала RTS. Однако существуют и иные приемопередатчики (например, 8251), в которых сигнал CTS отрабатывается аппаратно. Для них, а также для «честных» программ, использование сигнала CTS на разъемах (а то и на кабелях) обязательно. Преимущество протокола RTS/CTS во времени реакции (по сравнению с программным методом XON/XOFF) остается лишь для буферированной (в режиме FIFO) передачи.

Программный протокол управления потоком XON/XOFF предполагает наличие двунаправленного канала передачи данных. Работает протокол следующим образом: если устройство, принимающее данные, обнаруживает причины, по которым оно не может их дальше принимать, оно по обратному последовательному каналу посылает байт-символ XOFF (13h). Противоположное устройство, приняв этот символ, приостанавливает передачу. Когда принимающее устройство снова становится готовым к приему данных, оно посылает символ XON (11h), приняв который противоположное устройство возобновляет передачу. Время реакции передатчика на изменение состояния приемника по сравнению с аппаратным протоколом увеличивается, по крайней мере, на время передачи символа (XON или XOFF) плюс время реакции программы передатчика на прием символа (рис. 10). Из этого следует, что данные без потерь могут приниматься только приемником, имеющим дополнительный буфер принимаемых данных и сигнализирующим о неготовности заблаговременно (имея в буфере свободное место).

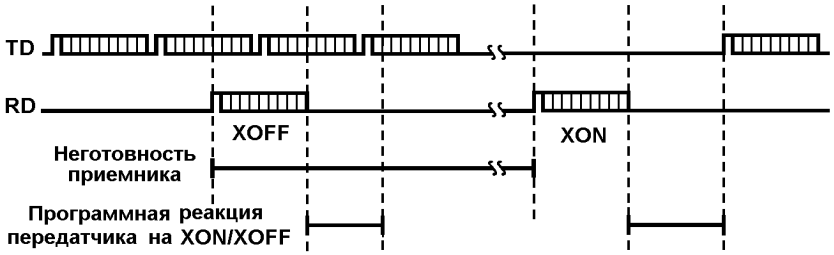

Рис. 10. Программное управление потоком XON/XOFF

Преимущество программного протокола заключается в отсутствии необходимости передачи управляющих сигналов интерфейса — минимальный кабель для двустороннего обмена может иметь только 3 провода (см. рис. 6*а).* Недостатком, помимо обязательного наличия буфера и большего времени реакции (снижающего общую производительность канала из-за ожидания сигнала XON), является сложность реализации полнодуплексного режима обмена. В этом случае из потока принимаемых данных должны выделяться (и обрабатываться) символы управления потоком, что ограничивает набор передаваемых символов.

Кроме этих двух распространенных стандартных протоколов, поддерживаемых и ПУ, и ОС, существуют и другие.

3.2. Микросхемы асинхронных приемопередатчиков (контроллеров)

В COM-портах преобразование параллельного кода в последовательный для передачи и обратное преобразование при приеме данных выполняют специализированные микросхемы UART (Universal Asynchronous Receiver-Transmitter — универсальный асинхронный приемопередатчик - УАПП). Эти же микросхемы формируют и обрабатывают управляющие сигналы интерфейса. COM-порты IBM PC/XT /AT базируются на микросхемах, совместимых на уровне регистров с UART i8250 — 8250/16450/16550A. Это семейство представляет собой усовершенствование начальной модели, направленное на повышение

быстродействия, снижение потребляемой мощности и загрузки процессора при интенсивном обмене. Отметим, что:

- 8250 имеет ошибки (появление ложных прерываний), учтенные в XT BIOS;
- 8250A — ошибки исправлены, но в результате потеряна совместимость с BIOS; эта микросхема работает в некоторых моделях AT, но непригодна для скорости 9600 бит/с;
- 8250B — исправлены ошибки 8250 и 8250A, восстановлена ошибка в прерываниях — возвращена совместимость с XT BIOS; работает в AT под DOS (кроме скорости 9600 бит/с).

Микросхемы 8250x имеют невысокое быстродействие по обращениям со стороны системной шины. Они не допускают обращения к своим регистрам в смежных шинных циклах процессора; для корректной работы с ними требуется введение программных задержек между обращениями CPU.

В компьютерах класса AT применяют микросхемы UART перечисленных ниже модификаций.

- 16450 — быстродействующая версия 8250 для AT. Ошибок 8250 и полной совместимости с XT BIOS не имеет.
- 16550 — развитие 16450. Может использовать канал DMA для обмена данными. Имеет FIFO-буфер, но некорректность его работы не позволяет им воспользоваться.
- 16550A — имеет работающие 16-байтные FIFO-буферы приема и передачи и возможность использования DMA. Именно этот тип UART должен применяться в ПК при интенсивных обменах на скоростях 9600 бит/с и выше. Совместимость с этой микросхемой обеспечивает большинство микросхем контроллеров портов ввода-вывода, входящих в современные чипсеты.

UART (УАПП) — это сложное переплетение логических и синхронизирующих схем, которые автоматизируют процесс выполнения последовательной связи. Точная последовательность операций, выполняемых УАПП в каждой конкретной ситуации, контролируется внешними параметрами. Однако в общих чертах работу УАПП при приеме и передаче можно описать следующим образом.

При передаче символа УАПП должен выполнить следующие операции.

- Принять символ (как данные в параллельной форме) от системного интерфейса ввода-вывода.
- Преобразовать символ в последовательность отдельных битов (параллельно-последовательное преобразование).
- Сформировать кадр путем добавления к информационным разрядам стартового и стопового битов, а также бита четности.
- Передать все биты сформированного кадра на интерфейс с требуемой скоростью (в бодах).
- Сообщить о готовности к передаче следующего символа.
- *Работая в режиме приемника,* УАПП должен выполнить обратную последовательность операций.
- Принять данные в последовательной форме от внешнего интерфейса.
- Проверить правильность структуры кадра: стартовый бит, информационные разряды, бит четности, столовый бит; если нет — выдать сигнал ошибки кадрирования.
- Осуществить проверку четности (независимо от ее типа); если четность нарушена, выдать сигнал ошибки четности.
- Преобразовать информационные разряды в символ. Передать символ (как данные в параллельной форме) в системный интерфейс ввода-вывода.
- Сообщить во внешний интерфейс, что символ принят.

3.2.1. Регистры УАПП 8250, 16450 и 16550.

Микросхемы UART 8250, 16450 и 16550 с программной точки зрения представляют собой набор регистров, доступ к которым определяется адресом (смещением адреса регистра относительно базового

адреса порта) и значением бита DLAB (бита 7 регистра LCR). В адресном пространстве микросхемы занимают 8 смежных адресов. Список регистров УАПП (UART) 16550 и способы доступа к ним приведены в табл. 3. Микросхемы 8250 и 16450 отличаются отсутствием регистра FCR и всех возможностей FIFO и DMA. Определения большинства полей-разрядов регистров микросхемы 16550 идентичны с определениями соответствующих разрядов регистров микросхем 8250 и 16450 как в режиме эмуляции УАПП 8250, так и в FIFO-режиме. Так, форматы перечисленных ниже регистров УАПП 16550 полностью совпадают с форматами регистров микросхем 8250 и 16450.

- Регистр хранения передатчика (РХП) (THR)
- Буферный регистр приемника (БРП) (RBR)
- Регистр управления линией (РУЛ) (LCR)
- Младший байт буфера делителя скорости передачи (МББ) (DLL)
- Старший байт буфера делителя скорости передачи (СББ) (DLM)
- Регистр сверхоперативной памяти (РСП) (CSR)
- Регистр управления модемом (РУМ) (MCR)

- Регистр состояния модема (PCM) (MSR)

В форматы регистров:
- регистр состояния линии, РСЛ (LSR),
- регистр разрешения прерываний, РРП (IER),
- регистр идентификации прерываний, РИП (IIR)

внесены изменения в части переопределения функций некоторых разрядов при работе УАПП в режиме FIFO.

Для обеспечения управления режимами работы УАПП 16550 в его архитектуру был добавлен один новый регистр - регистр управления FIFO-режимом (FCR).

Таблица 3. Регистры UART 16550A

Адрес (Смещение)	DLAB	Регистр Имя	Название	Чтение/запись R/W
0h	0	THR	Transmit Holding Register	WO
0h	0	RBR	Receiver Buffer Register	RO
0h	1	DLL	Divisor Latch LSB	R/W
1h	1	DLM	Divisor Latch MSB	R/W
1h	0	IER	Interrupt Enable Register	R/W
2h	X	IIR	Interrupt Identification Register	RO
2h	X	FCR	FIFO Control Register	WO
3h	X	LCR	Line Control Register	R/W
4h	X	MCR	Modem Control Register	R/W
5h	X	LSR	Line Status Register	R/W[1]
6h	X	MSR	Modem Status Register	R/W[1]
7h	X	SCR	Scratch Pad Register	R/W

[1] Некоторые биты допускают только чтение. Запись в регистр может привести к сбою протокола.

X - значение разряда DLAB безразлично.

RO - только чтение.

WO - только запись.

R/W - и запись и чтение.

Ниже приводится описание регистров UART 8250, 16450 и 16550 и их форматов.

THR *— промежуточный регистр данных передатчика* (только для записи). Данные, записанные в регистр, будут пересланы в выходной сдвигающий регистр TSR (когда он будет свободен), из которого поступят на выход TD. Бит 0 передается первым. При длине посылки менее 8 бит старшие биты игнорируются.

RBR — *буферный регистр принимаемых данных* (только для чтения). Данные, принятые входным сдвигающим регистром RSR, помещаются в регистр RBR, откуда они могут быть считаны процессором. Если к моменту окончания приема очередного символа предыдущий символ не был считан из регистра RBR, фиксируется ошибка переполнения. При длине посылки менее 8 бит старшие биты в регистре имеют нулевое значение.

DLL — *регистр младшего байта буфера делителя частоты.*

DLM *— регистр старшего байта буфера делителя частоты.* Содержимое делителя (DLL и DLM) в десятичном виде определяется по формуле D=115200/V, где V — скорость передачи, бит/с. В рамках архитектуры PC AT использовалась входная частота синхронизации 1,8432 МГц, поступавшая в контроллер, делилась на заданный коэффициент D, после чего получалась 16-кратная скорость передачи данных (V×16). В регистры делителя всегда заносится шестнадцатеричный эквивалент коэффициента D. В таблице 4 приведены рекомендуемые скорости обмена COM порта и соответствующие значения десятичной и шестнадцатеричной константы деления D.

Фирма IBM выпускала высокоскоростные контроллеры последовательного порта, которые использовали входную частоту синхронизации 11,0592 МГц, и были способны передавать данные со скоростями до 345 600 бод.

Следует отметить, что для контроллера 16550A и его аналогов в рамках чипсетов существует недокументированная возможность дополнительного программного увеличения скорости обмена по сравнению с представленными в табл. 4 стандартными значениями. Эта возможность реализуется записью в регистр состояния линии контроллера (LSR) определенных управляющих байтов (например 80h). Используя дополнительное изменение кратности деления частоты можно расширить диапазон скоростей обмена, устанавливаемых функцией 00h драйвера COM порта INT 14h, который обычно ограничен значением 9600 бод.

Таблица 4. Соответствие скорости обмена и констант деления.

Требуемая скорость приема/передачи	Значение делителя, соответствующее требуемой скорости	
	Десятичное	Шестнадцатеричное
50	2304	0900h
75	1536	0600h
150	768	0300h
300	384	0180h
600	192	00C0h
1200	96	0060h
2400	48	0030h
4800	24	0018h
7200	16	0010h
9600	12	000Ch
19200	6	0006h
38400	3	0003h
57600	2	0002h
115200	1	0001h

IER — регистр разрешения прерываний. Единичное значение бита разрешает прерывание от соответствующего источника.

Назначение бит регистра IER:

- биты [7:4]=0 — не используются;
- бит 3 — Mod_IE — по изменению состояния модема (любой из линий CTS, DSR, RI, DCD);
- бит 2 — RxL_IE — по обрыву/ошибке линии;
- бит 1 — TxD_IE — по завершению передачи;
- бит 0 — RxD_IE — по приему символа (в режиме FIFO — прерывание по временной задержке).

В FIFO-режиме установка бита 0 в 1 разрешает прерывание по причине временной задержки в работе FIFO-буфера приемника. Если это прерывание разрешено, то оно будет происходить при одновременном выполнении следующих условий.

- Хотя бы один символ находится в FIFO-буфере приемника.
- Текущий символ не принят за время, превышающее длительность четырех непрерывно следующих друг за другом символов.
- Информация из FIFO-буфера приемника не считывалась процессором в течение времени, превышающего длительность четырех непрерывно следующих друг за другом символов.

Длительность символа измеряется числом синхронизирующих импульсов контроллера-генератора скорости передачи, так что пороговый уровень задержки не зависит от скорости передачи. Если условия этого прерывания не реализуются, то таймер задержки сбрасывается в исходное состояние каждый раз, когда символ принимается или считывается из FIFO-буфера приемника.

IIR — *регистр идентификации прерываний и признака режима FIFO* (только для чтения). Для упрощения программного анализа UART выстраивает внутренние запросы прерывания по четырехуровневой системе приоритетов. Порядок приоритетов (по убыванию): состояние линии, прием символа, освобождение регистра передатчика, состояние модема. При возникновении условий прерывания UART указывает на источник с высшим приоритетом до тех пор, пока он не будет сброшен соответствующей операцией. Только после этого будет выставлен запрос с указанием следующего источника. Ниже объясняется назначение бит регистра *IIR*.

- Биты [7:6] — признак режима FIFO:
  - 11 — режим FIFO 16550A;
  - 10 — режим FIFO 16550;
  - 00 — обычный.
- Биты [5:4] — не используются.
- Бит 3 — прерывание по временной задержке приема в режиме FIFO (в буфере есть символы для считывания).
- Биты [2:1] — причина прерывания с наивысшим приоритетом (в обычном, не FIFO-режиме):
  - 11 — ошибка/обрыв линии, сброс выполняется чтением регистра состояния линии;
  - 10 — принят символ, сброс выполняется чтением данных;
  - 01 — передан символ (регистр THR пуст), сброс выполняется записью данных;
  - 00 — изменение состояния модема; сброс выполняется чтением регистра состояния модема.
- Бит 0 — признак необслуженного запроса прерывания (1 — нет запроса, 0 — есть запрос).

В режиме FIFO причину прерывания идентифицируют биты [3:1].

- 011 — ошибка/обрыв линии. Сброс выполняется чтением регистра состояния линии.

- 010 — принят символ. Сброс выполняется чтением регистра данных приемника.
- 110 — индикатор временной задержки (за 4-кратный интервал времени символа не передано и не принято ни одного символа, хотя в буфере имеется, по крайней мере, один). Сброс выполняется чтением регистра данных приемника.
- 001 — регистр THR пуст. Сброс выполняется записью данных.
- 000 — изменение состояния модема (CTS, DSR, RI или DCD). Сброс выполняется чтением регистра *MSR*.

FCR *— регистр управления FIFO* (только для записи). Ниже описано назначение бит регистра FCR:

- Биты [7:6] — ITL (Interrupt Trigger Level) — уровень заполнения FIFO-буфера, при котором вырабатывается прерывание:
  - 00 — 1 байт (по умолчанию);
  - 01 — 4 байта;
  - 10 — 8 байт;
  - 11 — 14 байт.
- Биты [5:4] зарезервированы.
- Бит 3 — разрешение операций DMA.
- Бит 2 — RESETTF (Reset Transmitter FIFO) — сброс счетчика FIFO-передатчика (записью единицы; сдвигающий регистр не сбрасывается).
- Бит 1 — RESETRF (Reset Receiver FIFO) — сброс счетчика FIFO-приемника (записью единицы; сдвигающий регистр не сбрасывается).
- Бит 0 — TRFIFOE (Transmit And Receive FIFO Enable) — разрешение (единицей) режима FIFO для передатчика и приемника. При смене режима FIFO-буферы автоматически очищаются.

LCR *— регистр управления линией* (настройки параметров канала). Ниже описано назначение бит регистра *LCR*.

- Бит 7 — DLAB (Divisor Latch Access Bit) — управление доступом к делителю частоты.
- Бит 6 — BRCON (Break Control) — формирование обрыва линии (посылка нулей) при BRCON=1.
- Бит 5 — STICPAR (Sticky Parity) — принудительное формирование бита паритета:
  - 0 — контрольный бит генерируется в соответствии с паритетом выводимого символа;
  - 1 — постоянное значение контрольного бита: при EVENPAR=1 — нулевое, при EVENPAR=0 — единичное.
- Бит 4 — EVENPAR (Even Parity Select) — выбор типа контроля: 0 — нечетность, 1 — четность.
- Бит 3 — PAREN (Parity Enable) — разрешение контрольного бита:
  - 1 — контрольный бит (паритет или постоянный) разрешен;
  - 0 — контрольный бит запрещен.
- Бит 2 — STOPB (Stop Bits) — количество стоп-бит:
  - 0 — 1 стоп-бит;
  - 1 — 2 стоп-бита (для 5-битного кода стоп-бит будет иметь длину 1,5 бит).
- Биты [1:0] — SERIALDB (Serial Data Bits) — количество бит данных:
  - 00 — 5 бит;
  - 01 — 6 бит;
  - 10 — 7 бит;
  - 11 — 8 бит.

MCR *— регистр управления модемом*. Ниже описано назначение бит регистра MCR.

- Биты [7:5]=0 — зарезервированы.
- Бит 4 — LME (Loopback Mode Enable) — разрешение режима диагностики:
  - 0 — нормальный режим;
  - 1 — режим диагностики (см. ниже).
- Бит 3 — IE (Interrupt Enable) — разрешение прерываний с помощью внешнего выхода OUT2; в режиме диагностики поступает на вход MSR. 7:
  - 0 — прерывания запрещены;
  - 1 — прерывания разрешены.
- Бит 2 — OUT1C (OUT1 Bit Control) — управление выходным сигналом 1 (в PC не используется); в режиме диагностики поступает на вход MSR. 6.
- Бит 1 — RTSC (Request To Send Control) — управление выходом RTS; в режиме диагностики поступает на вход MSR. 4:
  - 0 — активен (-V);
  - 1 — пассивен (+V).
- Бит 0 — DTRC (Data Terminal Ready Control) — управление выходом DTR; в режиме диагностики поступает на вход MSR. 5:

- 0 — активен (-V);
- 1 — пассивен (+V).

LSR *—регистр состояния линии* (точнее, состояния приемопередатчика). Ниже описано назначение разрядов регистра LSR.

- Бит 7 — FIFOE (FIFO Error Status) —ошибка принятых данных в режиме FIFO (буфер содержит хотя бы один символ, принятый с ошибкой формата, паритета или обрывом). В не FIFO-режиме всегда 0.
- Бит 6 — TEMPT (Transmitter Empty Status) — 1 - регистр передатчика пуст (нет данных для передачи ни в сдвиговом регистре, ни в буферных регистрах THR или FIFO).
- Бит 5 — THRE (Transmitter Holding Register Empty) — 1 - регистр передатчика готов принять байт для передачи. В режиме FIFO указывает на отсутствие символов в FIFO-буфере передачи. Может являться источником прерывания.
- Бит 4 — BD (Break Detected) — индикатор обрыва линии (вход приемника находится в состоянии 0 не менее чем время посылки символа).
- Бит 3 — FE (Framing Error) — ошибка кадра (неверный стоп-бит).
- Бит 2 — PE (Parity Error) — ошибка контрольного бита (паритета или фиксированного).
- Бит 1 — OE (Overrun Error) — переполнение (потеря символа). Если прием очередного символа начинается до того, как предыдущий выгружен из сдвигающего регистра в буферный регистр или в регистр FIFO, прежний символ в сдвигающем регистре теряется.
- Бит 0 — DR (Receiver Data Ready) — 1 - принятые данные готовы (в DHR или FIFO-буфере). Сброс — чтением приемника.

Индикаторы ошибок — биты [4:1] — сбрасываются после чтения регистра LSR. В режиме FIFO признаки ошибок хранятся в FIFO-буфере вместе с каждым символом. В регистре они устанавливаются (и вызывают прерывание) в тот момент, когда символ, принятый с ошибкой, находится на вершине FIFO (первый в очереди на считывание). В случае обрыва линии в FIFO заносится только один «обрывной» символ, и UART ждет восстановления и последующего старт-бита.

Примечание: Регистр LSR может быть использован для реализации недокументированной возможности некоторых реализаций контроллеров COM портов - увеличение скорости передачи в 16 раз по сравнению с устанавливаемой в счетчике делителя контроллера. Эта возможность реализуется записью в регистр LSR константы 80h. Скорость восстанавливается записью в LSR константы 00h.

MSR — *регистр состояния модема.* Ниже описано назначение бит регистра MSR:

- Бит 7 — DCD (Data Carrier Detect) — состояние линии DCD:
  - 0 — активна (-V);
  - 1 — пассивна (+V).
- Бит 6 — RI (Ring Indicator) — состояние линии RI:
  - 0 — активна (-V);
  - 1 — пассивна (+V).
- Бит 5 — DSR (Data Set Ready) — состояние линии DSR:
  - 0 — активна (-V);
  - 1 — пассивна (+V).
- Бит 4 — CTS (Clear To Send) — состояние линии CTS:
  - 0 — активна (-V);
  - 1 — пассивна (+V).
- Бит 3 — DDCD (Delta Data Carrier Detect) — изменение состояния DCD.
- Бит 2 — TERI (Trailing Edge Of Ring Indicator) — спад огибающей RI (окончание звонка).
- Бит 1 — DDSR (Delta Data Set Ready) — изменение состояния DSR.
- Бит 0 — DCTS (Delta Clear To Send) — изменение состояния CTS.

Признаки изменения (биты [3:0]) сбрасываются по чтению регистра.

SCR *— рабочий регистр* (8 бит), на работу UART не влияет, предназначен для временного хранения данных (в 8250 отсутствует).

В *диагностическом режиме* (при LME=1 в регистре MCR) внутри UART организуется внутренняя «заглушка»:

- выход передатчика переводится в состояние логической единицы;
- вход приемника отключается;
- выход сдвигающего регистра передатчика логически соединяется со входом приемника;
- входы DSR, CTS, RI и DCD отключаются от входных линий и внутренне управляются битами DTRC, RTSC, OUT1C, IE;
- выходы управления модемом переводятся в пассивное состояние (логический ноль).

Переданные данные в последовательном виде немедленно принимаются, что позволяет проверять внутренний канал данных порта (включая сдвигающие регистры) и отработку прерываний, а также

определять скорость работы UART.

3.3. Системная поддержка COM портов

COM-порты на уровне BIOS поддерживаются следующими средствами:

- процедурами POST на стадии инициализации и тестирования;
- сервисом BIOS Int 14h (драйвер COM-порта).

Системная BIOS не поддерживает аппаратные прерывания RS232, хотя имеются два вектора прерываний: IRQ4 (INT 0Ch) для RS232-1 и IRQ3 (INT 0Bh) для RS232-2.

В процессе начального тестирования POST BIOS проверяет наличие последовательных портов (регистров UART 8250 или совместимых) по стандартным адресам и помещает базовые адреса обнаруженных портов в ячейки BIOS Data Area (ячейки памяти рабочей области данных BIOS) 0:0400, 0402, 0404, 0406. Эти ячейки хранят адреса портов с логическими именами COM1 - COM4 (в сервисе Int 14h - логические номера 0 - 3). Нулевое значение адреса является признаком отсутствия порта с данным номером. При программировании RS232 необходимо пользоваться не жестко привязанными адресами, а адресами, извлеченными из области данных BIOS. В ячейки памяти 0:047C, 047D, 047E, 047F заносятся константы, задающие тайм-ауты для соответствующих портов. Значения этих тайм-аутов инициализируются на 1 при выполнении POST и могут быть установлены пользователями до значения 255 единиц. Каждая единица соответствует приблизительно 900 мс. Обнаруженные порты *инициализируются* на скорость обмена 2400 бит/с, 7 бит данных с контролем на четность (even), 1 стоп-бит. Управляющие сигналы интерфейса DTR и RTS и линия передачи данных TD переводятся в исходное состояние («включено» — отрицательное напряжение -12 В на соответствующих контактах интерфейсного разъема).

Сервис BIOS Int 14h обеспечивает выполнение следующих функций:

- *инициализация* (установка скорости обмена и формата посылок; запрет источников прерываний) — на сигналы DTR и RTS влияния не оказывает (после аппаратного сброса они пассивны);
- *вывод символа* — активируются сигналы DTR и RTS, и после освобождения регистра THR в него помещается выводимый символ;
- *ввод символа* — активируется только сигнал DTR (RTS переходит в пассивное состояние), и ожидается готовность принятых данных;
- *опрос состояния* модема и линии (чтение регистров MSR и LSR).

Ожидание готовности ввода и вывода ограничивается по тайм-ауту. Готовность проверяется опросом состояния.

В таблице 5 приведены системные ресурсы, выделяемые под стандартные COM-порты в ПК типа IBM PC/AT.

Таблица 5. Ячейки памяти рабочей области данных BIOS, используемые для COM-портов.

Ячейка памяти	Байт	Функция
0000:002C	4	Вектор INT 0Bh
0000:0030	4	Вектор INT 0Ch
0000:0050	4	Вектор INT 14h
0040:0000	2	Адрес базового порта RS232-1
0040:0002	2	Адрес базового порта RS232-2
0040:0004	2	Адрес базового порта RS232-3
0040:0006	2	Адрес базового порта RS232-4
0040:0071	1	Break bit
0040:007C	1	Тайм-аут для RS232-1
0040:007D	1	Тайм-аут для RS232-2
0040:007E	1	Тайм-аут для RS232-3
0040:007F	1	Тайм-аут для RS232-4

3.3.1. Драйвер COM-порта Int 14h.

Int 14h вызывается для выполнения всех функций, относящихся к вводу-выводу через RS232. BIOS указывает на точку входа в ПЗУ драйвера RS232.

ДЕЙСТВИЕ: после входа управление передается одной из четырех программ в соответствии с кодом функции в регистре AH. При использовании запрещенного кода функции происходит простой возврат управления. Сохраняются все регистры, кроме тех, которые возвращают значение.

НАЗНАЧЕНИЕ: прикладная программа может использовать Int 14h для прямого выполнения функций ввода-вывода через RS232 COM-порты).

По команде Int 14h выполняются следующие функции:

- 00h - ницализировать RS232;
- 01h - передать символ;
- 02h - принять символ;
- 03h - определить состояние.

При выполнениии функций прерывания остаются разрешенными. Маска для прерываний интерфейса и порт управления для прерываний RS232 не изменяются.

Перед выполнением команды Int14h необходимо занести определенную информацию в регистры процессора в соответствии с требованиями вызываемой функции. Результаты выполнения функции, если это предусмотрено функцией возвращаются программе также через регистры процессора.

Функция 00h: Инициализировать RS232

В функции указывается длина слова, число стоповых разрядов, тип паритета и скорость передачи для заданного порта.

Хотя BIOS поддерживает четыре COM-порта, при инициализации после включения питания BIOS ищет только порты 1 и 2. Для использования портов 3 и 4 пользователь должен установить базовые адреса интерфейсного порта в ячейки памяти 0040:0004h и 0040:0006h соответственно.

ВХОДНЫЕ ПАРАМЕТРЫ:

AH = 00h,
AL - конфигурация порта,
DX - номер RS232 (0 - 3).

ВЫХОДНЫХ ПАРАМЕТРОВ не предусмотрено.

В таблице 6 приведено назначение разрядов байта конфигурации (регистр AL).

Таблица 6. Байт конфигурации.

Разряды AL							
7	6	5	4	3	2	1	0
Скорость передачи: 000 - 110 бод 001 - 150 бод 010 - 300 бод 011 - 600 бод 100 - 1200 бод 101 - 2400 бод 110 - 4800 бод 111 - 9600 бод			Паритет: 00 - отсутствует 01 - нечетный 10 - отсутствует 11 - четный		Стоповые биты: 0 - 1 бит 1 - 2 бита	длина слова: 00 - 5 бит 01 - 6 бит 10 - 7 бит 11 - 8 бит	

Функция 01h Передать символ

INT 14h, AH = 01h выводит символ на заданный RS232.

ВХОДНЫЕ ПАРАМЕТРЫ:

AH = 01h,
AL - символ, который необходимо передать (00h.-.FFh),
DX - номер RS232 (0 - 3).

ВЫХОДНЫЕ ПАРАМЕТРЫ:

AH - состояние линии (см. определение для INT 14h, AH = 03h).

Функция 02h Принять символ

INT 14h, AH = 02h принимает символ с заданного RS232.

ВХОДНЫЕ ПАРАМЕТРЫ:

AH = 02h,
DX - номер RS232 (0 - 3).

ВЫХОДНЫЕ ПАРАМЕТРЫ:

AH - состояние линии (определение см. INT 14h, AH = 03h),
AL - символ, который принят (00h - FFh).

Функция 03h Определить состояние интерфейса

INT 14h, AH = 03h считывает регистры модема и состояния линии заданного RS232 и помещает эти значения в выходные регистры.

ВХОДНЫЕ ПАРАМЕТРЫ:

AH = 03h,
DX - номер RS232 (0 - 3).

ВЫХОДНЫЕ ПАРАМЕТРЫ:

AH - состояние линии,
AL - состояние модема.

В таблицах 7 и 8 приведено описание разрядов слов состояния линии и модема соответственно.

Таблица 7. Слово состояния линии.

Разряды AH							
7	6	5	4	3	2	1	0
1 - тайм-аут	1 - передатчик пуст	1 - готов к передаче	1 – пауза	1 - ошибка стоп разряда	1 - ошибка паритета	1 - пере-полнение	1 - готовы данные

Таблица 8. Слово состояния модема.

Разряды AL							
7	6	5	4	3	2	1	0
1 - (DCD) детектор принимаемого сигнала (цепь 109)	1 - (RI) индикатор вызова (цепь 125)	1 - (DSR) аппаратура передачи данных готова (цепь 107)	1 – (CTS) готов к передаче (цепь 106)	0	0	0	0

3.4. Конфигурирование COM-портов

Компьютер может иметь до четырех последовательных портов COM1-COM4; для машин класса AT типично наличие двух портов. Управление последовательным портом разделяется на два этапа — предварительное конфигурирование (Setup) аппаратных средств порта и текущее (оперативное) переключение режимов работы прикладным или системным ПО. Конфигурирование COM-порта зависит от его исполнения. Порт на плате расширения конфигурируется джамперами на самой плате. Порт на системной плате конфигурируется через BIOS Setup.

Конфигурированию подлежат перечисленные ниже параметры:

- *Базовый адрес,* который для портов COM1-COM4 обычно имеет значение 3F8h, 2F8h, 3E8h и 2E8h. При инициализации BIOS проверяет наличие портов по адресам именно в этом порядке и присваивает обнаруженным портам •логические имена COM1, COM2, COM3 и COM4. Для портов COM3 и COM4 возможны альтернативные адреса 3E0h, 338h и 2E0h, 238h соответственно. Для PS/2 стандартными для портов COM3-COM8 являются адреса 3220h, 3228h, 4220h, 4228h, 5220h и 5228h соответственно.
- Используемая *линия запроса прерывания:* для COM 1 и COM3 обычно используется IRQ4 или IRQ11, для COM2 и COM4 — IRQ3 или IRQ10. В принципе номер прерывания можно назначать в произвольных сочетаниях с базовым адресом (номером порта), но некоторые программы и драйверы (например, драйверы последовательной мыши) настроены на стандартные сочетания. Каждому порту, нуждающемуся в аппаратном прерывании, назначают отдельную линию, не совпадающую с линиями запроса прерываний других устройств. Прерывания необходимы для портов, к которым подключаются устройства ввода, UPS (источники бесперебойного питания) или модемы. При подключении принтера или плоттера прерываниями пользуются только многозадачные ОС (не всегда), и этот дефицитный ресурс PC можно сэкономить. Также прерывания обычно не задействуют при связи двух компьютеров нуль-модемным кабелем. Возможность разделяемого использования одной линии запроса несколькими портами (или ее разделения с другими устройствами) зависит от реализации аппаратного подключения и ПО. При использовании портов, установленных на шину ISA, разделяемые прерывания обычно не работают.
- *Канал DMA* (для микросхем UART 16450/16550, расположенных на системной плате) — разрешение использования и номер канала DMA. Режим DMA при работе с COM портами используют редко.

4. Указания по подготовке к лабораторной работе.

При подготовке к лабораторной работе, ее выполнении и оформлении отчета необходимо следовать рекомендациям раздела 4.1."Общие указания по выполнению и оформлению лабораторных работ".

Подготовка к данной лабораторной работе заключается в изучении описания лабораторной работы "Последовательный асинхронный порт ПК (COM-порт)" и раздела 2.1.2. "Интерфейс Serial Mouse" учебного пособия. Необходимо также ознакомиться с основными командами отладчика AFD и с командами процессора i8086.

5. Порядок выполнения лабораторной работы.

1. Ознакомьтесь с программой 1 (Comp1.txt), представленной в листинге 1 раздела 6 описания лабораторной работы, выделите в ней участки, реализующие инициализацию контроллера COM-порта и передачу символа в линию передачи данных интерфейса. Объясните этапы и параметры инициализации и алгоритм передачи потока символов.
2. В окне дизассемблера отладчика AFD наберите программу Comp1.txt, проверьте и запустить ее командой "G" отладчика. С помощью электронного осциллографа проконтролируйте сигналы на линии передаваемых данных порта (контакт 03 разъема порта). Определите параметры наблюдаемых сигналов (длительность битовых интервалов, время передачи одного символа, битовую скорость передачи) и идентифицируйте элементы формата передачи символа 05h (стартовый бит, символ, стоповые битовые интервалы и интервал паузы) используя информацию, представленную на рис. 11 описания лабораторной работы.

3. Выполните пункт 2 задания при передаваемом символе 55h. Сформулируйте соответствующие выводы.
4. Модифицируйте программу Comp1.txt так, чтобы в наблюдаемой последовательности сигналов отсутствовал интервал паузы (см. рис. 11).
5. Командой "O" отладчика занесите по адресу регистра LSR (3FDh) константу 80h, запустите программу Comp1.txt и определите по осциллографу временные параметры передаваемых сигналов, сравните их с параметрами, полученными на этапе 2 выполнения работы, и сформулируйте соответствующие выводы. После этого занесите в LSR константу 00h и проверьте факт восстановления временных параметров передачи символов программой Comp1.txt.
6. В программе Comp1.txt измените параметры инициализации контроллера порта в соответствии с заданием преподавателя и зарегистрируйте временные параметры передачи символов. Определите соответствие этих параметров заданному режиму работы контроллера COM-порта.
7. Ознакомьтесь с программой 2 (Comit1.txt), представленной в листинге 2 раздела 6 описания лабораторной работы, выделите в ней участки, реализующие инициализацию контроллера COM-порта и передачу символа в линию передачи данных интерфейса. Объясните этапы и параметры инициализации и алгоритм передачи потока символов.
8. В окне дизассемблера отладчика AFD наберите программу Comit1.txt, проверьте и запустить ее командой "G" отладчика. С помощью электронного осциллографа проконтролируйте сигналы на линии передаваемых данных порта (контакт 03 разъема порта). Определите параметры наблюдаемых сигналов (длительность битовых интервалов, время передачи одного символа, битовую скорость передачи) и идентифицируйте элементы формата передачи символа 05h (стартовый бит, символ, стоповые битовые интервалы и интервал паузы). Сравните результаты с соответствующими параметрами, полученными на этапе 2 выполнения работы и сформулируйте соответствующие выводы. Если в процессе работы программы сигналы на контакте 3 разъеме порта COM 1 не наблюдаются, то внимательно прочтите пояснения к программе Comit1.txt. Объясните, почему программа работает, а данные не выдаются?
9. Выполните для программы Comit1.txt задание пункта 5 и определите максимальную скорость передачи данных функциями драйвера BIOS COM-порта. Сформулируйте соответствующие выводы.
10. Выполните для программы Comit1.txt задание пункта 6 и сформулируйте соответствующие выводы.
11. Соедините между собой выводы контактов 3 и 2 разъема порта COM 1. В окне дизассемблера отладчика AFD наберите программу Comp2.txt, проверьте и запустите ее командой "G" отладчика. После остановки программы проверьте содержимое регистра AL. В нем должна находиться константа 05h. В программе Comp2.txt измените передаваемую константу на 55h и запустите ее. Какая константа зафиксировалась в регистре AL и почему?
12. В программу Comp2.txt внесите такие изменения, чтобы без искажений передавалась и принималась константа FFh.
13. Командой "O" отладчика занесите по адресу регистра LSR (3FDh) константу 80h и запустите модифицированную на этапе 12 программу Comp2.txt. Если константа поступит в регистр AL без искажений, то определить при какой максимальной скорости передачи константа будет предаваться без искажений. После этого командой "O" занесите в LSR константу 00h и проверьте факт восстановления временных параметров передачи и приема символов программой Comp2.txt.
14. Выполните задание пункта 11 для программы Comit2.txt и сравните результаты.
15. Выполните задания пунктов 12 и 13 для программы Comit2.txt и сравните результаты.

16. Соединить между собой COM-порты двух компьютеров согласно информации, приведенной на рис. 6а в разделе 3.1 описания лабораторной работы. На одном из соединенных компьютеров в среде отладчика AFD наберите инициализирующую и передающую часть программы Comit3.txt, а на другом - инициализирующую и принимающую часть этой программы. Запустите сначала программу приемника а затем передатчика и проверьте, принялся ли переданный символ в регистр AL приемника. Попробуйте передать символ запуская сначала передатчик а затем приемник. Какие изменения в программе и в конфигурации соединения необходимо внести, чтобы передача осуществлялась в любом случае.
17. Соединить между собой COM-порты двух компьютеров согласно информации, приведенной на рис. 6б в разделе 3.1 описания лабораторной работы. На одном компьютере в среде отладчика AFD набрать программу 6 (Comt1.bin), а на другом - программу 7 (Comr1.bin). Описание этих программ и их листинги представлены в разделе 6 описания лабораторной работы. С помощью осциллографа определить параметры сигналов на линиях интерфейса, указанных на рис.12 раздела 6 описания.

6. Примеры программ

Программа 1(Comp1.txt), представленная в листинге 1, реализует установку режима работы контроллера порта COM 1 со следующими параметрами:

- длина слова - 6 бит;
- контроль четности запрещен;
- 2 стоповых бита;
- скорость передачи - 2400 бод;
- генерация сигнала прерывания запрещена;
- прерывания по всем событиям в контроллере запрещены;
- диагностический режим запрещен.

После установки режимов программа переходит к выполнению передачи константы 05h в линию данных порта без контроля сигналов квитирования и управления потоком. Число передаваемых байт задается содержимым регистра BX.

На рис.11 представлена временная диаграмма процесса передачи шестиразрядного кода 05h, наблюдаемая на выводе 03 разъема последовательного порта COM 1. Согласно правилам интерфейса RS-232C передача битов данных производится двуполярным потенциальным кодом с использованием стартового и стоповых битов. Нулевым битам соответствует высокий уровень сигнала (10 В в нашем случае), единичным - низкий уровень (-10 В в нашем случае).

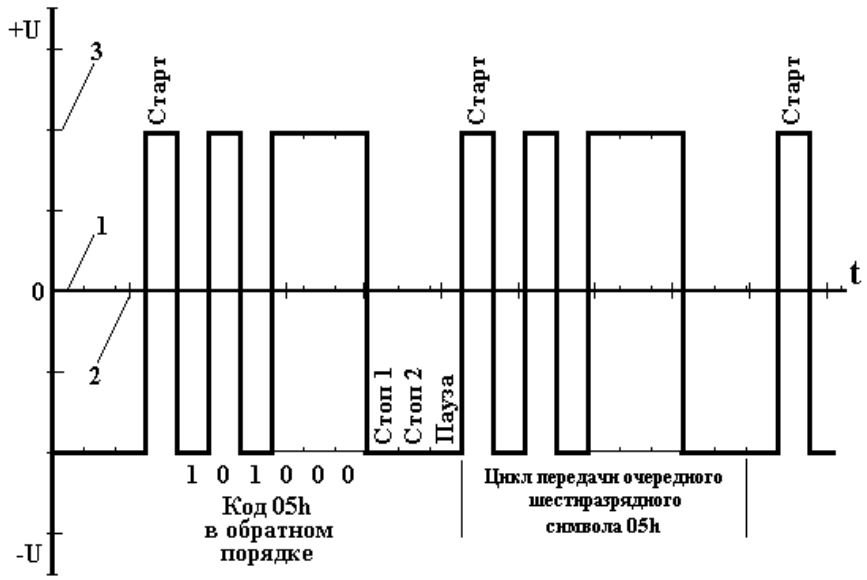

Рис. 11. Передача кода 05h, реализуемая программой 1: 1 - битовый интервал = 0,417мс; 2 - деления масштабной сетки по времени - 1 мс/деление; 3 - деления масштабной сетки по амплитуде - 5 В/деление.

Примечание. Во временной диаграмме присутствует пауза, которая формируется контроллером когда сдвиговый регистр передатчика освободился и буферный регистр данных передатчика пуст. Паузы можно избежать, если запись очередного байта в буферный регистр осуществлять не по освобождению сдвигового

регистра (D6=0 регистра LSR, как реализовано в программе 1) а по признаку освобождения буферного регистра передатчика (D5=0 регистра LSR).

Листинг 1. Инициализация COM 1 и передача в линию константы 05h.

```
2BF9:0100  BBFF2F  MOV    BX,2FFF   ;Число циклов передач.
2BF9:0103  BAFB03  MOV    DX,03FB   ;В регистре LCR устанавливается разряд D7
2BF9:0106  B080    MOV    AL,80     ;(DLAB), разрешающий обмен с регистрами
2BF9:0108  EE      OUT    DX,AL     ;делителя(DLL и DLM).
2BF9:0109  B030    MOV    AL,30     ;константу 30h заносим в DLL,
2BF9:010B  BAF803  MOV    DX,03F8   ;и константу 00h - в MLB,
2BF9:010E  EE      OUT    DX,AL     ;задавая скорость
2BF9:010F  B000    MOV    AL,00     ;передачи
2BF9:0111  BAF903  MOV    DX,03F9   ;величиной 2400 бод (см.
2BF9:0114  EE      OUT    DX,AL     ;таблицу 4 пункта 3.2).
2BF9:0115  BAFB03  MOV    DX,03FB   ;05h заносится в LCR, устанавливая:
2BF9:0118  B005    MOV    AL,05     :размер посылки - 6 бит, 2 стоп-бита и
2BF9:011A  EE      OUT    DX,AL     ;запрет контроля четности.
2BF9:011B  B000    MOV    AL,00     ;Занесением 00h в IER запрещаем
2BF9:011D  BAF903  MOV    DX,03F9   ;прерывания по всем событиям, могущим
2BF9:0120  EE      OUT    DX,AL     ;вызвать прерывание.
2BF9:0121  BAFC03  MOV    DX,03FC   ;Занесением 00h в IER запрещаем сигнал
2BF9:0124  EE      OUT    DX,AL     ;прерывания и режим диагностики.
2BF9:0125  BAFD03  MOV    DX,03FD   ;Чтение LSR и проверка: пусты ли регистры
2BF9:0128  EC      IN     AL,DX     ;передатчика
2BF9:0129  A840    TEST   AL,40     ;если передача не закончена (D6=0), то
2BF9:012B  74F8    JZ     0125      ;на новый цикл проверки.
2BF9:012D  BAF803  MOV    DX,03F8   ;В регистр передатчика заносится
2BF9:0130  B005    MOV    AL,05     ;константа 05h, передаваемая
2BF9:0132  EE      OUT    DX,AL     ;затем в линию.
2BF9:0133  B90200  MOV    CX,0002   ;Вложенные циклы программной
2BF9:0136  51      PUSH   CX        ;задержки, позволяющие
2BF9:0137  B92000  MOV    CX,0020   ;устанавливать временные
2BF9:013A  E2FE    LOOP   013A      ;промежутки между
2BF9:013C  59      POP    CX        ;передаваемыми
2BF9:013D  E2F7    LOOP   0136      ;байтами.
2BF9:013F  4B      DEC    BX        ;Декремент счетчика передаваемых байт.
2BF9:0140  75E3    JNZ    0125      ;Если не ноль, то очередная передача.
2BF9:0142  CC      INT3             ;Передача управления отладчику.
```

Программа 2(Comit1.bin), представленная в листинге 2, реализует те же действия и с теми же параметрами, что и программа 1, но при этом использует функции драйвера COM-порта BIOS INT 14h (см. раздел 3.3 описания). Следует отметить, что функции передачи и приема данных реализуют протокольные правила интерфейса RS-232C в части генерации и контроля сигналов квитирования и управления потоком. В связи с этим для корректной работы программы 2 необходимо скоммутировать входные и выходные линии контроля и управления порта в соответствии с рис. 6а. Если не провести эту коммутацию, то функции передачи и приема данных будут завершаться с признаком ошибки таймаута, а сам процесс передачи или приема не будет выполняться.

Листинг 2. Инициализация COM 1 и передача константы 05h с использование функций INT 14h.

```
2BF9:0100  BBFF2F  MOV    BX,2FFF   ;Счетчик циклов передач.
2BF9:0103  BA0000  MOV    DX,0000   ;Логический номер COM 1.
2BF9:0106  B400    MOV    AH,00     ;Функция инициализации. A5h - 2400 бод;
2BF9:0108  B0A5    MOV    AL,A5     ;без контроля; 2 стоп бита; длина 6 бит.
2BF9:010A  CD14    INT    14        ;Вызов драйвера.
2BF9:010C  B401    MOV    AH,01     ;Функция передачи символа.
2BF9:010E  B005    MOV    AL,05     ;Передаваемый символ.
2BF9:0110  CD14    INT    14        ;Вызов драйвера.
2BF9:0112  B90200  MOV    CX,0002   ;Вложенные циклы программной
2BF9:0115  51      PUSH   CX        ;задержки, позволяющие
2BF9:0116  B9FF00  MOV    CX,00FF   ;устанавливать временные
2BF9:0119  E2FE    LOOP   0119      ;промежутки между
2BF9:011B  59      POP    CX        ;передаваемыми
2BF9:011C  E2F7    LOOP   0115      ;байтами.
2BF9:011E  4B      DEC    BX        ;Декремент счетчика циклов передач.
2BF9:011F  75EB    JNZ    010C      ;Если не ноль, то очередная передача.
```

```
2BF9:0121  CC      INT3            ;Передача управления отладчику.
```

При правильной коммутации управляющих и контролирующих линий интерфейса на выводе 03 разъема COM-порта наблюдается сигнал, почти аналогичный сигналу, показанному на рис. 11. Отличие наблюдаемого сигнала заключается в отсутствии интервала паузы. Это позволяет сделать вывод, что функция передачи символа помещает очередной передаваемый символ в контроллер порта по признаку освобождения буферного регистра передатчика а не по признаку освобождения регистра сдвига передатчика, как реализовано в программе 1.

Программа 3(Comp2.bin), представленная в листинге 3, реализует одновременную передачу и прием передаваемого символа. Для реализации одновременной передачи и приема (т.е. дуплексного режима работы) необходимо соединить между собой линии передачи (контакт 03) и приема (контакт 02). Если часть программы, реализующей передачу символа, разместить на одном ПК, а программу приема - на другом, то соединение портов компьютеров необходимо осуществлять по схеме, представленной на рис.6б описания. Необходимо также учитывать, что контроллеры портов компьютеров должны быть инициализированы с одинаковыми параметрами.

Листинг 3. Инициализация COM 1, передача константы 05h и ее прием.

```
252D:0100  BAFB03  MOV    DX,03FB   ;В регистре LCR устанавливается разряд D7
252D:0103  B080    MOV    AL,80     ;(DLAB), разрешающий обмен с регистрами
252D:0105  EE      OUT    DX,AL     ;делителя(DLL и DLM).
252D:0106  B030    MOV    AL,30     ;Константу 30h заносим в DLL,
252D:0108  BAF803  MOV    DX,03F8   ;и константу 00h - в MLB,
252D:010B  EE      OUT    DX,AL     ;задавая скорость
252D:010C  B000    MOV    AL,00     ;передачи
252D:010E  BAF903  MOV    DX,03F9   ;величиной 2400 бод (см.
252D:0111  EE      OUT    DX,AL     ;таблицу 4 пункта 3.2 описания).
252D:0112  BAFB03  MOV    DX,03FB   ;05h заносится в LCR, устанавливая:
252D:0115  B005    MOV    AL,05     ;размер посылки - 6 бит, 2 стоп-бита и
252D:0117  EE      OUT    DX,AL     ;запрет контроля четности.
252D:0118  B000    MOV    AL,00     ;Занесением 00h в IER запрещаем
252D:011A  BAF903  MOV    DX,03F9   ;прерывания по всем событиям, могущим
252D:011D  EE      OUT    DX,AL     ;вызвать прерывание.
252D:011E  BAFC03  MOV    DX,03FC   ;Занесением 00h в IER запрещаем сигнал
252D:0121  EE      OUT    DX,AL     ;прерывания и режим диагностики.
252D:0122  BAFD03  MOV    DX,03FD   ;Чтение LSR и проверка: пусты ли регистры
252D:0125  EC      IN     AL,DX     ;передатчика,
252D:0126  A840    TEST   AL,40     ;если передача не закончена (D6=0), то
252D:0128  74F8    JZ     0122      ;на новый цикл проверки.
252D:012A  BAF803  MOV    DX,03F8   ;В регистр передатчика заносится
252D:012D  B005    MOV    AL,05     ;константа 05h, передаваемая
252D:012F  EE      OUT    DX,AL     ;затем в линию.
252D:0130  BAFD03  MOV    DX,03FD   ;Чтение регистра LSR и проверка:
252D:0133  EC      IN     AL,DX     ;принят ли символ во входной буфер
252D:0134  A801    TEST   AL,01     ;приемника (D0=1),
252D:0136  74FB    JZ     0133      ;если D0=0, то новый цикл ожидания.
252D:0138  BAF803  MOV    DX,03F8   ;Чтение из буферного регистра приемника
252D:013B  EC      IN     AL,DX     ;в регистр AL поступившего символа.
252D:013C  CC      INT3             ;Передача управления отладчику.
```

Программа 4(Comit2.bin), представленная в листинге 4, аналогична программе 3, но для передачи и приема символов использует функции драйвера INT 14h.

Листинг 4. Инициализация COM 1, передача константы 05h и ее прием с использованием функций инициализации, передачи и приема символа.

```
2BF9:0200  BA0000  MOV    DX,0000   ;Логический номер COM 1.
2BF9:0203  B400    MOV    AH,00     ;00 - Функция инициализации.
                                     ;Константа конфигурации A5h - 2400 бод;
2BF9:0205  B0A5    MOV    AL,A5     ;без контроля; 2 стоп бита; длина 6 бит.
2BF9:0207  CD14    INT    14        ;Вызов драйвера.
2BF9:0209  B401    MOV    AH,01     ;Функция передачи символа.
2BF9:020B  B005    MOV    AL,05     ;Передаваемый символ.
2BF9:020D  CD14    INT    14        ;Вызов драйвера.
2BF9:020F  B402    MOV    AH,02     ;Функция приема символа.
2BF9:0211  CD14    INT    14        ;Вызов драйвера.
2BF9:0213  CC      INT3             ;Передача управления отладчику.
```

Эта программа будет корректно работать только в рамках одного компьютера (на одном COM-порту или между COM-портами). Под корректностью работы в данном случае понимается успешное срабатывание функции приема символа за время таймаута (по умолчанию 1 секунда). Если же передавать символы с одного компьютера на другой, то для корректности работы программ передачи и приема программа приемника должна содержать процедуру ожидания поступления символа от другого компьютера, который может работать асинхронно с компьютером, принимающим символ..

Программа 5 (Comit3.bin), представленная в листинге 5, в разделе приема символа использует функцию 03h для определения момента поступления символа во входной буфер приемника.

Листинг 5. Инициализация COM-порта, передача символа, ожидание его поступления и его прим.

```
2BF9:0200  BA0000   MOV    DX,0000   ;Логический номер COM 1.
2BF9:0203  B400     MOV    AH,00     ;00 - Функция инициализации.
                                      ;Константа конфигурации A5h - 2400 бод;
2BF9:0205  B0A5     MOV    AL,A5     ;без контроля; 2 стоп бита; длина 6 бит.
2BF9:0207  CD14     INT    14        ;Вызов драйвера.
2BF9:0209  B401     MOV    AH,01     ;Функция передачи символа.
2BF9:020B  B005     MOV    AL,05     ;Передаваемый символ.
2BF9:020D  CD14     INT    14        ;Вызов драйвера.
2BF9:020F  B403     MOV    AH,03     ;Функция определения состояния интерфейса.
2BF9:0211  CD14     INT    14        ;Вызов драйвера.
2BF9:0213  F6C401   TEST   AH,01     ;Проверка: принят ли символ.
2BF9:0216  74F7     JZ     020F      ;Если нет, то новый цикл проверки.
2BF9:0218  B402     MOV    AH,02     ;Функция приема символа.
2BF9:021A  CD14     INT    14        ;Вызов драйвера.
2BF9:021C  CC       INT3             ;Передача управления отладчику.
```

Программы 6 и 7 (Comt1.bin и Comr1.bin).

Эти программы, представленные в листингах 6 и 7, реализуют протокол передачи данных с одного компьютера на другой через их COM-порты. Передача данных производится по асинхронному протоколу, использующему сигналы управления потоком и квитирования. Временная диаграмма протокола передачи-приема представлена на рис. 12. В протоколе учитывается исходное состояние выходных управляющих сигналов COM-портов (сигналы DTR и RTS), которые на стадии начальных тестов (POST) устанавливаются BIOS в состояние активного низкого уровня (-12 В). Протоколом предусмотрено восстановление исходного состояния этих сигналов после окончания передачи и приема данных. В программах отсутствуют процедуры инициализации контроллеров портов, поэтому используется режим, установленный DIOS на стадии POST (скорость - 2400 бод, поле данных -8 бит, 1 стоп-бит, отсутствие контроля четности). COM-порты компьютеров должны быть соединены меду собой кабелем, разводка которого представлена на рис. 6б описания лабораторной работы.

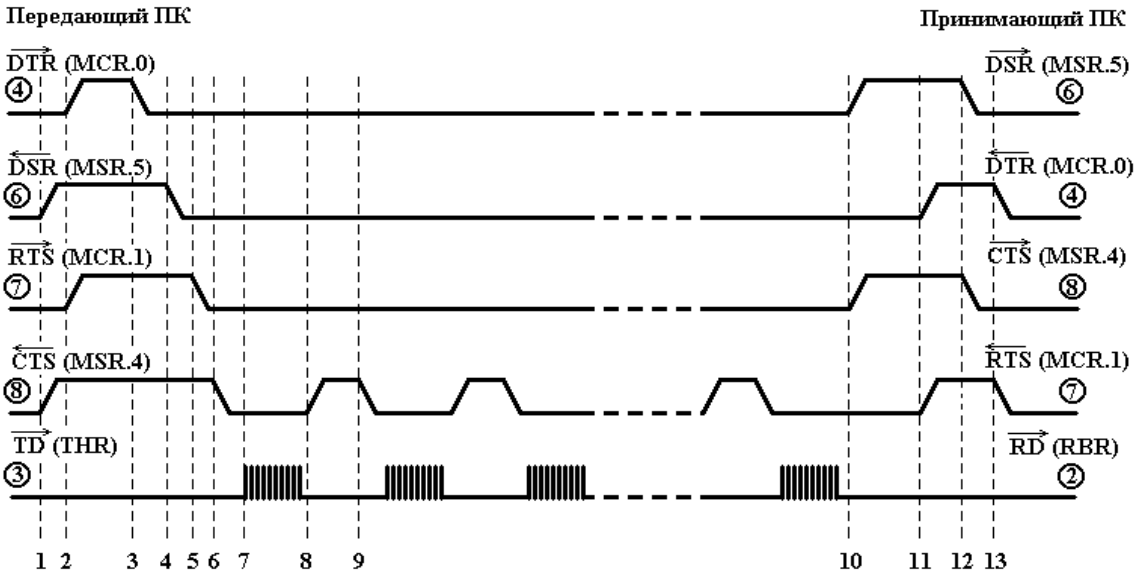

Рис. 12. Временная диаграмма протокола передачи-приема данных через COM-порты ПК.

На временной диаграмме показаны фазы изменения состояния линий интерфейса в процессе реализации алгоритма протокола в начальной стадии, в стадии передачи-приема данных и в стадии завершения работы. Описание фаз представлено ниже.

1. **Принимающий ПК** устанавливает высокий уровень управляющих сигналов DTR и RTS через разряды MCR.0 и MCR.1 регистра управления модемом (MCR) контроллера COM-порта и передает их по кабелю

интерфейса на входы DSR и CTS COM-порта принимающего ПК. После этого он переходит к ожиданию аналогичных действий со стороны предающего ПК.

2. **Передающий ПК** анализирует состояние входных сигналов DSR и CTS COM-порта через разряды MSR.5 и MSR.4 регистра состояния модема (MSR) контроллера COM-порта. Если сигналы имеют низкий уровень, то анализ повторяется, если оба сигнала переходят в состояние высокого уровня, то ПК переключает выходные сигналы DTR и RTS через регистр MCR своего COM-порта в состояние высокого уровня. Далее формируется длительность положительного состояния сигнала DTR (несколько мкс). **Принимающий ПК** получив от передающего ПК высокие уровни сигналов на входы DSR и CTS переходит к циклам ожидания поступления на вход DSR сигнала низкого уровня (готовность передающего ПК к обмену).
3. **Передающий ПК** формирует низкий уровень выходного сигнала DTR (готов к обмену) и переходит к циклам ожидания получения низкого уровня входного сигнала DSR (готов к обмену) от принимающего ПК.
4. **Принимающий ПК**, получив сигнал низкого уровня на вход DSR, формирует на своем выходе DTR сигнал низкого уровня (готов к обмену), поступающий на вход DSR передающего ПК и переходит к циклам ожидания поступления сигнала низкого уровня на вход CTS. В этих циклах ожидания декрементируется счетчик тайм-аута, содержимое которого проверяется при выходе из циклов ожидания. Если в счетчике нулевое значение, то осуществляется переход на процедуры завершения работы программы (фаза 11), если не нуль - на процедуры фазы 6.
5. **Передающий ПК**, получив сигнал низкого уровня на входе DSR, формирует сигнал низкого уровня на выходе RTS своего COM-порта (готов передать байт данных) и переходит к циклам ожидания поступления на вход CTS сигнала низкого уровня (разрешение передачи). В этих циклах ожидания декрементируется счетчик тайм-аута, содержимое которого проверяется при выходе из циклов ожидания. Если в счетчике нулевое значение, то осуществляется переход на процедуры завершения работы программы (фаза 10), если не нуль - на процедуры фазы 7.
6. **Принимающий ПК**, получив низкий уровень на входе CTS, формирует низкий уровень сигнала на выходе RTS.и переходит к циклам ожидания поступления во входной буфер RBR контроллера своего COM-порота символа от передающего ПК.
7. **Передающий ПК**, получив сигнал низкого уровня на входе CTS, передает через регистр THR контроллера COM-порта в линию TD байт данных в последовательном формате и переходит к циклам ожидания поступления сигнала высокого уровня на вход CTS.
8. **Принимающий ПК**, получив байт данных (символ) формирует на выводе RTS COM-порта сигнал высокого уровня определенной длительности. **Передающий ПК**, получив на входе CTS сигнал высокого уровня, декрементирует счетчик передаваемых байт и, если он не равен нулю, переходит на начало нового цикла передачи байта данных (фаза 3). Если счетчик обнулялся, то организуется переход на фазу 10.
9. **Принимающий ПК**, закончив формирование длительности положительного уровня на выходе RTS, переводит его в состояние низкого уровня и проверяет готовность передающего ПК (уровень сигнала на входе DSR). Если DSR имеет низкий уровень, ПК переходит на новый цикл приема символов (фаза 4), в противном случае осуществляется переход на процедуры завершения работы программы (фаза 11).
10. **Передающий ПК** формирует на своих выводах DTR и RTS сигналы высокого уровня, длительность которых в два раза превышает длительность положительных импульсов, формируемых принимающим ПК на выводе RTS своего COM-порта в циклах приема символов.
11. **Принимающий ПК** устанавливает высокий уровень на выходах DTR и RTS своего COM-порта на время длительности положительных импульсов, формируемых принимающим ПК на выводе RTS своего COM-порта в циклах приема символов.
12. **Передающий ПК** после программной задержки устанавливает сигналы на выходах DTR и RTS в состояние низкого уровня и передает управление отладчику.
13. **Принимающий ПК** после программной задержки устанавливает сигналы на выходах DTR и RTS в состояние низкого уровня и передает управление отладчику.

Ниже приведены две программы, реализующие алгоритмы протокола. Программа Comt1.bin, представленная в листинге 6 реализует алгоритмы протокола, относящиеся к передающему ПК, а программа Comr1.bin, представленная в листинге 7 реализует алгоритмы принимающего ПК.

Листинг 6. Программа, реализующая алгоритм работы передающего ПК **(Comt1.bin).**

```
            ; Ф А З А  2
252D:0100  BAFE03  MOV    DX,03FE   ;Проверка уровней входных сигналов
252D:0103  EC      IN     AL,DX     ;DSR и CTS на разъеме COM-порта
252D:0104  A830    TEST   AL,30     ;передающего компьютера:
252D:0106  7502    JNZ    010A      ;если высокий, - на продолжение программы,
252D:0108  EBF9    JMP    0103      ;иначе - новый цикл проверки.
252D:010A  BAFC03  MOV    DX,03FC   ;Установка через регистр MCR высокого
252D:010D  EC      IN     AL,DX     ;уровня на выводах DTR и RTS
```

```
252D:010E  0C03    OR      AL,03     ;передающего компьютера (неготовность
252D:0110  EE      OUT     DX,AL     ;к передаче данных).
252D:0111  BAFE03  MOV     DX,03FE   ;Перепроверка уровней входных сигналов
252D:0114  EC      IN      AL,DX     ;DSR и CTS на разъеме COM-порта:
252D:0115  A830    TEST    AL,30     ;если низкий уровень, то
252D:0117  74F8    JZ      0111      ;на новый цикл проверки.
           ;Ф А З А  3
252D:0119  BBFF00  MOV     BX,00FF   ;Число циклов передачи символов.
252D:011C  BAFC03  MOV     DX,03FC   ;Установка низкого уровня
252D:011F  EC      IN      AL,DX     ;сигнала DTR COM-порта
252D:0120  24FE    AND     AL,FE     ;передающего
252D:0122  EE      OUT     DX,AL     ;компьютера.
252D:0123  BAFE03  MOV     DX,03FE   ;Ожидание низкого уровня
252D:0126  EC      IN      AL,DX     ;на входе DSR COM-порта передающего
252D:0127  A820    TEST    AL,20     ;компьютера, если высокий уровень ,
252D:0129  75FB    JNZ     0126      ;то на новый цикл ожидания.
           ;Ф А З А  5
252D:012B  BAFC03  MOV     DX,03FC   ;Установка низкого уровня
252D:012E  EC      IN      AL,DX     ;сигнала RTS COM-порта
252D:012F  24FD    AND     AL,FD     ;передающего
252D:0131  EE      OUT     DX,AL     ;компьютера.
252D:0132  B9FFFF  MOV     CX,FFFF   ;Счетчик тайм-аута.
252D:0135  BAFE03  MOV     DX,03FE   ;Ожидание низких уровней входных
252D:0138  EC      IN      AL,DX     ;сигналов DSR и CTS
252D:0139  A830    TEST    AL,30     ;с декрементом счетчика тайм-аута,
252D:013B  E0FB    LOOPNZ  0138      ;если не нуль, - на новый цикл ожидания.
252D:013D  83F900  CMP     CX,0000   ;Если выход по тайм-ауту (CX=0),
252D:0140  7411    JZ      0153      ;та на окончание программы.
           ;Ф А З А  7
252D:0142  BAF803  MOV     DX,03F8   ;Передача на
252D:0145  B055    MOV     AL,5A     ;выход TD
252D:0147  EE      OUT     DX,AL     ;символа 5Ah.
252D:0148  BAFE03  MOV     DX,03FE   ;Проверка состояния входного
252D:014B  EC      IN      AL,DX     ;сигнала CTS:
252D:014C  A810    TEST    AL,10     ;если низкий уровень, то
252D:014E  74FB    JZ      014B      ;на новый цикл проверки.
           ;Ф А З А  8
252D:0150  4B      DEC     BX        ;Декремент счетчика передаваемых символов.
252D:0151  75C9    JNZ     011C      ;Если BX не нуль, - новый цикл передачи.
           ;Ф А З А  10
252D:0153  BAFC03  MOV     DX,03FC   ;Установка
252D:0156  EC      IN      AL,DX     ;высокого
252D:0157  0C03    OR      AL,03     ;уровня
252D:0159  EE      OUT     DX,AL     ;сигналов DTR и RTS.
252D:015A  B9FF00  MOV     CX,04FF   ;Формирование длительности
252D:015D  BAFF03  MOV     DX,03FF   ;удержания
252D:0160  EC      IN      AL,DX     ;высокого уровня
252D:0161  E2FD    LOOP    0160      ;сигналов DTR и RTS.
           ;Ф А З А  12
252D:0163  BAFC03  MOV     DX,03FC   ;Установка
252D:0166  EC      IN      AL,DX     ;низкого
252D:0167  24FC    AND     AL,FC     ;уровня
252D:0169  EE      OUT     DX,AL     ;сигналов DTR и RTS.
252D:016A  CC      INT3              ;Передача управления отладчику.
```

Листинг 6. Программа, реализующая алгоритм работы принимающего ПК **(Comr1.bin).**

```
           ;Ф А З А  1
2BF9:0100  BAFC03  MOV     DX,03FC   ;Установка через регистр MCR высокого
2BF9:0103  EC      IN      AL,DX     ;уровня на выводах DTR и RTS
2BF9:0104  0C03    OR      AL,03     ;принимающего компьютера (неготовность
2BF9:0106  EE      OUT     DX,AL     ;к приему данных).
2BF9:0107  BAFE03  MOV     DX,03FE   ;Проверка через регистр MSR неготовности
2BF9:010A  EC      IN      AL,DX     ;передающего компьютера (сигналы DSR и CTS
2BF9:010B  A830    TEST    AL,30     ;высокого уровня); Если не готов,
```

```
2BF9:010D  7502    JNZ    0111      ;то переход на продолжение программы,
2BF9:010F  EBF9    JMP    010A      ;иначе - на новый цикл проверки.
           ;Ф А З А  2
2BF9:0111  BAFE03  MOV    DX,03FE   ;Ожидание низкого уровня на контакте
2BF9:0114  EC      IN     AL,DX     ;DSR COM-порта:
2BF9:0115  A820    TEST   AL,20     ;если DSR высокого уровня, то
2BF9:0117  75F8    JNZ    0111      ;на новый цикл ожидания.
           ;Ф А З А  4
2BF9:0119  BAFC03  MOV    DX,03FC   ;Установка низкого уровня на контакте DTR
2BF9:011C  EC      IN     AL,DX     ;порта (принимающий
2BF9:011D  24FE    AND    AL,FE     ;компьютер готов
2BF9:011F  EE      OUT    DX,AL     ;работе).
2BF9:0120  B9FFFF  MOV    CX,FFFF   ;Счетчик тайм-аута ожидания.
2BF9:0123  BAFE03  MOV    DX,03FE   ;Проверка прекращения работы
2BF9:0126  EC      IN     AL,DX     ;передающего компьютера
2BF9:0127  A830    TEST   AL,30     ;по превышению
2BF9:0129  75FB    LOOPNZ 0126      ;тайм-аута.
2BF9:012B  83F900  CMP    CX,0000   ;Если CX=0 (тайм-аут), то переход на
2BF9:012E  7436    JZ     0166      ;процедуру остановки приема.
           ;Ф А З А  6
2BF9:0130  BAFC03  MOV    DX,03FC   ;Установка низкого уровня RTS - приемник
2BF9:0133  EC      IN     AL,DX     ;запрашивает у передающего
2BF9:0134  24FD    AND    AL,FD     ;компьютера
2BF9:0136  EE      OUT    DX,AL     ;передачу символа.
2BF9:0137  BAFD03  MOV    DX,03FD   ;Ожидание поступления
2BF9:013A  EC      IN     AL,DX     ;символа из линии во
2BF9:013B  A801    TEST   AL,01     ;входной буфер
2BF9:013D  74FB    JZ     013A      ;приемника COM-порта.
           ;Ф А З А  8
2BF9:013F  BAF803  MOV    DX,03F8   ;Чтение поступившего
2BF9:0142  EC      IN     AL,DX     ;символа и помещение его
2BF9:0143  88C4    MOV    AH,AL     ;в регистр AH.
2BF9:0145  BAFC03  MOV    DX,03FC   ;Установка высокого уровня
2BF9:0148  EC      IN     AL,DX     ;на выводе RTS COM-порта
2BF9:0149  0C02    OR     AL,02     ;принимающего
2BF9:014B  EE      OUT    DX,AL     ;компьютера.
2BF9:014C  B9FF04  MOV    CX,00FF   ;Формирование длительности
2BF9:014F  BAFF03  MOV    DX,03FF   ;положительного уровня
2BF9:0152  EC      IN     AL,DX     ;сигнала RTS (приемник
2BF9:0153  E2FD    LOOP   0152      ;получил символ и занят его обработкой).
           ;Ф А З А  9
2BF9:0155  BAFC03  MOV    DX,03FC   ;Установка низкого уровня
2BF9:0158  EC      IN     AL,DX     ;на выводе RTS COM-порта
2BF9:0159  24FD    AND    AL,FD     ;принимающего
2BF9:015B  EE      OUT    DX,AL     ;компьютера.
2BF9:015C  BAFE03  MOV    DX,03FE   ;Проверка готовности передающего
2BF9:015F  EC      IN     AL,DX     ;компьютера к передаче
2BF9:0160  A810    TEST   AL,10     ;очередного символа,
2BF9:0162  7502    JNZ    0166      ;если нет - переход на остановку,
2BF9:0164  EBBA    JMP    0120      ;если да - на новый цикл приема.
           ;Ф А З А  11
2BF9:0166  BAFC03  MOV    DX,03FC   ;Перевод сигналов на выводах DTR и RTS
2BF9:0169  EC      IN     AL,DX     ;в высокий уровень (приемник
2BF9:016A  0C03    OR     AL,03     ;не готов
2BF9:016C  EE      OUT    DX,AL     ;к работе).
2BF9:016D  B9FF08  MOV    CX,08FF   ;Формирование
2BF9:0170  BAFF03  MOV    DX,03FF   ;длительности
2BF9:0173  EC      IN     AL,DX     ;состояния
2BF9:0174  E2FD    LOOP   0173      ;неготовности.
           ;Ф А З А  13
2BF9:0176  BAFC03  MOV    DX,03FC   ;Перевод сигналов
2BF9:0179  EC      IN     AL,DX     ;на выводах DTR и RTS
2BF9:017A  24FC    AND    AL,FC     ;в низкий уровень (состояние сигналов
2BF9:017C  EE      OUT    DX,AL     ;по умолчанию после POST).
```

```
2BF9:017D  CC      INT3            ;Передача управления отладчику.
```

7. Рекомендуемая литература.

При подготовке к лабораторной работе и к ее защите рекомендуется пользоваться информацией, представленной в описании лабораторной работы, в разделе 2.1.2 учебно-методического пособия и в дополнительной литературе [3 - 7, 9, 19].

8. Контрольные вопросы

1. Дайте краткую характеристику COM-портам ПК.
2. Дайте краткую характеристику интерфейсу RS-232C.
3. Какие типы разъемов RS-232C используются в COM-портах ПК?
4. Что понимается под нуль-модемным кабелем?
5. Охарактеризуйте сигналы COM-порта ПК.
6. Охарактеризуйте последовательность управляющих сигналов протокола RS-232C.
7. Для чего в COM-портах используются преобразователи уровней?
8. Охарактеризуйте правила, по которым производится инверсия сигналов в COM-портах ПК.
9. Охарактеризуйте родственные RS-232C интерфейсы RS-422A и RS-423A.
10. Охарактеризуйте интерфейс RS-485.
11. Охарактеризуйте интерфейс "Токовая петля".
12. Охарактеризуйте формат посылки байта асинхронного режима RS-232C.
13. Каковы требования к рассогласованию скоростей передачи и приема в асинхронном режиме RS-232C?
14. Какие ошибки передачи позволяет выявить формат асинхронной посылки?
15. Что понимается под выражением: "количество стоп-битов равно полтора бита"?
16. В чем разница между квитированием и управлением потоком?
17. Охарактеризуйте аппаратный протокол управления потоком RTS/CTS.
18. Охарактеризуйте программный протокол управления потоком XON/XOFF.
19. Какие микросхемы асинхронных приемопередатчиков использовались в COM-портах в PC/XT и PC/AT?
20. Какие операции должен выполнять УАПП при передаче символа?
21. Какие операции должен выполнять УАПП при приеме символа?
22. Охарактеризуйте УАПП семейства 8250 / 16540.
23. Охарактеризуйте функциональные возможности УАПП 8250 / 16450.
24. Охарактеризуйте блок-схему УАПП 8250 / 16450 (рис. 2.17).
25. Охарактеризуйте особенности УАПП 16550.
26. Какие функции выполняет УАПП 16550?
27. Охарактеризуйте блок-схему УАПП 16550 (рис. 2.18).
28. Перечислите программно-доступные регистры УАПП 8250 / 16450.
29. Чем отличается регистровая архитектура УАПП 16550 от архитектуры 8250 / 16450?
30. Охарактеризуйте регистры THR и RBR.
31. Охарактеризуйте регистры DLL и DLM.
32. Охарактеризуйте регистр IER.
33. Охарактеризуйте регистр IIR.
34. Охарактеризуйте регистр FCR.
35. Охарактеризуйте регистр LCR.
36. Охарактеризуйте регистр MCR.
37. Охарактеризуйте регистр LSR.
38. Охарактеризуйте регистр MSR.
39. Охарактеризуйте диагностический режим УАПП.
40. В чем заключается системная поддержке COM-портов на уровне BIOS?
41. Что понимается под инициализацией COM-портов на стадии POST?
42. Какие функции поддерживает драйвер COM-порта INT 14h?
43. Каково назначение ячеек памяти рабочей области данных BIOS, выделяемых под COM-порты?
44. Дайте краткую характеристику драйверу COM-порта INT 14h.
45. Охарактеризуйте функцию 00h прерывания INT 14h.
46. Охарактеризуйте функции 01h и 02h прерывания INT 14h.
47. Охарактеризуйте функцию 03h прерывания INT 14h.
48. Что понимается под конфигурированием COM-портов?
49. Охарактеризуйте области использования COM-портов.
50. Как поддерживается спецификация PnP устройствами, подключаемыми к COM-портам?
51. Как маркируется строка идентификатора PnP, принимаемая через COM-порт?
52. Как проверяется отключение устройства PnP, подключенного к COM-порту?
53. Охарактеризуйте ошибки конфигурирования COM-портов.
54. Охарактеризуйте особенности функционального тестирования COM-портов.
55. Охарактеризуйте особенности питания маломощных устройств от выходных линий COM-порта.

8. Лабораторная работа 7 (описание) "Манипулятор Serial Mouse".

1. Цель работы.

Целью лабораторной работы является ознакомление студентов с принципами работы манипулятора Serial Mouse; изучение функционирования электрической схемы манипулятора с регистрвцией сигналов в контрольных точках схемы с помощью электронного осциллографа; изучение сигнального протокола интерфейса манипулятора а также используемых Serial Mouse форматов передачи данных.

2. Технические и программные средства , используемые в работе.

В работе используется персональный компьютер с манипулятором Serial Mouse, подключенным к порту COM 1 и имеющим зондовое устройство доступа к контрольным точкам его электрической схемы (ЗУДКТ). Могут использоваться и другие манипуляторы Serial Mouse с соответствующими ЗУДКТ и их описаниями. ПК работает под управлением ОС Windows 95/98/Mellenium. Операционная система загружается в режиме DOS (Command prompt only) с оболочкой типа Norton Commander. Непосредственное низкоуровневое программное взаимодействие с манипулятором осуществляется в среде отладчика типа AFD.

В работе используется также электронный осциллограф с полосой регистрации 0 - 10 МГц и выше.

3. Краткое описание манипулятора Serial Mouse и его интерфейса.

Serial Mouse — мышь с последовательным интерфейсом, подключаемая через 9-или 25-контактный разъем COM-порта (табл. 1).

Таблица 1. Разъемы Serial Mouse

Сигнал мыши	Сигнал COM-порта	Контакт DB9	Контакт DB25
Data	RD	2	3
GND	SG	5	7
+V(питание)	RTS, (DTR)	7 (4)	4 (20)
-V(питание)	TD	3	2

Эта мышь имеет встроенный микроконтроллер, который обрабатывает сигналы от координатных датчиков и кнопок. Каждое событие — перемещение мыши или нажатие-отпускание кнопки — кодируется двоичной посылкой по интерфейсу RS-232C. Для передачи информации применяется асинхронная передача, а двуполярное питание, требуемое по протоколу RS-232, обеспечивается от управляющих линий интерфейса. Недостатком мыши Serial Mouse является то, что она занимает COM-порт и требует монопольного использования его штатной линии прерывания (IRQ4 для COM1 и IRQ3 для COM2). Две основные разновидности *Serial Mouse — MS Mouse* (Microsoft Mouse) и *PC Mouse* (Mouse Systems) — требуют разных драйверов, многие мыши имеют переключатель MS/PC. Эти два типа мышей при одинаковой скорости 1200 бит/с, одном стоп-бите и отсутствии контроля четности используют разные форматы посылок, приведенные в разделе 3.3.

- *MS Mouse:* 7 битов данных, трехбайтный пакет (в «классическом» варианте), положительным значениям соответствует перемещение по координате *X* вправо, а по координате *Y* вниз. Для трехкнопочных мышей добавляется четвертый байт, передаваемый только при изменении состояния средней кнопки. Для 3D-мыши четвертый байт имеет иное назначение.
- *PC Mouse:* 8 битов данных, пятибайтный пакет, положительным значениям соответствует перемещение по координате *X* вправо, а по координате У вверх.
- Несовпадение форматов объясняет беспорядочные перемещения курсора на экране при несоответствии драйвера типу мыши.

Системная поддержка последовательной мыши осуществляется только на уровне ОС (в DOS сервисы вызываются через Int 33h) - драйвер мыши, загружаемый или встроенный в ОС. BIOS Serial Mouse не поддерживает, даже если и пользуется ею для навигации в CMOS Setup.

Конструкция Serial Mouse аналогична конструкции обычной PS/2 мыши и имеет свободно вращающийся массивный обрезиненный шарик в днище корпуса, передающий вращение на два координатных диска с фотоэлектрическими датчиками. Датчики для каждой координаты представляют собой две открытые оптопары (светодиод-фотодиод), в оптический канал которых входит вращающийся диск с прорезями. Оптопары датчиков могут оформляться в виде монолитных конструкций или быть просто

отдельными элементами, установленными на печатной плате. В мыши имеется *микроконтроллер,* который обрабатывает сигналы с датчиков и посылает в ПК информацию о перемещениях и состоянии кнопок.

3.1. Схемотехника манипуляторов Serial Mouse.

Схемотехника Serial Mouse имеет много общего со схемотехникой обычной PS/2 мыши. На рис. 1 представлена общая электрическая схема универсальной Serial Mouse, которая может работать в режиме MS Mouse и PC Mouse. Переключение режимов работы осуществляется переключателем SW4, расположенным на нижней части корпуса манипулятора.

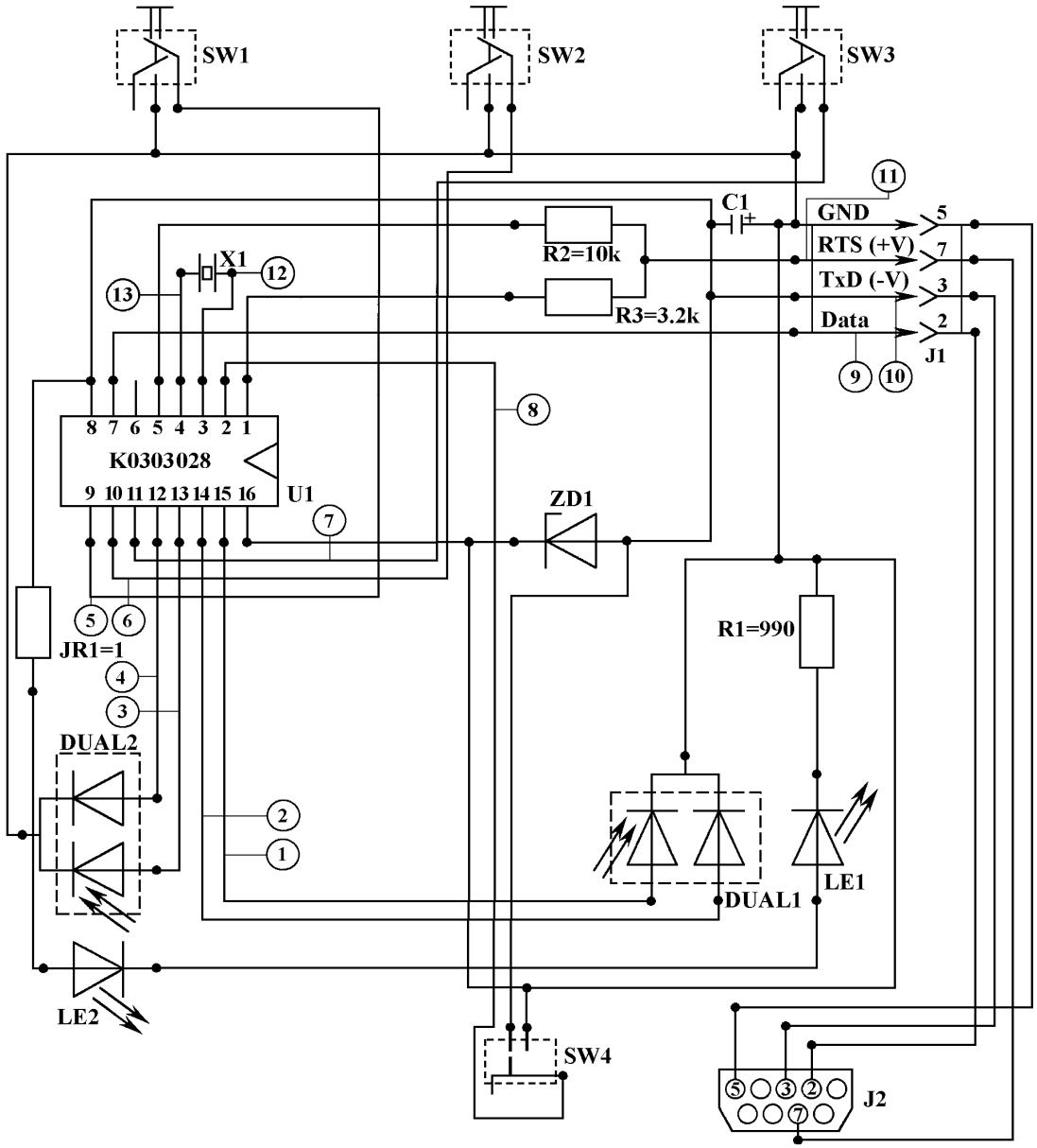

Рис.1. Общая электрическая схема универсальной COM мыши (Serial Mouse).

Компоненты электрической схемы.

- **SW1** - кнопочный датчик правой клавиши мыши.
- **SW2** - кнопочный датчик средней клавиши мыши.
- **SW3** - кнопочный датчик левой клавиши мыши.
- **X1** - кварцевый резонатор с частотой 32 768 Гц.
- **R2 и R3** - резисторы, через которые на контроллер мыши К0303028 подается напряжение питания положительной полярности.
- **C1** - конденсатор, фильтрующий высокочастотные помехи в цепи питания отрицательной полярности.
- **J1** - разъем подключения интерфейсного кабеля к электронному модулю мыши.
- **U1** - микроконтроллер мыши типа К0303028
- **JR1** - резистор - перемычка величиной 1 Ом, включенный в цепи питания светодиодов EL1 и EL1.
- **ZD1** - ограничительный стабилитрон напряжения цепи питания отрицательной полярности.

- **R1** - резистор величиной 990 Ом, задающий величину тока в цепи светодиодов LE1 и LE2.
- **LE1** - светодиод оптопары датчика перемещения по горизонтали.
- **DUAL1** - сдвоенный фотодиод оптопары датчика перемещения по горизонтали.
- **DUAL2** - сдвоенный фотодиод оптопары датчика перемещения по вертикали.
- **LE1** - светодиод оптопары датчика перемещения по вертикали.
- **SW4** - переключатель режимов работы мыши (MS Mouse/PC Mouse).
- **J2** - разъем интерфейсного кабеля, подключаемый к COM-порту компьютера.

Контрольные точки электрической схемы.

Контрольные точки на схеме указаны цифрами, обведенными окружностями. Они обозначают цепи электрической схемы COM мыши, в которых можно контролировать характерные электрические сигналы.

1, 2 - сигналы, поступающие с фотодиодов оптопары датчика перемещения по горизонтали; сдвиг по фазе этих сигналов составляет примерно 90°, что позволяет определить направление перемещения мыши по горизонтали. (

3, 4 - сигналы, поступающие с фотодиодов оптопары датчика перемещения по вертикали; сдвиг по фазе этих сигналов составляет примерно 90°, что позволяет определить направление перемещения мыши по вертикали.

5 - сигнал с кнопочного датчика **SW1** правой клавиши мыши.

6 - сигнал с кнопочного датчика **SW2** средней клавиши мыши.

7 - сигнал с кнопочного датчика **SW3** левой клавиши мыши.

8 - сигнал с переключателя **SW4** режима работы мыши.

9 - сигналы на линии данных интерфейса мыши.

10 - уровень напряжения питания мыши отрицательной полярности, оно поступает на мышь сразу же после инициализации контроллера последовательного порта процедурами начального теста POST BIOS.

11 - уровень напряжения питания мыши положительной полярности; процедуры начального теста POST BIOS устанавливают на этой линии отрицательный уровень напряжения (-12 В); положительный уровень напряжения устанавливает драйвер мыши после его загрузки установкой "единиц" в двух младших разрядах регистра управления модемом (MCR) контроллера COM порта.

12, 13 - сигналы на выводах кварцевого резонатора **X1**.

Примечание

В лабораторной работе используются манипуляторы с вмонтированными в них зондовыми устройствами доступа к контрольным точкам электрической схемы, в состав которых входит разъем доступа типа ГРПМ1-31Г02-В. Контрольные точки 1, 2, ……. 12, 13 выведены соответственно на контакты разъема с номерами: Б1, Б2, ……. Б12, Б13.

3.2. Интерфейс Serial Mouse.

Как уже упоминалось Serial Mouse подключается к персональному компьютеру типа PC/AT через последовательный асинхронный интерфейс - COM-порт. Универсальный последовательный внешний интерфейс — *COM-порт* (Communications Port — коммуникационный порт) присутствует в PC начиная с первых моделей. Этот порт обеспечивает *асинхронный* обмен по стандарту RS-232C. (*Синхронный* обмен в PC поддерживают лишь специальные адаптеры, например SDLC или V.35). COM-порты реализуются на микросхемах универсальных асинхронных приемопередатчиков *(UART),* совместимых с семейством i8250/16450/16550. Они занимают в пространстве ввода-вывода по 8 смежных 8-битных регистров и могут располагаться по стандартным *базовым адресам* 3F8h (COM1), 2F8h (COM2), 3E8h (COM3), 2E8h (COM4). Порты могут вырабатывать *аппаратные прерывания* IRQ4 (обычно используются для COM1 и COM3) и IRQ3 (для COM2 и COM4). С внешней стороны порты имеют линии последовательных данных передачи и приема, а также набор сигналов управления и состояния, соответствующий стандарту RS-232C. COM-порты имеют внешние разъемы-вилки (male — «папа») *DB25P* или *DB9P,* выведенные на заднюю панель компьютера (см. п. 2.1 учебного пособия). Характерной особенностью интерфейса является применение «не ТТЛ» сигналов — все внешние сигналы порта двуполярные. Гальваническая развязка отсутствует — схемная земля подключаемого устройства соединяется со схемной землей компьютера. Скорость передачи данных может достигать 115 200 бит/с и более.

Компьютер может иметь до четырех последовательных портов COM1-COM4 (для машин класса AT типично наличие двух портов) с поддержкой на уровне BIOS. Сервис BIOS Int 14h обеспечивает инициализацию порта, ввод и вывод символа (не используя прерываний) и опрос состояния. Через Int 14h скорость передачи программируется в диапазоне 110-9600 бит/с (меньше, чем реальные возможности порта). Для повышения производительности широко используется взаимодействие программ с портом на уровне регистров, для чего требуется совместимость аппаратных средств COM-порта с программной моделью i8250/16450/16550.

Название порта указывает на его основное назначение — подключение коммуникационного оборудования (например, модема) для связи с другими компьютерами, сетями и периферийными устройствами. К порту могут непосредственно подключаться и периферийные устройств с последовательным интерфейсом: принтеры, плоттеры, терминалы и другие. COM-порт широко используется

для подключения мыши, а также организации непосредственной связи двух компьютеров. К COM-порту подключают и электронные ключи.

Практически все системные платы (еще начиная с PCI-плат для процессоров 486) имеют встроенные адаптеры двух COM-портов. Один из портов может использоваться и для беспроводной инфракрасной связи с периферийными устройствами (IrDA). Существуют карты ISA с парой COM-портов, где они чаще всего соседствуют с LPT-портом, а также с контроллерами дисковых интерфейсов (FDC+IDE). Если возникает потребность в большом количестве последовательных интерфейсов, то в ПК можно установить специальные адаптеры-мультиплексоры. Это весьма дорогие карты, они выпускаются обычно на 4, 8, 12 и даже 16 портов. Такое большое число разъемов на заднюю стенку ПК вывести проблематично, и у мультиплексоров обычно имеется внешний блок с разъемами (и электроникой), соединяемый с адаптером кабелем с многоконтактными разъемами. BIOS мультиплексоры не поддерживает.

«Классический» COM-порт позволяет осуществлять обмен данными только программно-управляемым способом, при этом для пересылки каждого байта процессору приходится выполнять несколько инструкций. Современные порты, выполненные на базе контроллеров типа 16550A, имеют FIFO-буферы данных и позволяют выполнять обмен по каналу DMA, однако в BIOS Setup большинства современных PC совместимых компьютеров не предусмотрено выделение COM-портам каналов DMA, как, например, это предусмотрено для режима ECP LPT-порта.

Достаточно полное описание интерфейса RS 232C представлено в разделе 2 1 2 учебного пособия "Подсистемы клавиатуры и манипуляторов типа мышь и джойстик".

3.3.Форматы передачи данных манипуляторами типа Serial Mouse.

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 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 таблице 2. 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 2. 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 введен для двухкнопочной мыши. Средняя кнопка трехкнопочной 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 произошел сбой в процессе передачи и следует проигнорировать принятый пакет.

Скорость приема-передачи данных принята равной 1200 бод, длина передаваемого слова - 7 бит, контроль по четности 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 таблице 3.

Таблица 3. 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[1]	0	M	0	0	0	0	0

[1] Передача четвертого байта производится только в случае изменения состояния средней кнопки мыши.

Конкуренция между изготовителями компьютерного оборудования привела к тому, что на рынке стали появляться все больше «навороченных» устройства, снабженные рядом дополнительных функций. Дополнительные возможности требовали, естественно, передачи дополнительной информации от устройства к компьютеру. Например, формат данных трех координатной (3D) мыши, показанный в таблице 4, включает координату Z (Z0-Z3, целое число со знаком) и состояние специальной кнопки R_0.

Существует группа универсальных устройств, которые поддерживают два типа протоколов - PC Mouse и MS Mouse. Переключение между протоколами осуществляется аппаратно (переключателем на корпусе мыши) или программно (путем подачи специального сигнала со стороны драйвера мыши).

Таблица 4. Формат 3D Serial 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
4	0	0	R0	Z3	Z2	Z1	Z

Особенностью устройств данного типа является 8-битная длина слова данных во всех режимах работы (при этом в режиме MS Mouse шестой разряд в каждом слове данных просто дублируется в седьмой, как показано в таблице 5).Соответственно при использовании подобных устройств для последовательного порта необходимо установить следующие параметры работы: скорость приема-передачи данных 1200 бод, длина передаваемого слова - 8 бит, контроль по четности не используется, число стоповых бит равно 1. Однако даже если задана длина передаваемого слова 7 бит, в режиме MS Mouse седьмой разряд игнорируется, и нарушения процесса передачи не происходит. Таким образом, устройства данного типа могут успешно взаимодействовать с любым драйвером, поддерживающим стандартный протокол Microsoft.

Таблица 5. Нестандартный 8-битный формат типа MS Mouse для мыши с двумя альтернативными протоколами передачи данных

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

Предложенный фирмой IBM формат Mouse System (PC Mouse), который показан в таблице 6, поддерживается некоторыми универсальными устройствами.

Таблица 6. **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"

Обозначения в таблице расшифровываются следующим образом:

- X0-X7 - перемещение по оси X;
- Y0-Y7 - перемещение по оси Y;
- L - состояние левой кнопки (0 - нажата, 1 - отпущена);
- M - состояние средней кнопки (0 - нажата, 1 - отпущена);
- R - состояние правой кнопки (0 - нажата, 1 - отпущена).

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). Столь хитрый 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, то есть противоположно оси Y дисплея.

Для работы с устройством типа PC Mouse необходимо установить следующие параметры последовательного порта: скорость приема-передачи 1200 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.

4. Указания по подготовке к лабораторной работе.

При подготовке к лабораторной работе, ее выполнении и оформлении отчета необходимо следовать

рекомендациям раздела 4.1."Общие указания по выполнению и оформлению лабораторных работ".

Подготовка к данной лабораторной работе заключается в изучении описания лабораторной работы "Манипулятор PS/2 Mouse" и разделов 2.1. - 2.1.3 и 2.1.4.3 учебного пособия. Необходимо также ознакомиться с основными командами отладчика AFD и с командами процессора i8086.

5. Порядок выполнения лабораторной работы.

1. Ознакомиться с описанием электрической схемы универсальной COM мыши (Serial Mouse) и описанием ее контрольных точек.
2. Загрузить операционную систему Windows 95/98/Mel. в режиме Command Prompt Only, нажимая клавишу F8 перед началом загрузки ОС. При этом драйвер Serial Mouse мыши не загружается и на манипулятор не подается питание положительной полярности. В результате она остается в состоянии остановки передачи данных (Disable). В этом состоянии она выполняет все внутренние операции, связанные с перемещением, нажатием и отпусканием кнопок, однако не передает данные по линии интерфейса.
3. С помощью электронного осциллографа определить электрические параметры сигналов в контрольных точках электрической схемы манипулятора (состояние контрольных точек), находящегося в состоянии Disable (амплитуда, длительность и период сигналов или напряжение постоянного уровня):
   3.1. Определить состояние контрольных точек 1, 2, 3, и 4 при нахождение Serial Mouse в состоянии покоя, при передвижении мыши по вертикали и горизонтали и при нажатии клавиш.
   3.2. Определить состояние контрольных точек 5, 6 и 7 при нахождение Serial Mouse в состоянии покоя, при передвижении мыши и нажатии клавиш.
   3.3. Определить состояние контрольной точки 8 при нахождение переключателя SW4 в положение 2 и 3.
   3.4. Определить состояние контрольных точек 9, 10 и 11 при нахождение Serial Mouse в состоянии покоя, при нажатие кнопок и при перемещении манипулятора.
   3.5. Определить состояние контрольных точек 12 и 13.
4. Загрузить программу - драйвер Gmouse.com.
   4.1. Установить переключатель SW4 в положение 3. Определить состояние контрольных точек 9, 10 и 11 при нажатие и отпускание левой средней и правой кнопок мыши и при движение вверх, вниз, влево и вправо.
   4.2. Установить переключатель SW4 в положение 2. Определить состояние контрольных точек 9, 10 и 11 при нажатие и отпускание левой средней и правой кнопок мыши и при движение вверх, вниз, влево и вправо.
5. В среде AFD набрать программу 1 (**ms_m3.bin)**, представленную в листинге 1 раздела 6 описания. Установить переключатель SW4 в положение 2. Запустить программу командой G отладчика. В соответствии с описание программы 1 получить следующую информацию:
   5.1. Определить код идентификации мыши.
   5.2. Определить коды нажатия и отпускания левой и правой кнопок.
   5.3. Определить параметры сигналов передачи пакетов мышью при нажатие и отпускание левой и правой кнопок (контрольная точка 10).
   5.4. Определить признаки кодов передвижения мыши вверх, вниз, вправо и влево.
6. В среде AFD набрать программу 2 (**`mpc_m3.bin`**), представленную в листинге 2 раздела 6 описания. Установить переключатель SW4 в положение 3. Запустить программу командой G отладчика. В соответствии с описание программы 2 получить следующую информацию:
   6.1. Определить код идентификации мыши.
   6.2. Определить коды нажатия и отпускания левой, средней и правой кнопок.

6.3. Определить параметры сигналов передачи пакетов мышью при нажатие и отпускание левой, средней и правой кнопок (контрольная точка 10).

6.4. Определить признаки кодов передвижения мыши вверх, вниз, вправо и влево.

7. Получить дополнительное задание у преподавателя.

6. Примеры программ.

Программа 1 (ms_m3.bin), представленная в листинге 1 реализует прием от двухкнопочной MS Mouse 8 трехбайтовых пакетов данных и размещение их в буфере памяти с адреса DS:0000h. В программе предусмотрены процедуры инициализации контроллера порта COM 1, подача на манипулятор питания положительной полярности, поиск начального байта первого пакета. Программа выполняется в среде отладчика AFD.

Листинг 1. Прием пакетов данных от двухкнопочной MS Mouse (ms_m3.bin).

```
2BF9:0100  BF0000  MOV    DI,0000   ;Позиционирование на
2BF9:0103  8CD8    MOV    AX,DS     ;начало буфера
2BF9:0105  8EC0    MOV    ES,AX     ;памяти (DS:0000h).
2BF9:0107  BB2400  MOV    BX,0018   ;Счетчик принимаемых байт (24 байта).
2BF9:010A  BAFB03  MOV    DX,03FB   ;Установка разряда
2BF9:010D  B085    MOV    AL,85     ;DLAB (D7) в регистре
2BF9:010F  EE      OUT    DX,AL     ;LCR контроллера COM-порта.
2BF9:0110  B060    MOV    AL,60     ;Занесение в делитель
2BF9:0112  BAF803  MOV    DX,03F8   ;контроллера константы
2BF9:0115  EE      OUT    DX,AL     ;деления 0060h,
2BF9:0116  B000    MOV    AL,00     ;которая устанавливает
2BF9:0118  BAF903  MOV    DX,03F9   ;скорость
2BF9:011B  EE      OUT    DX,AL     ;передачи 1200 бод.
2BF9:011C  BAFB03  MOV    DX,03FB   ;передачи 1200 бод.
2BF9:011F  B002    MOV    AL,02     ;длина - 7 бит, без контроля четности,
2BF9:0121  EE      OUT    DX,AL     ;1 стоповый бит.
2BF9:0122  B003    MOV    AL,03     ;Включение питания мыши (+12 в) на
2BF9:0124  BAFC03  MOV    DX,03FC   ;линии DTR и RTS через
2BF9:0127  EE      OUT    DX,AL     ;регистр MCR контроллера COM-порта.
2BF9:0128  B401    MOV    AH,01     ;Признак поиска начального байта.
2BF9:012A  BAFD03  MOV    DX,03FD   ;Проверка: принят ли
2BF9:012D  EC      IN     AL,DX     ;COM-портом
2BF9:012E  A801    TEST   AL,01     ;очередной, если нет,
2BF9:0130  74F8    JZ     012A      ;то новый цикл проверки.
2BF9:0132  BAF803  MOV    DX,03F8   ;Чтение в регистр-аккумулятор AL
2BF9:0135  EC      IN     AL,DX     ;принятого символа.
2BF9:0136  F6C401  TEST   AH,01     ;Был ли уже принят начальный байт?
2BF9:0139  7406    JZ     0141      ;если да, то на запись в блок памяти.
2BF9:013B  A840    TEST   AL,40     ;Проверка признака начального байта, если
2BF9:013D  74EB    JZ     012A      ;нет признака, то на нов цикл чтения.
2BF9:013F  B400    MOV    AH,00     ;Признак: начальный байт принят.
2BF9:0141  268805  MOV    ES:[DI],AL ;Запись принятого байта в буфер.
2BF9:0144  47      INC    DI        ;Инкремент индекса DI.
2BF9:0145  4B      DEC    BX        ;Декремент счетчика принятых байт.
2BF9:0146  75E2    JNZ    012A      ;Если не нуль, то новый цикл чтения.
2BF9:0148  CC      INT3             ;Передача управления отладчику.
```

Результаты выполнения программы отображаются в окнах памяти экрана отладчика. В таблице 7 представлены результаты приема от манипулятора информации о последовательном нажатие и отпускание левой и правой кнопок (всего 8 событий) и движения мыши вверх, вниз, вправо и влево. Такой результат наблюдается после второго и последующих запусках программы. После первого запуска первым байтом будет отображаться код идентификации манипулятора (4Dh в нашем случае), который посылается мышью после подачи на нее питания положительной полярности. Поскольку в программе не предусмотрено отключение этого питания, то последующие запуски программы она будет принимать только коды изменения состояния мыши.

При движении мыши влево и вправо не всегда удается точно зафиксировать направление движения. Поэтому в таблицах могут присутствовать небольшие сдвиги по вертикали.

На рис. 2 - 5 представлены временные диаграммы на линии входных данных COM-порта, наблюдаемые с помощью электронного осциллографа при нажатие и отпускание левой и правой кнопок мыши. В формате передачи данных интервал A - стартовый бит, интервал B - стоповый бит или пауза. Цифрами обозначены

битовые интервалы соответствующих разрядов передаваемого символа (байта). Символы передаются начиная с младшего бита (D0).

Таблица 7. Состояние экрана 1 памяти AFD после выполнения программы 1.

	0	1	2	3	4	5	6	7
Нажатие и отпускание кнопок: сначала левой, затем правой без движения.								
DS:0000	60	00	00	40	00	00	50	00
DS:0008	00	40	00	00	60	00	00	40
DS:0010	00	00	50	00	00	40	00	00
Движение курсора вверх								
DS:0000	4C	00	3F	4C	00	34	4C	00
DS:0008	2A	4C	00	23	4C	00	27	4C
DS:0010	00	2E	4C	00	35	4C	00	3F
Движение курсора вниз								
DS:0000	40	00	01	40	00	05	40	00
DS:0008	09	43	3F	08	40	00	0F	40
DS:0010	00	10	40	00	16	40	00	1B
Движение курсора вправо								
DS:0000	40	01	00	40	05	00	40	05
DS:0008	00	40	09	00	40	0C	00	40
DS:0010	13	00	40	13	00	40	15	00
Движение курсора влево								
DS:0000	43	3F	00	43	3E	00	43	3A
DS:0008	00	43	34	00	43	2B	00	43
DS:0010	1F	00	43	18	00	43	17	00

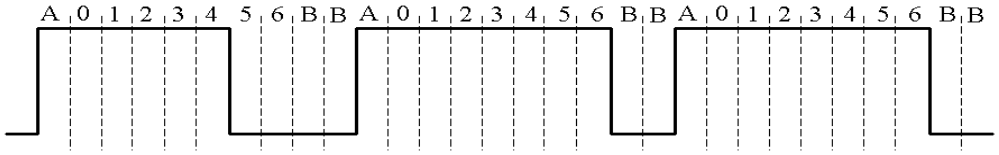

Рис. 2. байта информации при нажатие на левую кнопку:

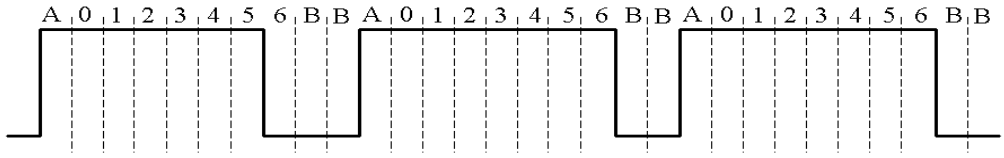

Рис.3. 3 байта информации при отпускании левой кнопки.

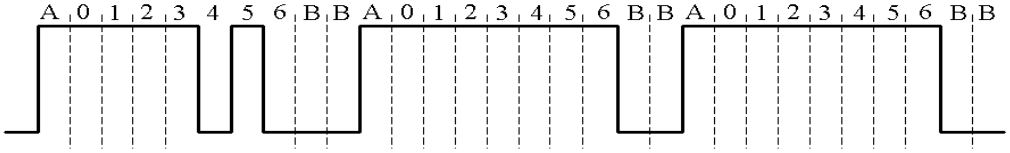

Рис.4. 3 байта информации при нажатии правой кнопки.

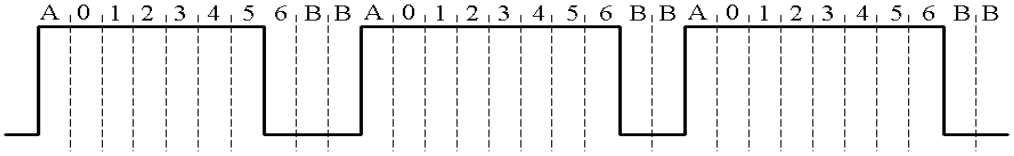

Рис.5. 3 байта информации при отпускании правой кнопки.

Программа 2 (**`mpc_m3.bin`**), представленная в листинге 2 реализует прием от трехкнопочной PC Mouse 12 пятибайтовых пакетов данных и размещение их в буфере памяти с адреса DS:0000h. В программе предусмотрены процедуры инициализации контроллера порта COM 1, подача на манипулятор питания положительной полярности, поиск начального байта первого пакета. Программа выполняется в среде отладчика AFD.

Листинг 2. Прием пакетов данных от трехкнопочной PC Mouse (`mpc_m3.bin`).

```
2BF9:0100  BF0000  MOV    DI,0000   ;Позиционирование на
2BF9:0103  8CD8    MOV    AX,DS     ;начало буфера
2BF9:0105  8EC0    MOV    ES,AX     ;памяти (DS:0000h).
2BF9:0107  BB3B00  MOV    BX,001D   ;Счетчик принимаемых байт (30 байт).
2BF9:010A  BAFB03  MOV    DX,03FB   ;Установка разряда
```

```
2BF9:010D  B085    MOV    AL,85     ;DLAB (D7) в регистре
2BF9:010F  EE      OUT    DX,AL     ;LCR контроллера COM-порта.
2BF9:0110  B060    MOV    AL,60     ;Занесение в делитель
2BF9:0112  BAF803  MOV    DX,03F8   ;контроллера константы
2BF9:0115  EE      OUT    DX,AL     ;деления 0060h,
2BF9:0116  B000    MOV    AL,00     ;которая устанавливает
2BF9:0118  BAF903  MOV    DX,03F9   ;скорость
2BF9:011B  EE      OUT    DX,AL     ;передачи 1200 бод.
2BF9:011C  BAFB03  MOV    DX,03FB   ;передачи 1200 бод.
2BF9:011F  B003    MOV    AL,03     ;длина - 8 бит, без контроля четности,
2BF9:0121  EE      OUT    DX,AL     ;1 стоповый бит.
2BF9:0122  B003    MOV    AL,03     ;Включение питания мыши (+12 в) на
2BF9:0124  BAFC03  MOV    DX,03FC   ;линии DTR и RTS через
2BF9:0127  EE      OUT    DX,AL     ;регистр MCR контроллера COM-порта.
2BF9:0128  B401    MOV    AH,01     ;Признак поиска начального байта.
2BF9:012A  BAFD03  MOV    DX,03FD   ;Проверка: принят ли
2BF9:012D  EC      IN     AL,DX     ;COM-портом
2BF9:012E  A801    TEST   AL,01     ;очередной, если нет,
2BF9:0130  74F8    JZ     012A      ;то новый цикл проверки.
2BF9:0132  BAF803  MOV    DX,03F8   ;Чтение в регистр-аккумулятор AL
2BF9:0135  EC      IN     AL,DX     ;принятого символа.
2BF9:0136  F6C401  TEST   AH,01     ;Был ли уже принят начальный байт?
2BF9:0139  740A    JZ     0145      ;если да, то на запись в блок памяти.
2BF9:013B  A880    TEST   AL,80     ;Проверка признака начального байта, если
2BF9:013D  74EB    JZ     012A      ;нет признака,(D7=0) - нов. цикл чтения.
2BF9:013F  A878    TEST   AL,78     ;если D6 - D3 не нули, то
2BF9:0141  75E7    JNZ    012A      ;новый цикл чтения.
2BF9:0143  B400    MOV    AH,00     ;Признак: начальный байт принят.
2BF9:0145  268805  MOV    ES:[DI],AL ;Запись принятого байта в буфер.
2BF9:0148  47      INC    DI        ;Инкремент индекса DI.
2BF9:0149  4B      DEC    BX        ;Декремент счетчика принятых байт.
2BF9:014A  75DE    JNZ    012A      ;Если не нуль, то новый цикл чтения.
2BF9:014C  CC      INT3             ;Передача управления отладчику.
```

Результаты выполнения программы отображаются в окнах памяти экрана отладчика. В таблице 8

Таблица 8. Состояние экрана 1 памяти AFD после выполнения программы 2.

	0	1	2	3	4	5	6	7
Нажатие и отпускание кнопок: сначала левой, затем средней и правой без движения.								
DS:0000	83	00	00	00	00	87	00	00
DS:0008	00	00	85	00	00	00	00	87
DS:0010	00	00	00	00	86	00	00	00
DS:0018	00	87	00	00	00	00	X	X
Движение курсора вверх								
DS:0000	87	00	01	00	01	87	00	01
DS:0008	00	02	87	01	03	00	05	87
DS:0010	00	05	00	09	87	00	04	00
DS:0018	04	87	00	02	00	08	X	X
Движение курсора вниз								
DS:0000	87	00	FF	00	FB	87	00	FF
DS:0008	00	FD	87	00	FF	00	00	87
DS:0010	00	FF	00	FD	87	00	FE	00
DS:0018	F9	87	00	F9	00	F9	X	X
Движение курсора вправо								
DS:0000	87	01	00	09	00	87	04	00
DS:0008	07	00	87	0B	00	0C	00	87
DS:0010	06	01	0C	00	87	0B	00	12
DS:0018	00	87	0B	00	0F	00	X	X
Движение курсора влево								
DS:0000	87	FF	00	FF	FF	87	FD	00
DS:0008	F7	FF	87	F8	00	EE	00	87
DS:0010	EF	00	F3	00	87	EA	00	D9
DS:0018	00	87	F8	00	DC	00	X	X

представлены результаты приема от манипулятора информации о последовательном нажатие и отпускание левой, средней и правой кнопок (всего 12 событий) и движения мыши вверх, вниз, вправо и влево. Код идентификации программа не регистрирует, поскольку он не проходит фильтр идентификации первого байта пакета (D7=1 и (D6 - D3)=0).

При движении мыши влево и вправо не всегда удается точно зафиксировать направление движения. Поэтому в таблицах могут присутствовать небольшие сдвиги по вертикали.

На рис. 6 - 8 представлены временные диаграммы сигналов на линии входных данных COM-порта, наблюдаемые с помощью электронного осциллографа при нажатие левой, средней и правой кнопок мыши, а на рис. 9 - временная диаграмма сигналов, формируемых при отпускании любой кнопки. В формате передачи данных интервал A - стартовый бит, интервал B - стоповый бит. Цифрами обозначены битовые интервалы соответствующих разрядов передаваемого символа (байта). Символы передаются начиная с младшего бита (D0).

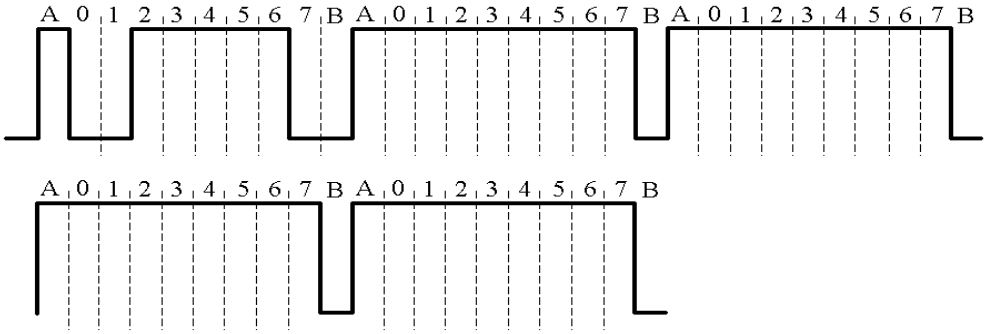

Рис.6. 5 байт информации при нажатии левой кнопки.

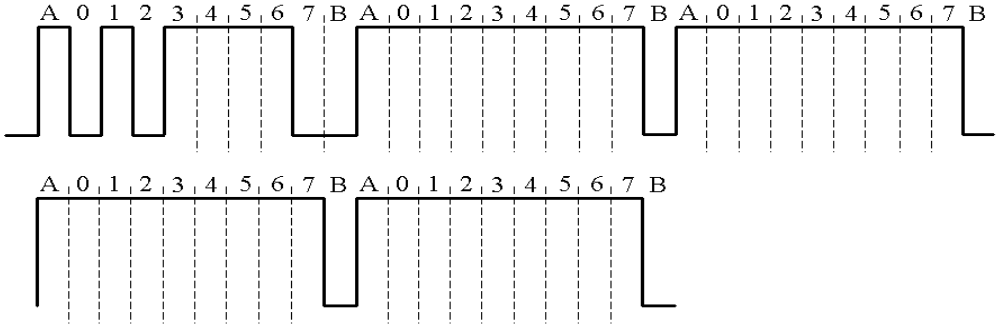

Рис.7. 5 байт информации при нажатии средней кнопки.

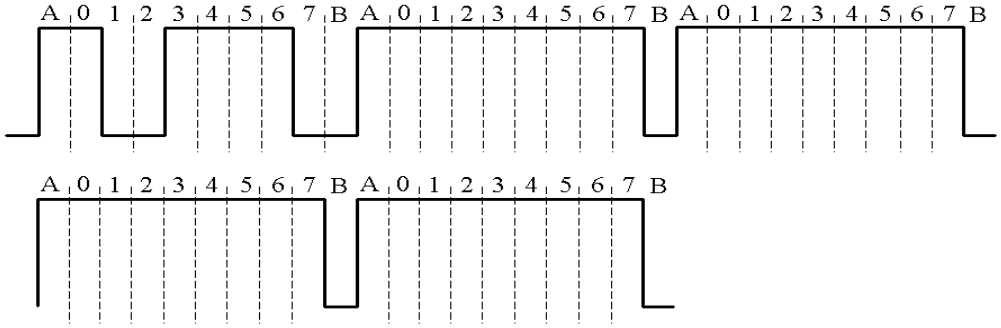

Рис.8. 5 байт информации при нажатии правой кнопки.

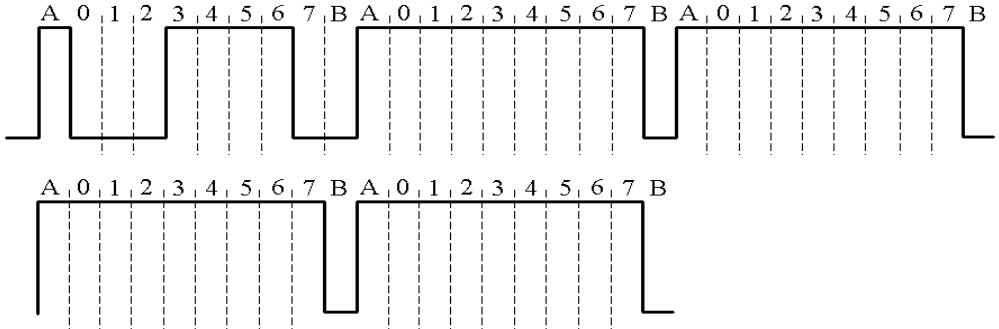

Рис.9. 5 байт информации при отпускании левой, средней и правой кнопок. порта COM1, инициализацию вектора прерывания INT 0Ch (IRQ 4), разрешение прерываний от порта COM1

Программа 3. В листинге 3 приведен программный модуль, реализующий: инициализацию, включение питания мыши и ожидание заполнения обработчиком прерывания буфера памяти (модуль CMPR.BIN), а в листинге 4 - программный модуль обработчика прерывания INT 0Ch (модуль CMINT.BIN). Программные модули сориентированы на их выполнение в среде отладчика AFD. Первый модуль загружается с адреса CS:0200h, а обработчик - с адреса CS:0100h. Запуск модуля CMPR.BIN осуществляется командой отладчика G 0200. Он прекращает свою работу после заполнения обработчиком прерывания INT 0Ch (модуль CMINT.BIN) буфера памяти размером 64 байта с базовым адресом DX:0000h и содержимое буфера отображается в окне памяти экрана отладчика AFD. После заполнения буфера питание мыши отключается, и запрещаются прерывания от контроллера порта COM 1.

Листинг 3 Программный модуль CMPR.TXT

```
2C06:0200  BAFB03    MOV    DX,03FB ;Установка в регистре LCR (адр. 3FBh)
2C06:0203  B080      MOV    AL,80   ;признака SET=1 (D7), разрешающего обраще-
2C06:0205  EE        OUT    DX,AL   ;ние к регистрам делителя.
2C06:0206  B060      MOV    AL,60   ;Установка скорости приема/передачи 1200
2C06:0208  BAF803    MOV    DX,03F8 ;бод, занесением в младший байт делителя
2C06:020B  EE        OUT    DX,AL   ;константы 60h,
2C06:020C  B000      MOV    AL,00   ;а в старший
2C06:020E  BAF903    MOV    DX,03F9 ;байт делителя
2C06:0211  EE        OUT    DX,AL   ;константы 00h.
2C06:0212  BAFB03    MOV    DX,03FB ;Установка режима (без контроля паритета,
2C06:0215  B006      MOV    AL,06   ;2 стоповых бите и 7 бит в поле данных),
2C06:0217  EE        OUT    DX,AL   ;занесением в LCR константы 06h.
2C06:0218  BAF903    MOV    DX,03F9 ;Запрещение выработки контроллером порта
2C06:021B  B000      MOV    AL,00   ;COM1 сигнала IRQ4 по всем событиям,
2C06:021D  EE        OUT    DX,AL   ;происходящим в порту (00h → IER).
2C06:021E  E421      IN     AL,[21] ;Разрешение обработки прерывания IRQ4
2C06:0220  24EF      AND    AL,EF   ;ведущим контроллером прерываний
2C06:0222  EB00      JMP    0224    ;установкой в единицу разряда D4
2C06:0224  EB00      JMP    0226    ;его регистра маски (адрес 21h).
2C06:0226  E621      OUT    [21],AL
2C06:0228  B80000    MOV    AX,0000 ;Занесение в регистр сегмента ES CPU
2C06:022B  8EC0      MOV    ES,AX   ;адреса 0000h.
2C06:022D  B80001    MOV    AX,0100 ;Формирование вектора прерывания INT 0Ch
2C06:0230  26A33000  MOV    ES:[0030],AX  ;по адресу 0000:0030h, указывающего
2C06:0234  8CC8      MOV    AX,CS   ;адрес обработчика прерывания, начинающе-
2C06:0236  26A33200  MOV    ES:[0032],AX  ;гося с адреса CS:0100h.
2C06:023A  BAFD03    MOV    DX,03FD ;Сброс флаговых разрядов регистра LSR
2C06:023D  EC        IN     AL,DX   ;его чтением по адресу 3FDh.
2C06:023E  BAFE03    MOV    DX,03FE ;Сброс флаговых разрядов регистра MSR
2C06:0241  EC        IN     AL,DX   ;его чтением по адресу 3FEh.
2C06:0242  BAFA03    MOV    DX,03FA ;Запрещение использования буфера FIFO
2C06:0245  B006      MOV    AL,06   ;в контроллере COM порта, записью
2C06:0247  EE        OUT    DX,AL   ;кода 06h по адресу 3FAh (регистр FCR).
2C06:0248  BAF903    MOV    DX,03F9 ;Разрешить выработку запроса прерывания
2C06:024B  B001      MOV    AL,01   ;IRQ4 при очередном приеме байта данных
2C06:024D  EE        OUT    DX,AL   ;во входной буферный регистр RBR.
2C06:024E  B90000    MOV    CX,0000 ;Сброс счетчика принятых байтов.
2C06:0251  B00B      MOV    AL,0B   ;Подача питания на мышь и разрешение
2C06:0253  BAFC03    MOV    DX,03FC ;генерации контроллером сигнала IRQ4
2C06:0256  EE        OUT    DX,AL   ;по событию, вызывающему прерывание.
2C06:0257  F6C180    TEST   CL,40   ;Проверка заполнения обработчиком
2C06:025A  74FB      JZ     0257    ;буфера памяти, размером 64 байта.
2BF9:025C  B000      MOV    AL,00   ;Отключение питания мыши (+12 В)
2BF9:025E  EE        OUT    DX,AL   ;и запрещение сигнала прерывания.
2BF9:025F  CC        INT3           ;Останов и передача управления в AFD.
```

Листинг 2.10. Обработчик прерывания INT 0Ch (линия IRQ4) (CMINT.BIN).

```
2C06:0100  52        PUSH   DX      ;Запоминание в стеке
2C06:0101  57        PUSH   DI      ;содержимого
2C06:0102  50        PUSH   AX      ;общих регистров
2C06:0103  06        PUSH   ES      ;процессора: DX,DI, AX, ES.
2C06:0104  BAFD03    MOV    DX,03FD ;Сброс флаговых разрядов регистра LSR
2C06:0107  EC        IN     AL,DX   ;его чтением по адресу 3FDh.
2C06:0108  BAFE03    MOV    DX,03FE ;Сброс флаговых разрядов регистра MSR
```

```
2C06:010B  EC        IN     AL,DX   ;его чтением по адресу 3FEh.
2C06:010C  8CD8      MOV    AX,DS   Позиционирование на сегмент буфера
2C06:010E  8EC0      MOV    ES,AX   ;памяти обработчика.
2C06:0110  B80000    MOV    AX,0000 ;Очистка регистра AX.
2C06:0113  BAF803    MOV    DX,03F8 ;Чтение очередного байта данных из
2C06:0116  EC        IN     AL,DX   ;буферного регистра RBR приемника.
2C06:0117  89CF      MOV    DI,CX   Позиционирование на очередной байт буф.
2C06:0119  268805    MOV    ES:[DI],AL  ;Запись очередного байта в буфер.
2C06:011C  41        INC    CX      ;Инкрементирование счетчика байтов.
2C06:011D  B020      MOV    AL,20   ;Посылка команды EOI (End Of Interrupt)
2C06:011F  E620      OUT    [20],AL ;в ведущий контроллер прерыв. (адр. 20h).
2C06:0121  07        POP    ES      ;Восстановление из стека
2C06:0122  58        POP    AX      ;содержимого
2C06:0123  5F        POP    DI      ;общих регистров
2C06:0124  5A        POP    DX      ;процессора: ES, AX, DI, DX.
2C06:0125  FB        STI            ;Разрешение прерываний.
2C06:0126  CF        IRET           ;Возврат из обработчика прерываний
```

7. Рекомендуемая литература.

При подготовке к лабораторной работе и к ее защите рекомендуется пользоваться информацией, представленной в описании лабораторной работы, в разделе 2 и приложении 1 учебно-методического пособия, а также в дополнительной литературе [5, 6, 9, 13, 15].

8. Контрольные вопросы

1. К какому типу устройств относятся манипуляторы мышь?
2. Какую информацию передает в систему COM мышь?
3. Как преобразуется механическое перемещение мыши в электрические сигналы?
4. Что понимается под названием "шинная мышь"?
5. Что представляет собой манипулятор трекбол?
6. Охарактеризуйте оптические манипуляторы мышь.
7. Что понимается под 3D-мышью?
8. По каким интерфейсам могут подключаться к системе манипуляторы мышь?
9. Чем отличается Serial Mouse от PS/2-Mouse?
10. Можно ли подключить PS/2-Mouse к COM-порту через переходник?
11. Можно ли подключить USB-мышь через переходник к порту PS/2-мыши?
12. Как реагирует ОС на подключение к системе нескольких устройств-указателей?
13. Охарактеризуйте основные причины сбоев в работе манипуляторов мышь.
14. Охарактеризуйте особенности "беспроводных мышей".
15. Как подключается к ПК Serial Mouse?
16. Охарактеризуйте две разновидности Serial Mouse.
17. Охарактеризуйте системную поддержку Serial Mouse на уровне BIOS и DOS.
18. Охарактеризуйте электрическую схему универсальной Serial Mouse.
19. Охарактеризуйте сигналы в контрольных точках электрической схемы Serial Mouse.
20. Охарактеризуйте особенности питания маломощных устройств от выходных линий COM-порта.
21. Охарактеризуйте стандартный формат передачи данных манипулятора NS Mouse.
22. Охарактеризуйте формат Microsoft Plus (M+) трехкнопочной мыши.
23. Охарактеризуйте формат 3D Serial Mouse.
24. Охарактеризуйте нестандартный 8-разрядный формат MS Mouse с двумя альтернативными протоколами передачи данных.
25. Охарактеризуйте формат PC Mouse.
26. Охарактеризуйте требования к средствам работы с Serial Mouse.
27. Кто устанавливает обработчик прерывания INT 33h?
28. Какие функции выполняет прерывание INT 33h?
29. Охарактеризуйте функцию 0000h прерывания INT 33h.
30. Охарактеризуйте функцию 0001h прерывания INT 33h.
31. Охарактеризуйте функцию 0002h прерывания INT 33h.
32. Объясните работу программы, представленной в листинге 2.1.
33. Охарактеризуйте функцию 0003h прерывания INT 33h.
34. Объясните работу программы, представленной в листинге 2.2.
35. Охарактеризуйте функцию 0004h прерывания INT 33h.
36. Охарактеризуйте функцию 0005h прерывания INT 33h.
37. Охарактеризуйте функцию 0006h прерывания INT 33h.
38. Охарактеризуйте функцию 0007h и 0008h прерывания INT 33h.

39. Охарактеризуйте функцию 0009h прерывания INT 33h.
40. Объясните работу программы, представленной в листинге 2.3.
41. Охарактеризуйте функцию 000Ah прерывания INT 33h.
42. Объясните работу программы, представленной в листинге 2.4.
43. Охарактеризуйте функцию 000Bh прерывания INT 33h.
44. Охарактеризуйте функцию 000Ch прерывания INT 33h.
45. Объясните работу программы, представленной в листинге 2.5.
46. Объясните работу программы, представленной в листинге 2.6.
47. Охарактеризуйте функцию 000Dh, 000Eh и 000Fh прерывания INT 33h.
48. Охарактеризуйте функцию 0010h и 0012h прерывания INT 33h.
49. Охарактеризуйте функцию 0013h и 0014h прерывания INT 33h.
50. Охарактеризуйте функцию 0015h, 0016h и 0017h прерывания INT 33h.
51. Охарактеризуйте функцию 0018h прерывания INT 33h.
52. Охарактеризуйте функцию 0019h и 001Ah прерывания INT 33h.
53. Охарактеризуйте функцию 001Bh и 001Ch прерывания INT 33h.
54. Охарактеризуйте функцию 001Dh и 001Fh прерывания INT 33h.
55. Охарактеризуйте функцию 0020h и 0021h прерывания INT 33h.
56. Охарактеризуйте функцию 0024h прерывания INT 33h.
57. Охарактеризуйте процесс установки нового драйвера мыши.
58. Как разрешается прерывание от Serial Mouse?
59. Как настраивается вектор прерывания COM-порта?
60. Как подается питание на Serial Mouse и разрешение контроллеру COM-порта генерировать запросы на прерывание?
61. Как определить: подключена ли к COM-порту мышь?
62. Какие действия выполняет "новый" драйвер мыши?
63. Объясните работу процедур, представленных в листинге 2.7.
64. Объясните работу программы, представленной в листинге 2.8.
65. Объясните работу программного модуля, представленного в листинге 2.9.

ПРИЛОЖЕНИЕ

1. Контроллер прерываний

Устройства ввода/вывода данных с помощью запросов прерываний сообщают центральному процессору о поступлении новой информации от объектов внешнего мира или о готовности получить очередной квант данных от процессора. Прерывания от клавиатуры, мыши PS/2-типа и других периферийных устройств прежде, чем поступить в процессор, проходят через контроллер прерываний, где подвергаются предварительной обработке. Контроллер позволяет управлять приоритетами, прохождением запросов и адресами векторов прерываний.

Контроллер прерываний IBM PC/AT состоял из двух микросхем Intel 8259A, включенных каскадно (рис. 1.). Первая микросхема была ведущей, а вторая — ведомой (ведомый контроллер подключен к входу IRQ2 ведущего). На входы IRQ0, IRQ1, IRQ3-IRQ7 ведущей микросхемы и на входы IRQ8-IRQ15 ведомой поступают запросы прерываний, из которых выбирается немаскированный запрос с наивысшим приоритетом, после чего контроллер вырабатывает сигнал INT и передает в процессор номер вектора прерывания.

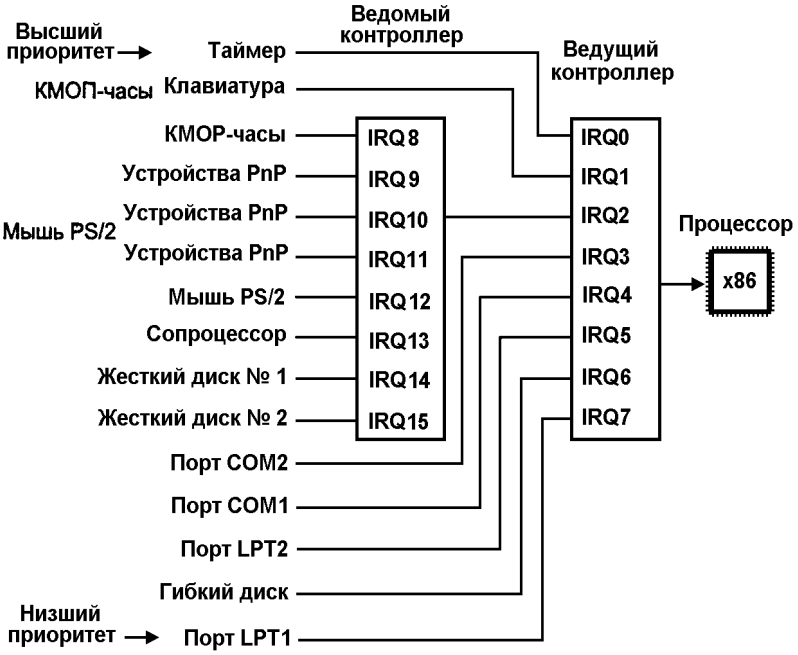

Рис. 1. Традиционный порядок подключения внешних устройств к контроллеру прерываний

Номер вектора формируется контроллером прерывания путем сложения базового значения (записанного в соответствующий регистр контроллера на стадии инициализации системы) и номера линии, на которую поступил запрос (ведущей микросхеме IRQ0 соответствует линия 0, IRQ7 — линия номер 7; ведомой микросхеме IRQ8 соответствует линия 0, IRQ15 — линия 7). Базовый вектор ведущей микросхемы в реальном режиме DOS имеет значение 08h, базовый вектор ведомой — 70h. Соответственно, ведущая схема вырабатывает вектора с номерами 08h-0Fh, ведомая — с номерами 70h-77h (табл. 1). Приоритеты запросов прерывания (по убыванию) располагаются в следующем порядке: IRQ0, IRQ1, IRQ8 - IRQ15, IRQ3 - IRQ7.

Все последующие модели AT-совместимых компьютеров вынуждены имитировать работу микросхем i8259A с целью сохранения совместимости со старым программным обеспечением. Большая часть возможностей указанных микросхем, к счастью для программистов, не используется в AT-совместимых персональных компьютерах: приоритеты прерываний, поступающих от периферийных устройств, и адреса соответствующих векторов жестко зафиксированы (устанавливаются BIOS в процессе начальной загрузки). Полностью перепрограммировать контроллер приходится только при переключении процессора в защищенный режим, так как при этом необходимо изменить номера векторов прерываний, а работа с контроллером в обычном реальном режиме DOS требует выполнения только двух типов операций:

- маскирование и размаскирование прерываний по отдельным линиям;
- посылка команды завершения обработки прерывания контроллеру.

Чтобы маскировать (запретить) прерывание, необходимо установить в 1 соответствующий ему бит в регистре маски (номера разрядов маски соответствуют номерам линий сигналов прерывания). Поскольку в маске должен быть изменен только один разряд, а остальные нужно сохранить в исходном состоянии, то вначале требуется прочитать содержимое регистра маски, выполнить операцию изменения соответствующего разряда, а затем записать полученное значение обратно в регистр маски. Между двумя последовательными обращениями к одному и тому же порту рекомендуется вставлять циклы задержки, поскольку контроллер по сравнению с процессором работает слишком медленно. Регистр маски прерываний

ведущего контроллера доступен для записи и считывания через порт 21h, ведомого контроллера — через порт Alh.

Таблица 1. Аппаратные прерывания AT-совместимых компьютеров

Преры-вание	Номер вектора	Адрес вектора	Источник сигнала прерывания
IRQ0	08h	0000:0020h	Системный таймер
IRQ1	09h	0000:0024h	Клавиатура
IRQ2	0Ah	0000:0028h	Ведомая микросхема контроллера
IRQ3	0Bh	0000:002Ch	Последовательный порт COM2
IRQ4	0Ch	0000:0030h	Последовательный порт COM1
IRQ5	0Dh	0000:0034h	Параллельный порт LPT2
IRQ6	0Eh	0000:0038h	Контроллер дисководов гибких дисков
IRQ7	0Fh	0000:003Ch	Параллельный порт LPT1
IRQ8	70h	0000:01C0h	Часы реального времени
IRQ9	71h	0000:01C4h	Любое устройство PnP
IRQ10	72h	0000:01C8h	Любое устройство PnP
IRQ11	73h	0000:01CCh	Любое устройство PnP
IRQ12	74h	0000:01D0h	Мышь PS/2-типа
IRQ13	75h	0000:01D4h	Математический сопроцессор
IRQ14	76h	0000:01D8h	Контроллер жесткого диска № 1
IRQ15	77h	0000:01DCh	Контроллер жёсткого диска № 2

Например, чтобы запретить прохождение сигнала прерывания от клавиатуры IRQ1 (ведущий контроллер, линия № 1), нужно выполнить следующий ряд команд:

```
; Прочитать регистр маски ведущего контроллера
      in      AL,21h
; Установить второй разряд маски
      or      AL,10b
; Вставить задержку
      jmp short $+2
      jmp short $+2
; Записать маску обратно в регистр
      out     21h,AL
```

Когда программист устанавливает собственный обработчик прерывания, он должен проделать обратную операцию — размаскировать линию сигнала запроса от соответствующего устройства. Например, для драйвера мыши PS/2-типа, обрабатывающего прерывание IRQ12 (ведомый контроллер, линия № 4), участок кода, в котором выполняется размаскирование, выглядит следующим образом:

```
; Прочитать регистр маски ведомого контроллера
      in      AL,0A1h
; Обнулить четвертый разряд маски
      and     AL,11101111b
; Вставить задержку
      jmp short $+2
      jmp short $+2
; Записать маску обратно в регистр
      out     0A1h,AL
```

Обработка сигнала прерывания выполняется контроллером следующим образом. Вначале выполняется проверка маски, и если запрос не запрещен, то происходит проверка его приоритета. Если нет других запросов или данный запрос имеет наивысший приоритет, то контроллер посылает микропроцессору сигнал запроса INT. Если маскируемые прерывания разрешены (установлен флаг IF в регистре флагов процессора), то процессор выдает контроллеру сигнал подтверждения прерывания INTA. После получения сигнала INTA контроллер выдает процессору номер вектора прерывания и переводит запрос в разряд обслуживаемых (соответствующий бит в регистре поступивших запросов контроллера сбрасывается, а бит в регистре обслуживаемых запросов устанавливается). Процессор начинает обработку прерывания с того, что записывает в стек содержимое регистра флагов, адрес очередной команды прерываемой программы и сбрасывает флаг IF. В результате перечисленных выше действий в момент начала обработки прерывания все маскируемые прерывания в процессоре запрещены, а контроллер не пропускает запросы прерываний, приоритеты которых ниже, чем у запроса, обрабатываемого процессором. Кроме того, следующие прерывания от устройства, запрос которого поступил на обработку, также будут заблокированы контроллером. Процессор должен разрешить обработку маскируемых прерываний (установить флаг IF командой STI), как только будет выполнен критический участок кода процедуры обработки прерывания, то есть группа операций, которую прерывать нельзя. Часто процедура вообще не содержит критических участков, и тогда команду STI следует установить в самом начале обработчика прерывания. С другой стороны, весь код процедуры может не допускать прерывания: в этом случае команда STI не используется, а

флаг прерываний автоматически восстанавливается при выходе из процедуры по команде IRET.

Разрешить контроллеру обрабатывать прерывания с таким же или более низким приоритетом следует как можно раньше, чтобы не возникала угроза потери информации от низкоприоритетных устройств. Как только будет завершен участок кода, не допускающий повторного прерывания от того же устройства, необходимо послать контроллеру команду EOI (End Of Interrupt), код которой в шестнадцатеричном виде имеет значение 20h. Регистр команд ведущей микросхемы доступен для записи через порт 20h, регистр команд ведомой — через порт A0h. Например, в процедуру обработчика прерывания от клавиатуры обязательно должна быть включена последовательность команд:

```
; Записать в регистр AL код команды EOI
      mov     AL,20h
; Послать команду EOI в ведущую микросхему
      out     20h,AL
```

Прерывание, поступившее через ведомую микросхему, блокирует также и обработку всех прерываний с более низкими приоритетами в ведущей. Поэтому процедура обработки такого прерывания должна посылать команду EOI не только в ведомый, но и в ведущий контроллер:

```
; Записать в регистр AL код команды EOI
      mov     AL,20h
; Послать команду EOI в ведомую микросхему
      out     0A0h,AL
; Послать команду EOI в ведущую микросхему
      out     20h,AL
```

В этом приложении использованы материалы из [6].

Более подробную информацию по организации аппаратных прерываний и программированию контроллеров прерывания PC/AT-подобных машин можно найти в [8, 13, 14, 15 и 17].

Контрольные вопросы

1. Для чего используются запросы аппаратных прерываний?
2. Чем управляет контроллер прерываний ПК?
3. Охарактеризуйте структуру контроллера прерываний IBM PC/AT.
4. Как формируется номер вектора прерывания в контроллере прерываний IBM PC/AT?
5. Какую систему приоритетов поддерживает контроллер прерываний IBM PC/AT?
6. Охарактеризуйте информацию, представленную в таблице П1.
7. Когда возникает необходимость полного перепрограммирования контроллера прерываний PC/AT-совместимых компьютеров?
8. Какие два типа операций достаточно выполнять при работе с контроллером прерываний в режиме DOS?
9. Охарактеризуйте правила маскирования линий запросов прерываний.
10. Объясните работу программных модулей, реализующих операции маскирования и размаскирования прерываний.
11. Как контроллер прерывания обрабатывает сигналы запроса прерываний?
12. Каким образом можно установить флаг IF в регистре флагов процессора?
13. Когда необходимо посылать в контроллер прерываний команду EOI (20h)?
14. Как посылается команда EOI при обработке прерываний ведущей и ведомой микросхем контроллера прерываний?

2. Полноэкранный отладчик реального режима

Введение

Полноэкранный отладчик AFD (аналогичен SDT) обеспечивает режим интерактивного доступа ко всем ресурсам ПК типа IBM PC/AT и удобный интерфейс для отладки программ.

Во время работы AFD на экране отображается содержимое всех регистров процессора, четыре верхних элемента стека и до девяти строк дисассемблированного кода. Кроме того имеется два независимых окна на память, которые позволяют отображать содержимое ячеек оперативной памяти в шестнадцатеричном виде и в коде ASCII (КОИ-8). Дисассемблированный код или данные в шестнадцатеричном виде или в коде ASCII могут быть выведены на печать или в файл. Интерфейс с пользователем организован очень продуманно, обеспечивая максимальное удобство в работе.

Наиболее часто используемые функции могут быть выполнены с помощью функциональных клавиш. Команды имеют длину от одного до двух символов и для проверки их синтаксиса можно, не прерывая работы, вызывать вывод на экран справочной информации. Шаг выполнения программы может быть выполнен нажатием единственной клавиши. Даже процедуры, которые вызываются отлаживаемой программой с помощью CALL или INT могут быть выполнены нажатием единственной клавиши. Любая ошибка ввода вызывает печать сообщения об ошибке и курсор позиционируется на символ, где во время разбора команды была обнаружена ошибка.

Введенные команды могут быть повторно извлечены из стека команд для повторения операции с помощью функциональной клавиши.

Команды и другая информация могут быть записаны в файл или во внутренний буфер для выполнения в качестве макрокоманды. Это позволяет выполнять запуск AFD из командного файла для установки точек останова по условиям или просто для выполнения последовательности команд с помощью макроопределения. Код загруженной программы выдается на экране в дисассемблированном виде, при этом поддерживается мнемоника всех команд микропроцессора типа i8088 и i80286, что позволяет читать код так же легко, как и исходный код с помощью экранного редактора текстов.

Машинные инструкции в коде программы могут быть легко изменены с помощью встроенного в AFD ассемблера. Для трансляции новой инструкции курсор устанавливается в окне дисассемблированного текста на инструкцию, которую нужно заменить, инструкция может быть изменена вводом новых символов поверх дисассемблированного текста.

Прикладные программы, использующие вывод данных на экран, могут использовать для этого другой дисплейный адаптер и экран, чтобы избежать смешения выходных данных программы с выводом на экран, выполняемым AFD. Это возможно, если на микро-ЭВМ установлены как монохромный, так и цветной графический адаптеры. При работе на ЭВМ с одним экраном пользователь может выбрать режим альтернативного экрана для разделения данных, выводимых AFD и прикладной программой. Переключение между экранами выполняется с помощью нажатия единственной клавиши.

Обращение к программе

Для запуска AFD его необходимо загрузить в память с помощью ввода команды: {устр:} AFD {фспец} {парам} {"команда AFD} в ответ на подсказку DOS (Например,'C:>').

В фигурных скобках указаны необязательные параметры. Параметр 'устр:' должен быть указан и определять дисковод, на котором находится AFD.COM (AFD.COM), если он отличен от дисковода, используемого по умолчанию. Если указан параметр 'фспец.', то для указанного файла после запуска AFD выполняется команда загрузки 'L'.

После спецификации файла, если нужно, могут быть заданы параметры, которые помещаются в префиксный сегмент (PSP) отлаживаемой программы.

Если AFD запускается без необязательных параметров, то выводится экран с названием программы. Для продолжения работы необходимо нажать какую-нибудь клавишу, после чего появляется основной экран AFD (SDT) рис. 1. Если при запуске AFD не передано ни одной команды перемещения курсора, то курсор устанавливается в поле командной строки (на рисунке в командной строке набрана команда g,CS:236). (Курсор - это полностью заполненное мерцающее знакоместо, если редактирование экрана AFD выполняется в режиме замены, или половина знакоместа в режиме вставки). Все команды вводятся в этом режиме. В дальнейшем мы будем ссылаться на эту часть экрана как на *командную строку*.

Поле над командной строкой показывает содержимое всех регистров процессора и четыре верхних элемента стека. Эту часть экрана в дальнейшем будем называть *областью регистров*. Имеются два регистра сегмента - HS и FS, которые не реализованы в процессоре аппаратно. Эти два регистра могут служить как вспомогательные. Они используются также для идентификации сегмента после выполнения команд поиска или сравнения.

Регистр HS (вспомогательный регистр сегмента) используется в командах поиска и сравнения. FS (фиксированный регистр сегмента) используется только в команде сравнения, если для адресации не может быть использовано содержимое никакого другого сегментного регистра.

Кроме того, эти регистры могут использоваться при определении значения регистра сегмента для адресации окна в памяти, которое не должно меняется даже, если отлаживаемая программа изменяет содержимое регистров сегментов. Пользователь может обращаться к ним так же, как к другим регистрам.

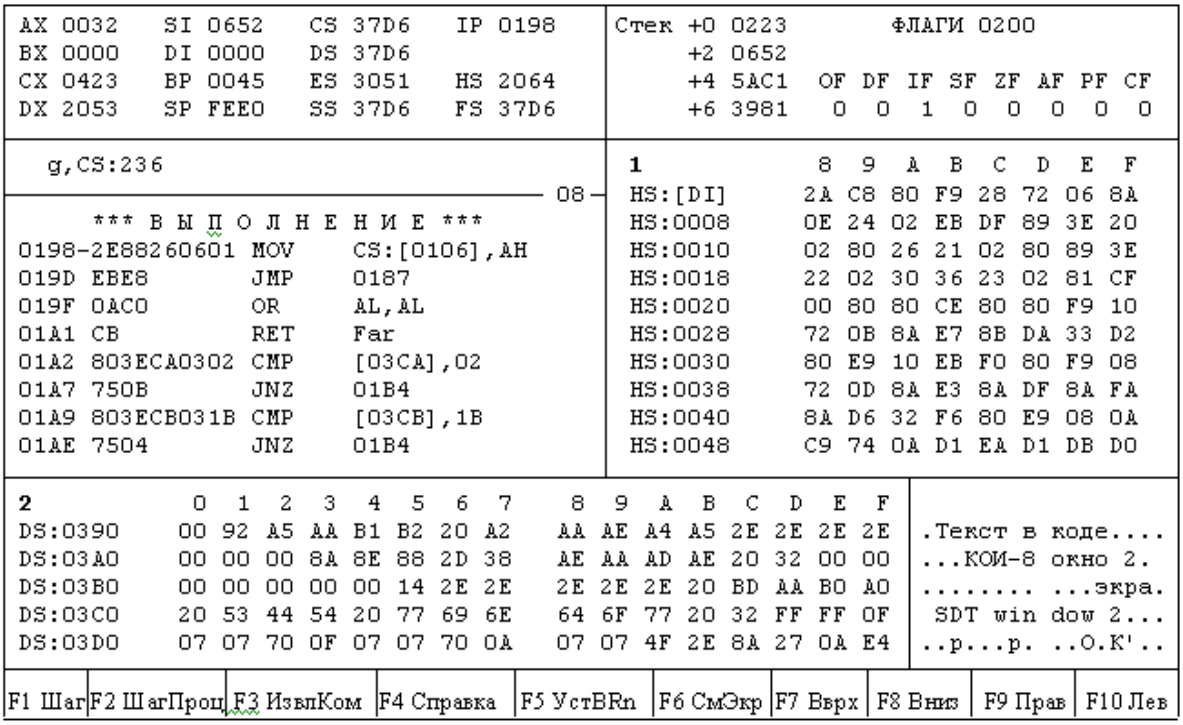

Рис 1. Основной экран AFD

Показанные флаги процессора имеют два представления. Первое - в виде 16-разрядного слова и второе, когда каждый из восьми флаговых разрядов (OF, DF, IF, SF, ZF, AF, PF и CF) представлен двоичным значением. Все неиспользуемые разряды слова флагов установлены в ноль для облегчения интерпретации 16-разрядного слова флагов. Флаг трассировки используется AFD и поэтому он недоступен пользователю. Флаги NT и IOPL в случае микропроцессора типа i80286 не имеют значения при работе с реальными адресами, используемыми операционной системой ДОС. Поэтому их содержимое не показано отдельными значениями, а может быть проконтролировано только при интерпретации флагового регистра, представленного в виде 16-разрядного слова.

Область ниже командной строки содержит текст отлаживаемой программы в дисассемблированном виде, начиная с ячейки CS:IP. Самое левое поле в каждой строке содержит смещение адреса, следующие шестнадцатеричные значения представляют объектный код каждой инструкции. Дисассемблированный текст выглядит аналогично ассемблерному листингу, за исключением того, что не используются символические имена.

Активная строка дисассемблированной области показывается на экране в обратном фоне. Инструкция в этой строке называется *текущей инструкцией*, которая выполняется следующей по команде 'G' или 'Шаг' (функциональная клавиша F1). При выполнении команды 'Шаг' выполненная инструкция сдвигается на одну строку вверх. Строка над строкой в обратном фоне называется строкой предыдущей команды.

Эта верхняя строка дисассемблированной области (строка предыдущей команды) перекрывается выводимыми сообщениями о состоянии системы и сообщениями об ошибках, когда они появляются. Поэтому на эту строку в зависимости от информации, которая в ней представлена, ссылаются так же, как на строку состояния.

Дисассемблируемая область может быть сдвинута вверх и вниз с помощью клавиш ↑ ('курсор вверх'), ↓ ('курсор вниз'), клавиш PgUp и PgDn. При сдвиге дисассемблируемой области для отображения кода с соответствующего адреса используются все 9 строк (включая строку предыдущей команды). При сдвиге дисассемблируемой области с помощью клавиш 'курсор вверх' или PgUp в область, не содержащую кода программы, может появляться сообщение:

'Нельзя дисассемблировать назад - данные'

Если это сообщение выводится при использовании режима автоповтора клавиатуры, буфер клавиатуры может быть очищен с помощью ввода Ctrl/Break.

В случае если текущая инструкция ссылается на ячейку памяти, содержимое этой ячейки показывается в обратном фоне в конце строки, расположенной после командной строки. В зависимости от типа инструкции может быть показано значение слова или байта.

Область справа от командной строки - это окно памяти номер 1, а область под дисассемблируемой областью - окно памяти номер 2. Каждое окно в первой колонке содержит адреса, а содержимое памяти выводится в шестнадцатеричном виде. Смещение от начального адреса каждой строки показано в верхней строке каждого окна.

В каждой строке окна 1 в шестнадцатеричном виде показаны восемь байтов.

Окно 2 разделено на две области, которые используют одни и те же адреса. Левая часть отображает

содержимое памяти также как и окно 1, но по 16 байтов в строке. Правая часть показывает содержимое той же области памяти, что и левая, но в коде КОИ-8 (ASCII). Адреса окон могут отличаться друг от друга.

Все символы, выведенные с повышенной яркостью, могут быть изменены пользователем вводом на их место новых значений. Курсор можно свободно передвигать в каждом окне. Для перехода из одного окна в другое нужно использовать клавиши F7-F10.

В самой нижней строке основного экрана отладчика приведено назначение функциональных клавиш. Ниже перечислены функциональные клавиши и функции, которые они вызывают при работе с основным экраном:

F1 - один шаг программы;
F2 - шаг, выполняющий процедуру;
F3 - извлечение последней команды из стека команд;
F4 - вывод экрана справочной информации;
F5 - вход в меню определения точек останова;
F6 - переключение на альтернативный экран и обратно;
F7 - курсор на поле вверх;
F8 - курсор на поле вниз;
F9 - курсор на поле влево;
F10 - курсор на поле вправо.

Все функциональные клавиши поддерживаются как клавиши с возможностью автоповтора ввода. Это означает, что, когда клавиша нажата и не отпускается, вызываемая функция будет повторяться в темпе приблизительно 10 раз в секунду.

Если в команде вызова AFD указано имя файла, то выполняется загрузка этого файла, а если есть дополнительные команды, они передаются в AFD в порядке, указанном пользователем. Эти команды, а также команда загрузки остаются в стеке сохраненных команд, что позволяет выполнить их последующий просмотр для проверки.

При загрузке AFD по умолчанию вывод сообщения об ошибке сопровождается звуковым сигналом. Он может быть выключен с помощью команды BEEP OFF.

AFD проверяет также тип процессора, на котором он выполняется. В соответствии с этим устанавливается режим дисассемблера и ассемблера, определяющий используемое подмножество мнемоники машинных инструкций. Этот режим может быть изменен с помощью команды '286'.

Если по включению питания AFD нужно резидентно установить в системе, то в качестве последней команды в файл AUTOEXEC.BAT должна быть включена команда:

AFD "QUIT R

Эта команда загружает AFD и немедленно завершает его работу, оставляя загруженным в резидентном режиме. Все другие резидентные программы должны загружаться перед AFD для того, чтобы впоследствии позволить освободить память, используемую отладчиком.

Описание основных команд отладчика

К основным функциям AFD (SDT) пользователь может обратиться с помощью функциональных клавиш и указывая в командной строке соответствующие команды полноэкранного отладчика.

Ниже приведено краткое описание всех команд AFD, которые можно вводить в командной строке. Описание получено с помощью функции "**Вывод экрана справочной информации" (F4).** Эта функция обеспечивает возможность получения справочной информации непосредственно на экране, когда выполняется ввод команды и использование документации невозможно.

Пользователю доступны четыре экрана со справочной информацией, объясняющей синтаксис и действие команд, которые можно задавать в командной строке.

Символы, введенные в командной строке основного экрана, не стираются при нажатии F4 и при возврате к основному экрану курсор устанавливается на то же место в строке, где он располагался перед вызовом справочной информации.

Команды на экране справочной информации расположены не в алфавитном порядке, а по мере их важности. Вторая часть четвертого экрана объясняет общие для различных команд правила задания параметров.

Переход от одного справочного экрана к другому выполняется нажатием клавиши пробела или цифры (1 - 4), определяющей номер экрана. Для возврата к основному экрану нужно нажать клавишу Enter или F4. Нажатие других клавиш недопустимо и вызывает звуковой сигнал.

```
                    КОМАНДЫ AFD М Е Н Ю  1
L фспец {парам.}{,адр.}  Загрузка файла в память. Задав 'адр.' Можно
                  определить место загрузки. По умолчанию CS:0100.
                  После выполнения функции в BX,CX указано число
                  загруженных байтов.
W фспец,адр.,длина  Запись данных в файл. Сегмент 'адр.' по умолчанию DS.
                  'длина' указывает число байтов - четыре 16-ные цифры.
```

{R} рег=знач. Установка регистра. FL=знач. устанавливает регистр флага как 16-ти битовый регистр. Доступ к отдельным битам по их именам: OF,DF,IF,SF,ZF,AF,PF,CF.

D адр. Вывод кода на экран. Начальный адрес дисассемблируемой области равен 'адр'. По умолчанию 'Сегмент'- CS. CS:n определяет CS последней выполненной команды.

M n адр. Вывод окна памяти (n=1 или n=2).По умолчанию 'Сегмент' тот же, что в окне. Для вывода фиксированного окна можно использовать FS. Для косвенной адресации можно указать содержимое регистра (например, [SI]).

G {старт.адр}{,адр.BRK} Запуск программы с текущей ячейки или со 'старт.адр'. Может быть определена дополнительная точка останова. По умолчанию 'сегмент' для BR - текущее значение CS. Запущенную программу можно прервать по Ctrl/Esc.

QUIT {R{ESIDENT}} Конец и возврат в ДОС. Вариант 'R' делает AFD резидентным. В этом случае AFD можно вызвать по Ctrl/Esc.

КОМАНДЫ AFD М Е Н Ю 2

A {адр.} Режим ассемблирования. Если адрес не указан, выбирается текущая команда. Ввод Enter - ассемблирование команды. Для перемещения по коду вверх и вниз можно использовать клавиши управления курсором.

P адр.,строка Замена в памяти. По умолчанию 'сегмент' - CS.

F адр.,повор,строка Заполнение памяти указанной строкой. По умолчанию сегмент адреса - DS. 'Повтор' определяет сколько раз строку нужно поместить в память.

S {{адр.},строка} Поиск данных в памяти. Если 'адр.' не указан, поиск начинается с CS:0. Сегмент по умолчанию - CS. Когда данные найдены,M2 отображает эту область, используя HS. Команда S без параметров запускает повторный поиск.

C адр.,адр.,длина Сравнение двух областей памяти. При несовпадении M1 отображает область, заданную первым параметром, M2 - область, заданную вторым ('Сегмент' по умолчанию - DS:)

CO ис.адр,пр.адр,длина Копирование данных из области, заданной 1-м параметром в область 2-го ('Сегмент' по умолчанию - DS:)

I адр. Ввод и отображение данных из порта B/B. 'Адр.'- 8 или 16 битовый адр. или содержимое регистра.

O адр.,знач. Вывод значения в порт B/B. Если 'знач.' слово, то выполняется операция с 16-ти разрядным словом

КОМАНДЫ AFD М Е Н Ю 3

T{B} Вывод на экран буфера трассировки. При пропуске 'B' вывод идет на основной экран. В противном случае выбирается отдельный формат вывода.

BW фспец Запись точек останова в файл.

BL фспец Загрузка точек останова из файла.

PH адр.,длина{,фспец} Печать данных в КОИ-8 и 16-м виде. По умолчанию: вывод на принтер, 'сегмент' - DS. 'Длина' определяет число байт.

PD адр.,длина{,фспец} Печать дисассемблированного кода. По умолчанию сегмент - CS. 'Длина' - число команд.

PT {старт,длина{,фспец}} Печать содержимого буфера трассировки. 'Старт' - смещение до 1-ой распечатываемой команды. Количество выводимых команд определяется по 'длине' или по числу действительно записанных команд. По умолчанию - все.

286 ON Включение/выключение режима для процессора 286.
OFF Первоначально установлен режим для процессора типа 86.

MO{DE} M{ONO} Установка режима экрана. 'M' - монохромный адаптор,
C{OLOR} 'C'- цветной; 'A ON' - включение возможности использо-
A{LTERN} ON вания при выполнении дополнительного экрана. F6 перекл
OFF чает основной и дополнительный экраны. Без параметров показывает текущие характеристики прикладного экрана.

BE{EP} ON Вкл/Выкл звукового сигнала
OFF

КОМАНДЫ AFD М Е Н Ю 4

XT Запуск режима обучения. Все введенные коды сохраняются в буфере и могут быть вывердены в файл или выполнены. Конец режима - заполнение буфера или по Ctrl/Break.

XX {фспец} Выполнение сохраненных кодов. Если указана 'фспец', то данные читаются из файла и затем выполняются.

XW фспец Запись сохраненных кодов в файл.

XL фспец Загрузка данных файла в буфер клавиатуры.

\- -

фспец Спецификация файла в ДОС. Для команды 'L' по умолчанию - расширение'.EXE'

адр. Задается как {сег:}смещен. Если 'сег' не указан, используются умолчания.'Смещен' м.б. любым значением. Например,DS:SI+BX-123 или * (*=адрес следующей команды)

рег Любой 8 или 16 битовый регистр с основного экрана Например, AX или BL или SI ...

знач. Содержимое регистра или байт или слово. '*' заменяется содержимым IP. Десятичные значения с % (%123) Арифметические выражения допустимы (например,AX+BX*3/2) и вычисляются строго слева направо!

строка Список значений или строк КОИ-8 (в кавычках), разделенных пробелами или запятыми, напр,1234 BX,'ASCII' FF.

Большинство команд имеют длину 1 или 2 символа и могут быть введены как строчными, так и промежуточными буквами. Команду QUIT необходимо вводить полностью, что позволяет избежать нежелательного завершения программы.

Все символы, показанные в описании команд большими буквами, должны быть введены. Некоторые команды допускают сокращение, например, команда BEEP. Символы, которые можно опускать, заключены в скобки ( ), например BE(EP). Параметр OFF можно задавать и с одним F.

Более подробное описание отладчика AFD (SDT), а также отладчика Debug приведено в [23].

3. Описание команд отладчика DEBUG

Отладчик DEBUG предназначен для решения широкого круга задач. К ним относятся, например, следующие задачи:

- Изучение текущего содержимого оперативной памяти;
- Редактирование отдельных секторов на флоппи дисках и на винчестере;
- Дизассемблирование COM и .EXE-файлов;
- Разработка и отладка собственных программ на языке ассемблера (точнее на мнемокодах);
- Изучение работы программ и их модификация;
- Тестирование периферийного оборудования, для работы с портами ввода/вывода напрямую (в диалоговом режиме);
- Изучение системы команд процессора, прерываний BIOS и MS-DOS.

При изучении схемотехники PC/AT необходим инструмент, позволяющий напрямую работать с периферийными портами и оперативной памятью, а также создавать короткие тестовые программы для генерации нужных сигналов на системной плате и периферийных адаптерах. Отладчик наилучшим образом подходит для этих целей.

В версии MS-DOS 3.3 размер отладчика составляет 15897 байтов. Существует два способа запуска отладчика: debug ENTER и debug filename ENTER. После запуска отладчик загружается в оперативную память, а содержимое сегментных регистров CS, DS, ES, SS - на первый свободный параграф сразу после самого отладчика. Регистр IP устанавливается равным 100.

Отладчик DEBUG имеет специальный указатель адреса данных, который используется по умолчанию во многих командах отладчика.

После запуска отладчика слева на экране появляется черта "–", которая указывает на то, что отладчик ждет команду. Все числа интерпретируются отладчиком в шестнадцатеричной системе исчисления.

Рассмотрим непосредственно команды отладчика DEBUG (далее символ ↵ будет обозначать нажатие клавиши возврата каретки **(Enter)**) :

1. Команда A - установка режима ассемблирования. Эта команда позволяет вводить программы с использованием мнемокода команд процессора в оперативную память. Ввод команды отладчика

A <число>↵

заставляет его перейти в режим приема команд с клавиатуры и последовательного размещения их, начиная с адреса, равному указанному в команде. При вводе можно использовать две популярные инструкции ассемблера DB и DW. Например:

```
DB      1,2,3,"EXAMPLE"
DW      1000,2000,"FFFF"
```

Отладчик поддерживает мнемоники всех команд процессора, а также и сопроцессора 80287. При ассемблировании команд JMP и CALL по умолчанию, если это возможно, используется SHORT-вариант этих команд. Но можно указывать перед адресом перехода NEAR и FAR, что приведет к генерации соответствующих команд.

Мнемоникой оператора RET, соответствующего дальнейшему вызову CALL FAR, является RETF. Возможно, а в сомнительных случаях необходимо, использовать указатели WORD PTR или BYTE PTR. При вводе программы допускается печать префиксов CS:, ES:, SS: впереди команды в той же строке.

2. Команда C - сравнение. Эта команда сравнивает побайтно две области памяти и печатает все различия между ними в форме

<адрес> <содержимое> <содержимое> <адрес>

В данной записи слева приведена информация о первой области памяти, а справа - о второй.

3. Команда D - дамп оперативной памяти. Эта команда выводит на экран монитора адреса и содержимое указанной области оперативной памяти в шестнадцатеричной системе исчисления, а справа дает их ASCII-эквиваленты. Причем, кодовые комбинации, не имеющие символьного представления в стандарте ASCII, изображаются точками.

В строке отображается шестнадцать байт. При этом слева указывается полный адрес самого левого байта. Таким образом, в одной строке приводится шестнадцатеричный дамп, а также ASCII-дамп шестнадцати байт оперативной памяти.

Если команда D дана без параметров, то всего на экране отображается 128 байт (30h) в восьми строках. В каждой строке имеется знак «-», разделяющий 16 байт пополам: между восьмым и девятым байтами.

4. Команда E - изменение содержимого байтов. Эта команда позволяет побайтно просматривать содержимое памяти вперед и назад и, в случае необходимости, изменять содержимое просматриваемых байтов.

Ввод команды:

E <адрес> ↵

вызывает переход отладчика в режим редактирования отдельных байтов. При этом печатается содержимое текущего байта в шестнадцатеричной форме, за которым следует точка. После этого можно набрать величину нового содержимого байта, но можно и оставить содержимое байта прежним, не набирая ничего.

Затем необходимо набрать один из трех управляющих символов:

а) «пробел», что означает переход к редактированию следующего байта;

б) «ENTER». Это приведет к выходу из режима побайтного редактирования на командный уровень отладчика;

в) «-». Нажатие этого знака приведет к переходу на редактирование предыдущего байта,

Пример 1.

Изменим значение счетчика системных часов. Для этого введем команду **E 40:6C** ↵ и наберем четыре числа 70 70 70 70, разделенных пробелами, затем символ возврата каретки (↵). Далее необходимо выйти из отладчика и выполнить команду **time** операционной системы.

5. Команда F - заполнение. Эта команда позволяет заполнить содержимое указанного диапазона оперативной памяти повторяющейся цепочкой байтов заданными значениями. В частности, когда цепочка состоит из одного байта, эта команда позволяет обнулить нужную область памяти, когда байт равен 00, а также занести во все биты единицу, заполняя байтом равным FF.

Команда имеет следующий синтаксис:

F <диапазон_памяти> <цепочка_байтов> ↵

6. Команда G - запуск программы. Эта команда предназначена для запуска программы на исполнение. При этом, если той области памяти, где хранятся данные, передается управление как программе, компьютер «зависает», и требуется его повторный запуск. В случае, когда в программе имеются серьезные ошибки (например, отсутствует оператор INT 20 возврата на командный уровень отладчика), попытка исполнить ее с помощью оператора G также приводит к зависанию компьютера.

Этот оператор обычно используется в одной из следующих четырех форм:

а) G ↵

Исполнение этого оператора сводится к передаче управления адресу CS:IP. Значения CS и IP можно узнать, набрав команду R(Enter) дампа всех регистров процессора;

б) G=<адрес> ↵

При этом производится запуск по указанному адресу. Например, команда **G=FFFF:0** ↵ приводит к запуску процедуры **POST**;

в) G <адрес> ↵

В этом случае программа запускается с адреса CS:IP и при достижении команды с указанным адресом осуществляется BREAK-остановка исполнения программы. Распространенной ошибкой начинающих является пропуск знака = при использовании команды G. В этом случае указанный адрес воспринимается как адрес останова, и если CS:IP указывал, например, на область данных, то компьютер «зависнет».

г) G=<адрес> <другой_адрес> ↵

В этом случае производится запуск программы с указанного после знака = адреса и в случае достижения программой команды с величиной адреса, указанного вторым (другой _адрес), происходит остановка.

Такой способ остановки программы называется введением контрольной точки останова. При указании контрольной точки останова соответствующий байт команды по этому адресу заменяется командой INT 3, имеющей размер 1 байт. При достижении контрольной точки исходное значение байта восстанавливается.

7. Команда H - шестнадцатеричная (гекс) арифметика. Эта команда позволяет получить сумму и разность двух указанных в команде шестнадцатеричных чисел.

8. Команда I - ввод из порта. Эта команда позволяет прочесть содержимое порта ввода/вывода с указанным адресом, распечатав его на экране в шестнадцатеричном виде.

Синтаксис команды:

I <адрес_порта> ↵

9. Команда L - загрузка с диска. Эта команда позволяет загружать как логические сектора с флоппи-дисков и винчестеров, так и отдельные файлы.

Загрузка секторов с диска производится командой:

L <адрес> <номер_диска> <начальный_сектор> <число_секторов> ↵

Здесь <адрес> означает начальный адрес в оперативной памяти, начиная с которого будет последовательно размещаться содержимое блоков-секторов. Переменная <номер диска> указывает с какого диска будет производиться загрузка. Число 0 означает диск A, число 1 -диск B, число 2 - диск C и т. д. Следующие две переменные соответственно указывают с какого сектора начинается чтение и общее число прочитанных секторов.

Пример 1.

Загрузить BOOT-блок с флоппи-диска A: в оперативную память, начиная с адреса DS:1000, для чего необходимо выполнить команду

L 1000 0 0 1 ↵

Затем содержимое BOOT-блока можно распечатать командой

D 1000 L200 ↵

и

U 1000 L200 ↵

Загрузка файлов в оперативную память производится двумя командами **N** и **L**. Сначала надо командой **N** указать отладчику имя необходимого файла:

N <имя_файла> ↵

Далее необходимо исполнить команду **L** без аргументов. В результате этих действий файл (за исключением файлов с расширением EXE), будет загружен в оперативную память, начиная с адреса CS:100. Если загружаемый файл имеет расширение EXE, то отладчик расшифрует PSP и в соответствии с полученной при этом информацией загрузит файл, начиная с адреса CS:0. Число прочитанных байтов хранится в регистровой паре BX:CX после выполнения команды **L**.

Пример 2.

Загрузка файла COMMAND.COM в оперативную память для его возможного изучения:

N C:\COMMAND.COM ↵

L ↵

10. Команда M - копирование содержимого части оперативной памяти в другую область оперативной памяти.

Синтаксис команды следующий:

M <диапазон> <начальный_адрес> ↵

Здесь «диапазон» указывает на копируемую область памяти, а «начальный адрес» - на адрес первого байта, начиная с которого размещаются скопированные данные. Эту команду можно использовать для временного сохранения содержимого оперативной памяти с ее последующим восстановлением.

11. Команда N - указание имени. Эта команда, прежде всего, определяет имя файла, который далее либо будет считываться с диска командой **L**, либо записываться на диск командой **W**.

Синтаксис этой команды следующий:

N <имя_файла> ↵

Пример 1.

Прочтем с диска в оперативную память файл AUTOEXEC.BAT. Для этого выполним

N C:\AUTOEXEC.BAT ↵

L ↵

В регистровую пару BX:CX, будет занесена длина загруженного файла в байтах.

12. Команда O - вывод данных в порт. Эта команда позволяет вывести указанный байт в порт с заданным адресом. Она полезна для тестирования работы периферийных устройств и периферийных БИС на системной плате в диалоговом режиме.

Синтаксис команды:

O <адрес_порта> <величина> ↵

Здесь вместо «адреса порта» подставляется адрес необходимого порта в диапазоне 0-FFFF, а вместо «величина» - значение, которое затем загружается в порт. Если порт шестнадцатибитный, то «величина» может быть четырехразрядным шестнадцатеричным числом, которое загружается в порт.

13. Команда P - высокоуровневая трассировка. Эта команда также, как и команда **T**, предназначена для трассировки .COM и .EXE-программ. Однако, в отличие от T-трассировки, эта трассировка менее детальна. Она не отслеживает досконально подпрограммы и программные прерывания, а также циклы LOOP и строковые инструкции с повторением. Но, как и в случае T-трассировки после каждой инструкции печатается содержимое всех регистров и следующей исполняемой инструкции.

Синтаксис команды:

P=<адрес> <число_инструкций> ↵

Можно опустить любой из двух параметров командной строки: «адрес» и/или «число инструкций». Параметр «адрес» задает начальный адрес, начиная с которого будет производиться трассировка, а параметр «число инструкций» будет указывать общее число инструкций, которые будут исполнены после нажатия клавиши (Enter) . Обязателен ввод символа = при указании адреса для того, чтобы не спутать адрес с числом исполняемых инструкций.

Этот тип P-трассировки крайне полезен при изучении общей логики выполнения программы, избавляя от утомительного отслеживания деталей. При начальном изучении программы необходимо начинать именно

с Р-трассировки.

14. Команда Q - выход из отладчика. Команда **Q** приводит к выходу из отладчика на следующий верхний программный уровень. При этом рабочие файлы не сохраняются.

15. Команда R - дамп/коррекция регистров. Эта команда позволяет просматривать содержимое всех регистров сразу, а также флаги, или просматривать значения отдельных регистров и регистра флагов с возможностью их изменения.

Команда

R ↵

дает распечатку всех регистров, а также команды, на которую указывает CS:IP. Команда

R <имя-регистра> ↵

дает распечатку содержимого указанного регистра, и затем печатает двоеточие на следующей строке. После этого пользователь может нажать клавишу ↵ и выйти обратно на монитор команд отладчика, либо задать новое значение регистра перед нажатием клавиши ↵. В последнем случае старое значение регистра заменится на новое.

Перечислим возможные имена регистров:

AX, BX, CX, DX, SP, BP, SI, DI, DS, ES, SS, CS, IP, F (регистр флагов).

16. Команда S - поиск упорядоченного набора байтов. Эта команда позволяет провести поиск указанной цепочки байт в заданном диапазоне оперативной памяти и имеет синтаксис:

S <диапазон памяти> <цепочка байтов>↵

Ответ выдается в виде списка адресов начиная с которых располагается указанная цепочка байтов.

Пример.

Предположим, мы хотим определить встречается ли цепочка «DOS» в первых 32К оперативной памяти. Для этого необходимо ввести команду

S 0:0 L8000 44 4F 53 ↵

17. Команда T - трассировка. Эта команда позволяет исполнить одну или несколько инструкций в режиме трассировки с печатью содержимого всех регистров и мнемоники следующей декодированной исполняемой инструкции. После трассировки текущей инструкции указатель **IP** сдвигается так, что он указывает на следующую инструкцию.

T ↵ производит трассировку одной текущей инструкции, на которую указывает CS:IP, с соответствующим изменением IP

Команда **T <число>** ↵ производит последовательную трассировку нескольких инструкций, общее число которых задастся в команде.

Команда **T =<адрес>** ↵ трассирует одну инструкцию по указанному адресу. Общий вид команды **T** следующий:

T =<адрес> <число_инструкций> ↵

18. Команда U - дизассемблирование. Эта команда позволяет дизассемблировать .COM и .EXE -программы, загруженные в рабочую область отладчика.

Общий синтаксис:

U <начальный_адрес> <длина> ↵

или

U <начальный_адрес>, <конечный_адрес> ↵

Команда

U ↵

вызывает дизассемблирование 32 байт, начиная с байта, на который указывает CS:IP с выводом результирующего листинга на дисплей.

Если при запуске отладчика переадресовать вывод в файл, то можно получить листинг дизассемблированной программы. При этом команду U и команду Q (выход из отладчика) придется набирать «вслепую» без эхопечати на дисплее.

19. Команда W - запись на диск. Эта команда позволяет записывать на диск (флоппи-диски или винчестер) в последовательные блоки указанный диапазон оперативной памяти, а также записывать в файлы.

Запись в секторы на диск производится командой

W <адрес> <номер_диска> <начальный_сектор> <число_секторов> ↵

Эта команда записывает на диск не более 80h.

Параметр «адрес» означает адрес оперативной памяти, начиная с которого содержимое памяти копируется на диск.

Параметр «номер диска» указывает накопитель, на который производится запись (0-диск A:, 1-диск B:,..., 80 - винчестер C:,).

Параметр «начальный сектор» указывает номер логического сектора, начиная с которого будет производиться загрузка образа оперативной памяти.

Параметр «число секторов» указывает общее количество записываемых на диске секторов.

Пример 1.

Предположим, необходимо заменить диагностические сообщения в BOOT-секторе на свои собственные. Для этого выполним последовательность действий:

загрузка BOOT-сектора:

L 100 80 0 l ↵

Затем, используя команду **E**, проведем редактирование диагностических сообщений (при этом ни в коем случае нельзя испортить текст программы). И, наконец, запомним отредактированный загрузчик на диске:

W 100 80 0 1 ↵

Запись в файл на диске производится командой **W** без аргументов. Однако, предварительно необходимо командой **N** указать имя файла, куда будет производиться запись, а также в пару регистров BX:CX занести длину записываемого файла. Начало области памяти, откуда будет производиться запись, всегда по умолчанию является байтом с адресом CS:0100. Если же используется команда

W <адрес> ↵

для записи в файл, то параметр <адрес> указывает начало области памяти, копируемой в файл.

Пример 2.

Скопируем в файл на диске содержимое BIOS-области оперативной памяти. Предполагая наличие PC/AT со стандартным BIOS размером 64K, обнулим сначала регистр BX, а затем занесем в регистр CX 2000H. Далее зададим имя файла, куда будем копировать старшую половину BIOS, командой

N BIOS.COM ↵

И, наконец, команда .

W F000:E000 ↵

занесет в файл BIOS.COM содержимое BIOS.

Пример 3.

Запомним загрузчик BOOT как файл BOOT.COM на диске. Для этого в регистровую пару BX:CX занесем 0000:0200. Затем исполним команды:

L 100 80 0 1 ↵

N BOOT.COM ↵

W ↵

В результате на диске появится файл BOOT.COM, исполнение которого приведет к инициализации системы.

Более подробную информацию по использованию команд отладчика DBUG можно найти в [11], [23]..

В данном разделе использованы материалы из [11].

3. Система команд микропроцессора i8086

Микропроцессор i8086 (МП ВМ86 - отечественный аналог) относится к классу однокристальных с фиксированной системой команд. При рассмотрение системы команд удобно обращаться к программной модели МП, содержащей его функциональные узлы (регистры), доступные программисту (рис. 1). Общие регистры разбиты на две группы: 1) группа HL, состоящая из регистров AX, BX, CX, DX, которые предназначены прежде всего для хранения данных и допускают раздельную адресацию их старших (H) и младших (L) половин; 2) группа PI, содержащая указательные регистры BP, SP и индексные регистры SI, DI, в которых обычно хранится адресная информация. Для общих и сегментных регистров указаны коды, используемые в форматах команд для их адресации. Регистр флагов F и указатель команд IP адресуются в командах неявно.

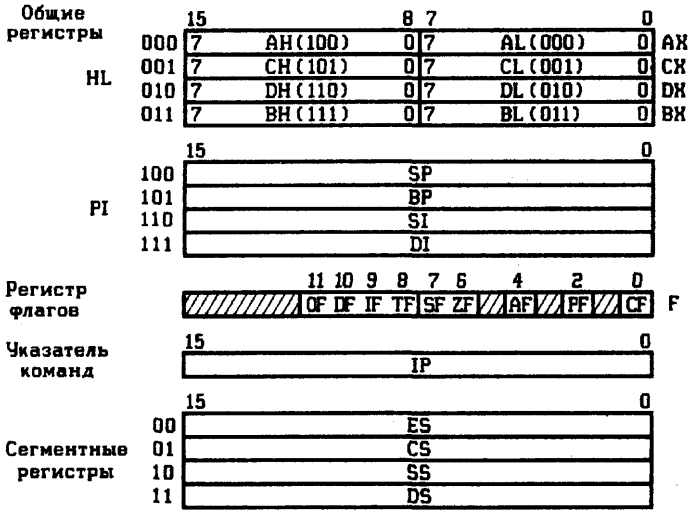

Рис. 1. Программно-доступные регистры МП ВМ86

Форматы команды

Команды МП ВМ86 могут адресовать один или два операнда, причем двухоперандные команды являются, как правило, симметричными, так как результат операции может быть направлен на место любого из операндов. Однако в таких командах один из операндов должен обязательно располагаться в регистре, поскольку имеются команды типа регистр — регистр, регистр — память и память — регистр, но команды типа память — память отсутствуют (за исключением команды пересылки цепочки байт или слов).

В общем виде формат команды приведен на рис. 2, где штриховыми линиями обозначены необязательные байты команды.

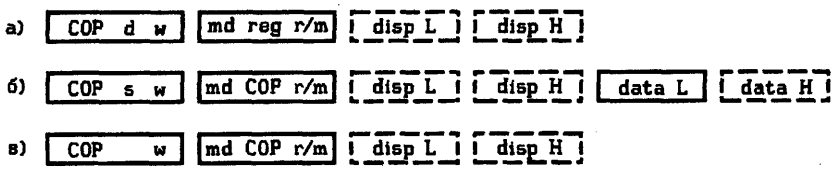

Рис. 2. Форматы команд: а и б – разновидности двухоперандных команд; в – однооперандная команда.

Первый байт команды содержит код операции COP и два однобитовых поля: направления d и слова w. При d = 1 осуществляется передача операнда или результата операции в регистр, который определяется полем reg второго байта команды; при d = 0 — передача из указанного регистра. Поле w идентифицирует тип (разрядность) операндов: при w = 1 команда оперирует словом, при w = 0 — байтом.

Второй байт, называемый постбайтом, определяет участвующие в операции регистры или регистр и ячейку памяти. Постбайт состоит из трех полей: md — режим, reg — регистр, r/m — регистр/память. Поле reg определяет операнд, который обязательно находится в регистре МП и условно считается вторым операндом. Поле r/m определяет операнд, который может находиться в регистре или памяти и условно считается первым. Поле md показывает, как интерпретируется поле r/m для нахождения первого операнда: если md = 11, то операнд содержится в регистре, в остальных случаях — в памяти.

Описание системы команд

Система команд ВМ86 содержит 91 мнемокод и позволяет совершать операции над байтами, двухбайтовыми словами, отдельными битами, а также цепочками байтов и слов. Имеется широкий набор арифметических команд, включающий умножение и деление, который ориентирован на обработку как беззнаковых, так и знаковых чисел. Весьма разнообразны команды пересылки данных, логических операций, переходов в программе и вызовов подпрограмм, а также управления микропроцессором. Число вариантов команд, т.е. число различных машинных кодов, превышает 3800 благодаря использованию восьми способов

адресации в их различных модификациях. Тем самым обеспечивается гибкость применения большого числа команд, что позволяет выбрать наиболее рациональные способы адресации в конкретных случаях. Это особенно важно при обработке сложных структур данных.

По функциональному признаку система команд BM86 разбивается на шесть групп: пересылка данных, арифметические операции, логические операции и сдвиги, передача управления, обработка цепочек и управление микропроцессором.

Полный набор команд, упорядоченный по мнемокоду, машинные коды команд и описание выполняемых операций приведены в табл. 9.

Для удобства пользования в таблице представлены все варианты мнемонических обозначений команд условных переходов и команд управления циклами. В табл. 9 введены следующие обозначения:

r — общий регистр; sr — сегментный регистр; m — адрес ячейки памяти, который указывается в мнемокоде в соответствии с используемым способом адресации; d — непосредственный операнд; ac — аккумулятор AX или AL; p — адрес 8-разрядного порта ввода — вывода; disp — смещение при адресации относительно IP; addr — указатель адреса при межсегментных переходах и вызовах; type — тип (вектор) прерывания; cond — условия в команде условных переходов; disp 8/16—смещение в формате команды, состоящее из одного или двух байтов; d 8/16 — одно- или двухбайтовая константа; sbr — имя подпрограммы; diff— разница между адресом перехода и содержимым указателя команд IP (при адресации относительно IP); label — метка, к которой осуществляется переход. Для каждой цепочечной команды приведено только общее мнемоническое обозначение, которое требует дополнительного указания разрядности элементов цепочки, например MOVSB — пересылка байтов, MOVSW — пересылка слов.

Данные табл. 9 позволяют осуществить переход от мнемокодов к машинным кодам команд. При необходимости выполнить обратный переход целесообразно воспользоваться системой команд, упорядоченной по машинному коду операции. Естественно, что наиболее удобным является автоматическое преобразование кодов с помощью соответствующих программ ассемблера и ре-ассемблера (дисассемблера).

Таблица 9

№ п/п	Команда	Байты кода команды			Символическое описание и/или содержание операции
		B1	B2	B3-B6	
1	AAA	00110111			ASCII - коррекция для сложения
2	AAD	11010101	00001010		ASCII - коррекция для деления
3	AAM	11010100	00001010		ASCII - коррекция для умножения
4	AAS	00111111			ASCII - коррекция для вычитания
5	ADC r,r/m	0001001W	md reg r/m	(disp8/16)	r←r+r/m+CF; сложение с переносом
	r/m,r	0001000W	md reg r/m	(disp8/16)	r/m←r+r/m+CF
	r/m,d	1000000W	md 010 r/m	(disp8/16)d8/16	r/m←r/m+d+CF
	r16/m16,d8	10000011	md 010 r/m	data L	r/m←r/m+d+CF
	ac,d	0001010W	data L	(data H)	ac←ac+d+CF
6	ADD r,r/m	0000001W	md reg r/m	(disp8/16)	r←r+r/m; сложение
	r/m,r	0000000W	md reg r/m	(disp8/16)	r/m←r+r/m
	r/m,d	1000000W	md 000 r/m	(disp8/16)d8/16	r/m←r/m+d
	r16/m16,d8	10000011	md 000 r/m	data L	r/m←r/m+d
	ac,d	0000010W	data L	(data H)	ac←ac+d
7	AND r,r/m	0010001W	md reg r/m	(disp8/16)	r←r∧r/m; конъюнкция, И
	r/m,r	0010000W	md reg r/m	(disp8/16)	r/m←r∧r/m
	r/m,d	1000000W	md 100 r/m	data L data H	r/m←r/m∧d
	ac,d	0010010W	data L	(data H)	ac←ac∧d
8	CALL				
	NEAR sbr	11101000	diff L	diff H	IP←IP+diff; вызов прямой
	NEAR r16/m16	11111111	md 010 r/m	(disp8/16)	IP←IP+r/m; вызов косвенный
	FAR sbr	10011010	IP-L	IP-H CS-L CS-H	IP←IP-L,IP-H; CS←CS-L,CS-H; вызов прямой
	FAR m32	11111111	md 011 r/m	(disp8/16)	IP←m16;CS←m16; вызов

Таблица 9 (продолжение)

№ п/п	Команда	Байты кода команды			Символическое описание и/или содержание операции
		B1	B2	B3-B6	
					косвенный
9	CBW	10011000			Преобразование байта в слово
10	CLC	11111000			CF←0; сброс переноса
11	CLD	11111100			DF←0; сброс триггера направления
12	CLI	11111010			IF←0; запрет прерывания
13	CMC	11110101			CF←CF; инверсия переноса
14	CMP r,r/m	0011101W	md reg r/m	(disp8/16)	r-r/m; сравнение
	r/m,r	0011100W	md reg r/m	(disp8/16)	r/m-r
	r/m,d	1000000W	md 111 r/m	(disp8/16)d8/16	r/m-d
	r16/m16,d8	10000011	md 111 r/m	data L	r/m-d
	ac,d	0011110W	data L	(data H)	ac-d
15	CMPS	1010011W			Сравнение элементов цепочек
16	CWD	10011001			Преобразование слова в двойное слово
17	DAA	00100111			Десятичная коррекция для сложения
18	DAS	00101111			Десятичная коррекция для вычитания
19	DEC r/m	1111111W	md 001 r/m	(disp8/16)	r/m←r/m-1; декремент
	r	01001reg			
20	DIV r/m	1111011W	md 110 r/m	(disp8/16)	Деление чисел без знака
21	ESC OPCODE,r/m	11011XXX	md YYY r/m	(disp8/16)	Переключение на сопроцессор
22	HLT	11110100			Останов
23	IDIV r/m	1111011W	md 111 r/m	(disp8/16)	Деление чисел со знаком
24	IMUL r/m	1111011W	md 101 r/m	(disp8/16)	Умножение чисел со знаком
25	IN ac,port	1110010W	port8		Ввод из фиксированного порта
	ac,DX	1110110W			Ввод из порта с косвенной адресацией
26	INC r/m	1111111W	md 000 r/m	(disp8/16)	r/m←r/m+1; инкремент
	r	01000reg			r←r+1;
27	INT type	11001101			Прерывание заданного типа
	INTO	11001110			Прерывание по переполнению
	INT3	11001100			Прерывание типа 3
28	IRET	11001111			Возврат из прерывания
29	JA label	01110111	diff L		IP←IP+diff L; перейти, если выше
30	JAE label	01110011	diff L		IP←IP+diff L; если выше или равно
31	JB label	01110010	diff L		IP←IP+diff L; если ниже
32	JBE label	01110110	diff L		IP←IP+diff L; если ниже или равно
33	JC label	01110010	diff L		IP←IP+diff L; если есть перенос
34	JCXZ label	11100011	diff L		IP←IP+diff L; если CX равен нулю
35	JE label	01110100	diff L		IP←IP+diff L; если равно
36	JG label	01111111	diff L		IP←IP+diff L; если больше
37	JGE label	01111101	diff L		IP←IP+diff L; если больше или равно
38	JL label	01111100	diff L		IP←IP+diff L; если меньше
39	JLE label	01111110	diff L		IP←IP+diff L; если меньше или равно

Таблица 9 (продолжение)

№ п/п	Команда	Байты кода команды			Символическое описание и/или содержание операции
		B1	B2	B3-B6	
40	JMP SHORT label	11101011	diff L		IP←IP+diff L; прямой короткий переход
	NEAR label	11101001	diff L	diff H	IP←IP+diff L; прямой переход
	NEAR r16/m16	11111111	md 100 r/m	(disp8/16)	IP←IP+r/m; косвенный переход
	FAR label	11101010	IP-L	IP-H CS-L CS-H	IP←IP-L,IP-H; CS←CS-L,CS-H; прямой переход
	FAR m32	11111111	md 101 r/m	(disp8/16)	IP←m16;CS←m16; косвенный переход
41	JNA label	01110110	diff L		IP←IP+diff L; если не выше
42	JNAE label	01110011	diff L		IP←IP+diff L; если не выше или равно
43	JNB label	01110111	diff L		IP←IP+diff L; если не ниже
44	JNC label	01110011	diff L		IP←IP+diff L; если нет переноса
45	JNE label	01110101	diff L		IP←IP+diff L; если не равно
46	JNG label	01111110	diff L		IP←IP+diff L; если не больше
47	JNGE label	01111100	diff L		IP←IP+diff L; если не больше или равно
48	JNL label	01111101	diff L		IP←IP+diff L; если не меньше
49	JNLE label	01111111	diff L		IP←IP+diff L; если не меньше или равно
50	JNO label	01110001	diff L		IP←IP+diff L; если нет переполнения
51	JNP label	01111011	diff L		IP←IP+diff L; если нет паритета
52	JNS label	01111001	diff L		IP←IP+diff L; если результат положительный
53	JNZ label	01110101	diff L		IP←IP+diff L; если не нуль
54	JO label	01110000	diff L		IP←IP+diff L; если есть переполнение
55	JP label	01111010	diff L		IP←IP+diff L; если есть паритет
56	JPE label	01111010	diff L		IP←IP+diff L; если паритет четный
57	JPO label	01111011	diff L		IP←IP+diff L; если паритет нечетный
58	JS label	01111000	diff L		IP←IP+diff L; если результат отрицательный
59	JZ label	01110100	diff L		IP←IP+diff L; если нуль
60	LAHF	10011111			AH←FL; пересылка младшего байта F в AH
61	LDS r16,m32	11000101	md reg r/m	(disp8/16)	r,DS←m32; загрузка указателя адреса
62	LEA r16,m	10001101	md reg r/m	(disp8/16)	r←EA; загрузка эффективного адреса
63	LES r16,m32	11000100	md reg r/m	(disp8/16)	r,ES←m32; загрузка указателя адреса
64	LODS	1010110W			Загрузка элемента цепочки
65	LOOP label	11100010	diff L		IP←IP+diff L; зациклить
66	LOOPE/LOOPZ label	11100001	diff L		IP←IP+diff L; зациклить, если равно
67	LOOPNE/LOOPNZ label	11100000	diff L		IP←IP+diff L; зациклить, если не равно

Таблица 9 (продолжение)

№ п/п	Команда	Байты кода команды			Символическое описание и/или содержание операции
		B1	B2	B3-B6	
68	MOV r,r/m	1000101W	md reg r/m	(disp8/16)	r←r/m; пересылка
	r/m,r	1000100W	md reg r/m	(disp8/16)	r/m←r;
	r/m,d	1100011W	md 000 r/m	(disp8/16)d8/16	r/m←d;
	r,d	1011Wreg	data L	(data H)	r←d
	ac,m	1010000W	disp L	disp H	ac←m; при прямой адресации
	m,ac	1010001W	disp L	disp H	m←ac; при прямой адресации
	sr,r/m	10001110	md 0sr r/m	(disp8/16)	sr←r/m; запись в сегментный регистр
	r/m,sr	10001100	md 0sr r/m	(disp8/16)	r/m←sr; чтение из сегментного регистра
69	MOVS	1010010W			Пересылка элемента цепочки
70	MUL r/m	1111011W	md 100 r/m	(disp8/16)	Умножение чисел без знака
71	NEG r/m	1111011W	md 011 r/m	(disp8/16)	Смена знака числа
72	NOP	10010000			Пустая операция
73	OR r,r/m	0000101W	md reg r/m	(disp8/16)	r←rvr/m; дизъюнкция, ИЛИ
	r/m,r	0000100W	md reg r/m	(disp8/16)	r/m←r/mvr
	r/m,d	1000000W	md 001 r/m	(disp8/16)d8/16	r/m←r/mvd
	ac,d	0000110W	data L	(data H)	ac←acvd
74	OUT ac,port	1110011W	port8		Вывод в фиксированный порт
	ac,DX	1110111W			Вывод в порт при косвенной адресации
75	POP r/m	10001111	md 000 r/m	(disp8/16)	Чтение из стека
	r	01011reg			
	sr	000sr111			
76	POPF	10011101			Загрузка регистра флагов из стека
77	PUSH r/m	11111111	md 110 r/m	(disp8/16)	Запись в стек
	r	01010reg			
	sr	000sr110			
78	PUSHF	10011100			Запись регистра флагов в стек
79	RCL r/m,1	1101000W	md 010 r/m	(disp8/16)	Циклический сдвиг
	r/m,CL	1101001W	md 010 r/m	(disp8/16)	влево через CF
80	RCR r/m,1	1101000W	md 011 r/m	(disp8/16)	Циклический сдвиг
	r/m,CL	1101001W	md 011 r/m	(disp8/16)	вправо через CF
81	RET (NEAR)	11000011			Возврат из подпрограммы
	d(NEAR)	11000010	data L	data H	Возврат с прибавлением константы к SP
	(FAR)	11001011			Возврат из подпрограммы
	d(FAR)	11001010	data L	data H	Возврат с прибавлением константы к SP
82	ROL r/m,1	1101000W	md 000 r/m	(disp8/16)	Циклический сдвиг
	r/m,CL	1101001W	md 000 r/m	(disp8/16)	влево
83	ROR r/m,1	1101000W	md 001 r/m	(disp8/16)	Циклический сдвиг
	r/m,CL	1101001W	md 001 r/m	(disp8/16)	вправо
84	SAHF	10011110			Пересылка AH в регистр F
85	SAL r/m,1	1101000W	md 100 r/m	(disp8/16)	Арифметический сдвиг
	r/m,CL	1101001W	md 100 r/m	(disp8/16)	влево

86	SAR r/m,1	1101000W md 111 r/m (disp8/16)		Арифметический сдвиг
	r/m,CL	1101001W md 111 r/m (disp8/16)		вправо
87	SBB r,r/m	0001101W md reg r/m (disp8/16)		r←r-r/m-CF; вычитание с заемом
	r/m,r	0001100W md reg r/m (disp8/16)		r/m←r-r/m-CF
	r/m,d	1000000W md 011 r/m (disp8/16)d8/16		r/m←r/m-d-CF
	r16/m16,d8	10000011 md 011 r/m data L		r/m←r/m-d-CF
	ac,d	0001110W data L	(data H)	ac←ac-d-CF
88	SCAS	1010111W		Сканировать элемент цепочки

Таблица 9 (продолжение)

№ п/п	Команда	Байты кода команды			Символическое описание и/или содержание операции
		B1	B2	B3-B6	
89	SHL r/m,1	1101000W md 100 r/m (disp8/16)			Логический сдвиг
	r/m,CL	1101001W md 100 r/m (disp8/16)			влево
90	SHR r/m,1	1101000W md 101 r/m (disp8/16)			Логический сдвиг
	r/m,CL	1101001W md 101 r/m (disp8/16)			вправо
91	STC	11111001			CF←1; установка флага переноса
92	STD	11111101			DF←1; установка триггера направления
93	STI	11111011			IF←1; разрешение прерывания
94	STOS	1010101W			[DI]←ac; запоминание ac в цепочке
95	SUB r,r/m	0010101W md reg r/m (disp8/16)			r←r-r/m; вычитание
	r/m,r	0010100W md reg r/m (disp8/16)			r/m←r-r/m
	r/m,d	1000000W md 101 r/m (disp8/16)d8/16			r/m←r/m-d
	r16/m16,d8	10000011 md 101 r/m data L			r/m←r/m-d
	ac,d	0010110W data L	(data H)		ac←ac-d;
96	TEST r,r/m	1000010W md reg r/m (disp8/16)			r∧r/m; проверка, неразрушающее И
	m,r	1000010W md reg r/m (disp8/16)			r∧r/m
	r/m,d	1111011W md 000 r/m (disp8/16)d8/16			r/m d
	ac,d	1010100W data L	(data H)		ac∧d
97	WAIT	00111011			Ожидание
98	XCHG AX,r	10010reg			AX↔r; обмен
	r,AX	10010reg			r↔AX
	r,r/m	1000011W md reg r/m (disp8/16)			r↔r/m
	m,r	1000011W md reg r/m (disp8/16)			m↔r
99	XLAT	11010111			Преобразование байта из AL
100	XOR r,r/m	0011001W md reg r/m (disp8/16)			r←r⊕r/m; суммирование по модулю 2 (исключающее ИЛИ)
	r/m,r	0011000W md reg r/m (disp8/16)			r/m←r⊕r/m;
	r/m,d	1000000W md 110 r/m (disp8/16)d8/16			r/m←r/m⊕d
	ac,d	0011010W data L	(data H)		ac←ac⊕d
	Префиксы:				
	LOCK	11110000			Префикс блокировки шины
	REP/REPE/REPZ	11110011			Повторять цепочечную операцию
	REPNE/REPNZ	11110010			То же
	SEG	001sr110			Префикс замены сегмента

Более подробное описание команд процессоров i8086 и i80286 можно найти в [20 - 22].

В данном разделе использованы материалы из [21].

Список литературы

1. Рудометов Е., Рудометов В. Материнские платы и чипсеты. 3-е изд. - СПб: Питер, 2002 - 352 с.: ил. - (Серия "Анатомия ПК).
2. Павлов В.А. Чипсеты, материнские платы и BIOS. Учебно-методические материалы. СарФТИ, Саров. 2002. -189 с.: ил.
3. Хаммел Р. Л. Последовательная передача данных: Руководство для программиста: Пер. с англ. - М.: Мир, 1996. - 752 с., ил.
4. Гук М. Аппаратные интерфейсы ПК. Энциклопедия. -СПб.: Питер, 2002. -528 с.; ил.
5. Гук М. Аппаратные средства IBM PC. Энциклопедия, 2-е изд. -СПб.: Питер, 2001. - 928 с.: ил.
6. Кулаков В. Программирование на аппаратном уровне: Специальный справочник. 2-е изд. СПб.: Питер, 2003. - 848 с.: ил.
7. Ан П. Сопряжение ПК с внешними устройствами / Пей Ан; Пер. с англ. Мерещука П.В. - 2-е изд., стер. - М.: ДМК Пресс; СПб,: Питер, 2004. - 320с.: ил.
8. Вегнер В.А. и др. Аппаратура персональных компьютеров и ее программирование. IBM PC/XT/AT и PS/2. М: Радио и связь, 1995. 224с. - (Библиотека системного программиста).
9. Фролов А.В., Фролов Г.В. Аппаратное обеспечение персонального компьютера. -М.: ДИАЛОГ-МИФИ, 1997. - 304 с. - (Библиотека системного программиста; Т. 33).
10. Новиков Ю.В., Калашников О.А., Гуляев С.Э. Разработка устройств сопряжения для персонального компьютера типа IBM PC. Под общей редакцией Ю.В. Новикова. Практ. пособие - М.: ЭКОМ., 1997 - 224 с.: ил.
11. Левкин Г. Н., Левкина В. Е. Введение в схемотехнику ПЭВМ PC/AT. - М.: Изд-во МПИ, 1991. - 96 с.: ил.
12. Агуров П.В. Последовательные интерфейсы ПК. Практика программирования. - СПб.: БХВ-Петербург, 2004. - 496 с.: ил.
13. Несвижский Всеволод. Программирование аппаратных средств в Windows. - СПб.: БХВ-Петербург, 2004. - 880 с.: ил.
14. Джордейн Р. Справочник программиста персональных компьютеров типа IBM PC, XT и AT: Пер. с англ./ Предисл. Н.В. Гайского. -М.: Финансы и статистика, 1992. - 544 с.: ил.
15. Зубков С.В. Assembler для DOS, Windows и UNIX / Зубков Сергей Владимирович. - 3-е изд., стер. - М.: ДМК Пресс; СПб.: Питер, 2004 - 608 с.: ил. - (Серия "Для программистов").
16. PC 99 System Design Guide. - Intel Corporation and Microsoft Corporation.
17. Гук М. Аппаратные средства IBM PC. Энциклопедия, 3-е изд. -СПб.: Питер, 2006. - 1072 с.: ил.
18. Гук М. Аппаратные средства IBM PC. Энциклопедия - СПб.: Питер Ком, 1998. - 816 с.: ил.
19. Гук М. Интерфейсы ПК: справочник - СПб: ЗАО "Издательство Питер", 1999. - 416с.: ил.
20. Лю Ю-Чжен, Гибсон Г. Микропроцессоры семейства 8086/8088. Архитектура, программирование и проектирование микрокомпьютерных систем /Пер. с англ.- М.: Радио и связь, 1987. - 512 с.: ил.
21. Микропроцессорный комплект К1810: Структура, программирование, применение: Справочная книга / Ю. М. Казаринов, В. Н. Номоконов, Г. С. Подклктнова, В. Ф. Филиппов; Под ред. Ю. М. Казаринова. - М.: Высш. шк., 1990. - 296 с.: ил.
22. Пильщиков В.Н. Программирование на языке ассемблера IBM PC. - М.: "ДИАЛОГ-МИФИ", 1994. -288 с.
23. Павлов В. А. Система ввода-вывода ПК. Параллельный порт. Учебно-методическое пособие и практикум по циклу "Система ввода/вывода ПК". СарФТИ, Саров, 2005. - 204 с.: ил.