САРОВСКИЙ ГОСУДАРСТВЕННЫЙ
ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

В.А. Павлов

Видеоадаптеры ПК.

Учебное пособие

Саров 2004г.

В.А. Павлов

Видеоадаптеры ПК

Учебное пособие для вузов

Рекомендовано
учебно-методическим объединением вузов
по университетскому политехническому образованию
в качестве учебного пособия
для студентов, обучающихся по специальности 220100 -
"Вычислительные машины, комплексы, системы и сети"

Саров
СарФТИ
2003

УДК 681.3.06
П12

Рецензенты:
д-р техн. наук, профессор кафедры МО ЭВМ Нижегородского государственного университета Ю.Л. Кетков;
к-т техн. наук, доцент кафедры "Компьютерные системы и сети" МГТУ им. Н.Э. Баумана И.В. Баскаков;
кафедра "Вычислительная техника" Нижегородского государственного технического университета

Павлов Виктор Александрович
П12 Видеоадаптеры ПК. Учебное пособие. Саров, СарФТИ, 2004. 291с.: ил.

В учебном пособии рассмотрены общие принципы работы видеосистемы ПК и её эволюция от монохромной текстовой системы MDA до видеосистемы современного мультимедийного компьютера. Основное внимание уделено основной составляющей видеосистемы - видеоадаптеру. Содержание учебного пособия является информационной поддержкой курса "Адаптеры и контроллеры ЭВМ", читаемого автором в Саровском физико-техническом институте (СарФТИ). Пособие предназначено для студентов технических вузов, обучающихся по специальности "Вычислительные машины, комплексы, системы и сети", а также родственным ей специальностям.

©В.А. Павлов, 2003

СОДЕРЖАНИЕ

Стр.

ВВЕДЕНИЕ -- -- -- -- -- -- -- -- -- 10

Контрольные вопросы -- -- -- -- -- -- -- 11

1. Структура видеосистемы PC -- -- -- -- -- -- 12

1.1. Архитектура PC -- -- -- -- -- -- -- -- 12

1.2. Принцип действия видеосистемы PC -- -- -- -- -- 15

1.2.1. Формирование изображения на экране монитора -- -- -- -- 15

1.2.1.1. Конструкция ЭЛТ -- -- -- -- -- -- -- -- 15

1.2.1.2. Формирование растра -- -- -- -- -- -- -- 15

1.2.1.3. Разрешение изображения -- -- -- -- -- -- -- 17

1.2.1.4. Количество отображаемых цветов -- -- -- -- -- -- 19

1.2.2. Формирование видеосигнала-- -- -- -- -- -- -- 20

1.2.2.1. Графический режим -- -- -- -- -- -- -- -- 21

1.2.2.2. Текстовый режим -- -- -- -- -- -- -- -- 21

1.3. Эволюция видеосистемы PC -- -- -- -- -- -- 26

1.3.1. MDA-- -- -- -- -- -- -- -- -- -- 26

1.3.2. CGA -- -- -- -- -- -- -- -- -- 30

1.3.3. EGA -- -- -- -- -- -- -- -- -- 27

1.3.3.1. Структурная схема адаптера EGA -- -- -- -- -- -- 31

1.3.3.2. Формирование изображения -- -- -- -- -- -- -- 33

1.3.3.3. Регистры видеоадаптера EGA. -- -- -- -- -- -- 34

1.3.3.3.1. Общие регистры. -- -- -- -- -- -- -- -- 34

1.3.3.3.2. Регистры контроллера синхронизации (генератора последовательности) -- 35

1.3.3.3.3. Регистры видеоконтроллера (контроллера электронно-лучевой трубки). -- 36

1.3.3.3.4. Регистры контроллера атрибутов -- -- -- -- -- -- 39

1.3.3.3.5. Регистры графического контроллера -- -- -- -- -- 41

1.3.4. VGA -- -- -- -- -- -- -- -- -- -- 43

1.3.5. Super VGA -- -- -- -- -- -- -- -- -- 46

1.3.6. Видеосистема мультимедийного PC -- -- -- -- -- -- 48

Контрольные вопросы -- -- -- -- -- -- -- 53

2. Видеоадаптер мультимедийного PC -- -- -- -- -- 55

2.1. Устройство и особенности работы стандартного видеоадаптера VGA -- 55

2.1.1. Видеопамять-- -- -- -- -- -- -- -- -- 57

2.1.1.1. Логическая структура адресного пространства PC -- -- -- -- 57

2.1.1.1.1. Линейная адресация данных в реальном режиме DOS -- -- -- 58

2.1.1.1.2. Подпрограммы установки режима линейной адресации -- -- -- 61

Листинг 2.1. Подпрограмма, устанавливающая режим линейной адресации данных **(list2_01.inc)** -- -- -- -- -- -- 62

2.1.1.1.3. Использование области верхней памяти видеосистемой PC -- -- -- 63

2.1.1.2. Организация видеопамяти -- -- -- -- -- -- -- 64

2.1.1.2.1. Формат PDF -- -- -- -- -- -- -- -- 64

2.1.1.2.2. Метод цветовых плоскостей -- -- -- -- -- -- 65

4.1.1.3. Разрядность видеопамяти -- -- -- -- -- -- -- 65

2.1.2. Video BIOS -- -- -- -- -- -- -- -- -- 66

2.1.2.1. Видеосервис BIOS -- -- -- -- -- -- -- -- 66

2.1.2.1.1. Аппаратные прерывания -- -- -- -- -- -- -- 66

2.1.2.1.2. Программные прерывания -- -- -- -- -- -- -- 66

2.1.2.2. Функции Video BIOS -- -- -- -- -- -- -- 67

2.1.3. Контроллер ЭЛТ -- -- -- -- -- -- -- -- 76

2.1.3.1. Регистры контроллера ЭЛТ -- -- -- -- -- -- -- 76

2.1.3.2. Программная настройка растра -- -- -- -- -- -- 82

2.1.4. Графический контроллер -- -- -- -- -- -- -- 84

2.1.4.1. Регистры графического контроллера -- -- -- -- -- -- 84

2.1.4.2. Двумерная адресация -- -- -- -- -- -- -- 87

2.1.4.3. Двумерное маскирование -- -- -- -- -- -- -- 88

2.1.5. Секвенсер -- -- -- -- -- -- -- -- -- 89

2.1.5.1. Регистры секвенсора-- -- -- -- -- -- -- -- 89

2.1.6. Контроллер атрибутов -- -- -- -- -- -- -- 91
2.1.6.1. Регистры контроллера атрибутов -- -- -- -- -- -- 91
2.1.7. RAMDAC -- -- -- -- -- -- -- -- -- 93
2.1.7.1. Регистры цифро-аналогового преобразователя -- -- -- -- 94
2.1.8. Синхронизатор -- -- -- -- -- -- -- -- 96
2.1.9. Тактовые генераторы -- -- -- -- -- -- -- 96
2.1.10. Интерфейс с шиной ввода/вывода -- -- -- -- -- -- 97
2.1.11. Общие регистры -- -- -- -- -- -- -- -- 97
2.1.12. Кодирование цвета пиксела в различных видеорежимах -- -- -- 98
2.1.12.1. 16-цветные режимы -- -- -- -- -- -- -- 99
2.1.12.2. 256-цветный режим (13h) -- -- -- -- -- -- -- 101
2.1.12.3. Монохромные режимы работы -- -- -- -- -- -- 101
2.1.12.4. Монохромные (двухцветные) режимы работы -- -- -- -- 102
2.1.13. Недостатки видеоадаптера VGA -- -- -- -- -- -- 102
Контрольные вопросы. -- -- -- -- -- -- -- 103
2.2. Основные направления совершенствования видеоадаптера VGA. Видеоадаптеры SVGA -- -- -- -- -- -- -- -- 105
2.2.1. Аппаратное ускорение графических функций -- -- -- -- 106
2.2.1.1. Основные графические функции -- -- -- -- -- -- 107
2.2.1.2. Видеоадаптер с графическим акселератором -- -- -- -- -- 109
2.2.1.3. Видеоадаптер с графическим сопроцессором -- -- -- -- -- 110
2.2.2. Память для видеоадаптеров -- -- -- -- -- -- -- 111
2.2.2.1. Однопортовая видеопамять -- -- -- -- -- -- -- 112
2.2.2.1.1. FPM DRAM -- -- -- -- -- -- -- -- 112
2.2.2.1.2. EDO DRAM -- -- -- -- -- -- -- -- 113
2.2.2.1.3. BEDO DRAM -- -- -- -- -- -- -- -- 113
2.2.2.1.4. SDRAM -- -- -- -- -- -- -- -- -- 113
2.2.2.1.5. SGRAM -- -- -- -- -- -- -- -- -- 113
2.2.2.1.6. MDRAM -- -- -- -- -- -- -- -- -- 114
2.2.2.2. Двухпортовая видеопамять -- -- -- -- -- -- -- 114
2.2.2.2.1. VRAM -- -- -- -- -- -- -- -- -- 114
2.2.2.2.2. WRAM -- -- -- -- -- -- -- -- -- 114
2.2.2.3. Перспективные типы видеопамяти -- -- -- -- -- -- 114
2.2.2.3.1. Direct RDRAM -- -- -- -- -- -- -- -- 114
2.2.2.3.2. 3D RAM -- -- -- -- -- -- -- -- -- 115
2.2.2.3.3. CDRAM -- -- -- -- -- -- -- -- -- 115
2.2.2.3.4. DDR SDRAM -- -- -- -- -- -- -- -- 115
2.2.2.3.5. ESDRAM -- -- -- -- -- -- -- -- -- 115
2.2.3. Интерфейс между видеоадаптером и материнской платой -- -- -- 115
2.2.3.1. Шина ISA -- -- -- -- -- -- -- -- -- 116
2.2.3.2. Локальная шина VLB -- -- -- -- -- -- -- 116
2.2.3.3. Шина PC1 -- -- -- -- -- -- -- -- -- 117
2.2.4. Унификация возможностей видеоадаптеров SVGA. Спецификация VESA -- 117
2.2.4.1. VESA-расширение B1OS -- -- -- -- -- -- -- 119
2.2.4.1.1. Информационные функции VBE -- -- -- -- -- -- 120
2.2.4.1.1.1. Прерывание Int 10h, функция 4Fh, подфункция 00h: ПОЛУЧИТЬ ИНФОРМАЦИЮ О ВЕРСИИ VESA BIOS -- -- -- -- -- 120
2.2.4.1.1.2. Прерывание Int 10h, функция 4Fh, подфункция 01h: ПОЛУЧИТЬ ИНФОРМАЦИЮ О ПАРАМЕТРАХ ВИДЕОРЕЖИМА -- -- -- 120
2.2.4.1.2. Управляющие функции VBE -- -- -- -- -- -- 123
2.2.4.1.2.1. Прерывание Int 10h, функция 4Fh, подфункция 02h: УСТАНОВИТЬ ВИДЕОРЕЖИМ С ЗАДАННЫМ НОМЕРОМ -- -- -- -- -- 123
2.2.4.1.2.2. Прерывание Int 10h, функция 4Fh, подфункция 03h: ОПРЕДЕЛИТЬ КОД ТЕКУЩЕГО ВИДЕОРЕЖИМА -- -- -- -- -- -- 123
2.2.4.1.2.3. Прерывание Int 10h, функция 4Fh, подфункция 04h: СОХРАНИТЬ ИЛИ ВОССТАНОВИТЬ СОСТОЯНИЕ ВИДЕОКОНТРОЛЛЕРА -- -- -- 123
2.2.4.1.2.4. Прерывание Int 10h, функция 4Fh, подфункция 05h: УПРАВЛЕНИЕ ОКНАМИ ВИДЕОПАМЯТИ -- -- -- -- -- -- 124
2.2.4.1.2.5. Прерывание Int 10h, функция 4Fh, подфункция 06h: ПОЛУЧИТЬ ИЛИ УСТАНОВИТЬ ДЛИНУ ЛОГИЧЕСКОЙ СТРОКИ РАЗВЕРТКИ -- -- 124

2.2.4.1.2.6. Прерывание Int 10h, функция 4Fh, подфункция 07h: ПОЛУЧИТЬ ИЛИ УСТАНОВИТЬ КООРДИНАТЫ ЛЕВОГО ВЕРХНЕГО УГЛА ЭКРАНА-- -- 125
2.2.4.1.2.7. Прерывание Int 10h, функция 4Fh, подфункция 08h: ПОЛУЧИТЬ ИЛИ ИЗМЕНИТЬ ФОРМАТ РЕГИСТРОВ ПАЛИТРЫ -- -- -- -- 126
2.2.4.1.2.8. Прерывание Int 10h, функция 4Fh, подфункция 09h: СОХРАНИТЬ ИЛИ ИЗМЕНИТЬ СОДЕРЖИМОЕ РЕГИСТРОВ ЦАП -- -- -- -- 126
2.2.4.1.2.9. Прерывание Int 10h, функция 4Fh, подфункция 0Ah: ПОЛУЧИТЬ ТАБЛИЦУ ДОСТУПА К ИНТЕРФЕЙСУ ЗАЩИЩЕННОГО РЕЖИМА -- 126
2.2.4.1.3. Дополнительные функции VBE -- -- -- -- -- -- 127
Контрольные вопросы -- -- -- -- -- -- -- 127
2.2.5. Пример использования функций VESA BIOS. -- -- -- -- 130
Листинг 2.2. Проверка наличия VESA BIOS -- -- -- -- -- 130
2.2.5.1.Процедуры вывода данных на экран и ввода информации с клавиатуры -- -- 132
Листинг 2.3. Процедуры ввода-вывода общего назначения **(list1_02.inc)** -- 133
2.2.5.2. Мнемоническое обозначение управляющих клавиш и цветовых кодов -- -- 137
Листинг 2.4. Мнемонические обозначения кодов управляющих клавиш и цветовых оттенков **(list1_03.inc)** -- -- -- -- -- -- 138
2.2.5.3. Макрокоманды для вывода данных на экран -- -- -- -- -- 139
Листинг 2.5. Макрокоманды для вывода данных на экран **(list1_04.inc).** -- 139
2.2.5.4. Перевод чисел из десятичного кода в двоичный и наоборот -- -- -- 140
Листинг 2.6. Набор подпрограмм, предназначенных для перевода целых и вещественных чисел из двоичной системы счисления в десятичную и наоборот **(list2_05.inc)** -- -- -- -- -- -- 142
2.2.5.5. Программа тестирования функций перевода целых и вещественных чисел. -- 149
Листинг 2.7. Тестирование подпрограмм перевода вещественных чисел -- 149
Контрольные вопросы -- -- -- -- -- -- -- 150
2.2.6. Особенности работы с видеоадаптером в различных режимах -- -- 151
2.2.6.1. Особенности работы в текстовом режиме -- -- -- -- -- 151
2.2.6.2. Работа в современных графических режимах -- -- -- -- -- 152
2.2.6.2.1. Организация видеопамяти в 256-цветных режимах -- -- -- -- 152
2.2.6.2.2. Организация видеопамяти в режимах типа DirectDraw -- -- -- 152
2.2.6.2.3. Режимы адресации и распределение видеопамяти -- -- -- -- 154
2.2.6.2.4. Рисование линий при линейной адресации памяти -- -- -- -- 154
2.2.6.2.5. Вывод текста и статических изображений в графических режимах -- -- 156
Листинг 2.8. Процедуры общего назначения, предназначенные для установки графических режимов и работы в них **(list4_02.inc)** -- -- -- -- 158
Листинг 2.9. Процедуры вывода символа и очистки экрана для 256-цветных режимов с линейной адресацией видеобуфера **(list4_03.inc)** -- -- -- 161
Листинг 2.10. Вывод на экран текста и линий в режиме 256 цветов с разрешением 640×480 -- -- -- -- -- -- -- 162
Листинг 2.11. Подпрограмма рисования линии по алгоритму Брезенхема для 256-цветных режимов **(list4_09.inc)** -- -- -- -- -- 163
Листинг 2.12. Рисование линий по алгоритму Брезенхема в режиме 256 цветов с разрешением 640×480 -- -- -- -- -- -- -- 165
2.2.6.2.6. Масштабирование изображений -- -- -- -- -- -- 166
Листинг 2.13. Просмотр символов шрифта 8×16 в графическом режиме в увеличенном масштабе -- -- -- -- -- -- -- 167
2.2.6.2.7. Анимация двухмерных изображений -- -- -- -- -- 168
Листинг 2.14. Подпрограмма, выполняющая настройку регистра FS на видеопамять, регистра GS — на дополнительную память **(list4_14.inc)** -- -- -- 171
Листинг 2.15. Процедуры вывода символа и очистки экрана для 256-цветных режимов с линейной адресацией видеобуфера **(list4_15.inc)** -- -- -- 174
Листинг 2.16. Описание констант, глобальных переменных и макрокоманд **(list4_16.inc)** -- -- -- -- -- -- -- 175
Листинг 2.17. Набор масок объектов, массивов координат и случайных чисел для игры «Самолет и ракета» **(list4_17.inc)** -- -- -- -- -- 176
Листинг 2.18. Набор подпрограмм для вывода фона и динамических объектов в одностраничном режиме **(list4_18.inc)** -- -- -- -- -- 178
Листинг 2.19. Универсальные подпрограммы, пригодные для одностраничного и многостраничного режимов **(list4_19.inc)** -- -- -- -- -- 180

Листинг 2.20. Игра «Самолет и ракета» — основной модуль программы для одностраничного режима -- -- -- -- -- -- -- 183
2.2.6.3. Простые форматы графических файлов -- -- -- -- -- 186
2.2.6.3.1. Формат BMP для несжатого RGB-изображения -- -- -- -- 186
2.2.6.3.2. Формат PCX для 256-цветных изображений -- -- -- -- -- 186
Контрольные вопросы -- -- -- -- -- -- -- 188
2.3. Две видеосистемы в одном компьютере -- -- -- -- -- 190
2.3.1. Аппаратная совместимость видеоадаптеров -- -- -- -- -- 190
2.3.1.1. Видеоадаптеры с интерфейсом ISA -- -- -- -- -- -- 190
2.3.1.2. Видеоадаптеры с интерфейсами PCI и AGP -- -- -- -- -- 190
2.3.2. Статус видеоадаптеров -- -- -- -- -- -- -- 191
2.3.3. Инсталляция двух видеосистем -- -- -- -- -- -- 191
2.3.3.1. Установка видеоадаптеров -- -- -- -- -- -- -- 191
2.3.3.2. Программная поддержка Windows 98 -- -- -- -- -- -- 191
2.3.3.3. Программная настройка видеосистем -- -- -- -- -- -- 193
2.3.4. Преимущества и недостатки одновременного использования двух видеосистем -- -- -- -- -- -- -- -- -- 195
Контрольные вопросы -- -- -- -- -- -- -- 196
2.4. Мультимедийные функции видеоадаптера. TV-тюнер -- -- -- 197
2.4.1. Устройство TV-тюнера -- -- -- -- -- -- -- 197
2.4.2. Принцип работы TV-тюнера -- -- -- -- -- -- -- 198
2.4.2.1. Прием и преобразование телевизионного сигнала -- -- -- 198
2.4.2.2. Преобразование параметров развертки -- -- -- -- -- 199
2.4.2.3. Смешение RGB- и ТВ-сигналов -- -- -- -- -- -- 199
2.4.2.3.1. Аналоговое смешение -- -- -- -- -- -- -- 199
2.4.2.3.2. Цифровое смешение -- -- -- -- -- -- -- 200
2.4.3. Дополнительные функции TV-тюнера -- -- -- -- -- 200
2.4.3.1. Захват отдельных кадров и ввод видеосигналов -- -- -- -- 201
2.4.3.2. Дистанционное управление -- -- -- -- -- -- -- 201
2.4.3.3. Прием радиопрограмм -- -- -- -- -- -- -- 201
Контрольные вопросы -- -- -- -- -- -- -- 201
3. Аппаратные средства ускорения трехмерной графики -- -- 203
3.1. Основы технологии синтеза 3D-изображений -- -- -- -- 204
3.1.1. 3D-конвейер -- -- -- -- -- -- -- -- -- 204
Контрольные вопросы -- -- -- -- -- -- -- 207
3.1.2. Методы коррекции трехмерных изображений -- -- -- -- 209
3.1.2.1. Затенение -- -- -- -- -- -- -- -- -- 209
3.1.2.1.1. Плоское (простое) затенение -- -- -- -- -- -- 209
3.1.2.1.2. Затенение по методу Гуро -- -- -- -- -- -- -- 210
3.1.2.1.3. Затенение по методу Фонга -- -- -- -- -- -- -- 211
3.1.2.2. Удаление скрытых поверхностей -- -- -- -- -- -- 213
3.1.2.2.1. Отбрасывание задних граней -- -- -- -- -- -- 214
3.1.2.2.2. z-сортировка -- -- -- -- -- -- -- -- 214
3.1.2.2.3. z-буферизация -- -- -- -- -- -- -- -- 215
3.1.2.3. Оптимизация наложения текстур -- -- -- -- -- -- 215
3.1.2.3.1. Мип-мэппинг (MIP-mapping) -- -- -- -- -- -- 215
3.1.2.3.2. Фильтрация -- -- -- -- -- -- -- -- 216
3.1.2.3.3. Мультитекстурирование -- -- -- -- -- -- -- 218
3.1.2.3.4. Текстурирование с коррекцией перспективы -- -- -- -- -- 218
3.1.2.3.5. Моделирование рельефа при наложении текстур -- -- -- -- 218
3.1.2.3.5.1. Текстурирование методом вдавливания -- -- -- -- -- 219
3.1.2.3.5.2. Текстурирование с использованием карт окружающей среды -- -- 220
3.1.2.3.5.3. Динамическое изменение рельефной карты -- -- -- -- 221
3.1.2.4. Управление прозрачностью и цветовой насыщенностью текстур -- -- 221
3.1.2.4.1. Альфа-смешение -- -- -- -- -- -- -- -- 221
3.1.2.4.2. Затуманивание -- -- -- -- -- -- -- -- 222
3.1.2.4.3. Морфинг уровней детализации -- -- -- -- -- -- 223
3.1.2.4.4. Управление прозрачностью с использованием маски -- -- -- -- 223

3.1.2.4.5. Зеркальные блики -- -- -- -- -- -- -- -- 223
3.1.2.5. Способы хранения текстур -- -- -- -- -- -- -- 223
3.1.2.5.1. Режим с палитрой -- -- -- -- -- -- -- -- 223
3.1.2.5.2. Технология сжатия текстур S3TC -- -- -- -- -- -- 224
3.1.2.6. Антиалиасинг -- -- -- -- -- -- -- -- 225
3.1.2.6.1. Краевой антиалиасинг -- -- -- -- -- -- -- 225
3.1.2.6.2. Полный антиалиасинг -- -- -- -- -- -- -- 225
3.1.2.7. Интерполяция недостающих цветов -- -- -- -- -- -- 226
Контрольные вопросы -- -- -- -- -- -- -- 227
3.1.3. Интерфейсы прикладного программирования -- -- -- -- 229
3.1.3.1. DirectX -- -- -- -- -- -- -- -- -- 229
3.1.3.2. OpenGL -- -- -- -- -- -- -- -- -- 230
3.1.3.2.1. ICD -- -- -- -- -- -- -- -- -- 230
3.1.3.2.2. MCD -- -- -- -- -- -- -- -- -- 230
3.1.3.2.3. Минипорт -- -- -- -- -- -- -- -- -- 230
3.1.3.2.4. Раппер -- -- -- -- -- -- -- -- -- 231
3.1.3.2.5. Game Engine -- -- -- -- -- -- -- -- 231
3.1.3.3. Проект Fahrenheit -- -- -- -- -- -- -- -- 231
3.1.3.4. Оптимизация 3D-конвейера средствами API -- -- -- -- -- 231
Контрольные вопросы -- -- -- -- -- -- -- 234
3.2. Устройство и основные характеристики ускорителей 3D-графики -- 235
3.2.1. Структура типового 3D-акселератора -- -- -- -- -- 235
3.2.2. Основные характеристики 3D-акселераторов -- -- -- -- 236
3.2.3. 3D-акселераторы фотореалистичной графики -- -- -- -- 237
3.2.4. Ускоренный графический порт (AGP) -- -- -- -- -- 238
3.2.4.1. Режимы «2x» и «4x» -- -- -- -- -- -- -- -- 239
3.2.4.2. Режим Pipelining -- -- -- -- -- -- -- -- 239
3.2.4.3. Режим SBA -- -- -- -- -- -- -- -- -- 240
3.2.4.4. Режим DiME -- -- -- -- -- -- -- -- 240
3.2.4.5. Слот AGP -- -- -- -- -- -- -- -- -- 241
3.2.4.6. Преимущества и недостатки интерфейса AGP -- -- -- -- -- 241
3.2.4.7. Программно-аппаратная настройка видеосистемы с интерфейсом AGP -- -- 242
Контрольные вопросы -- -- -- -- -- -- -- 243
3.3. Chipset для 3D-акселераторов -- -- -- -- -- -- 245
3.3.1. 3Dfx Interactive -- -- -- -- -- -- -- -- 245
3.3.1.1. Voodoo Graphics -- -- -- -- -- -- -- -- 245
3.3.1.2. Voodoo Rush -- -- -- -- -- -- -- -- 245
3.3.1.3. Voodoo^2 -- -- -- -- -- -- -- -- -- 246
3.3.1.4. Voodoo Banshee -- -- -- -- -- -- -- -- 246
3.3.1.5. Voodoo^3 -- -- -- -- -- -- -- -- -- 247
3.3.2. Intel -- -- -- -- -- -- -- -- -- -- 247
3.3.2.1. i740 -- -- -- -- -- -- -- -- -- -- 247
3.3.3. nVidia -- -- -- -- -- -- -- -- -- 248
3.3.3.1. Riva 128 -- -- -- -- -- -- -- -- -- 248
3.3.3.2. Riva 128ZX -- -- -- -- -- -- -- -- -- 249
3.3.3.3. Riva 128TNT (Riva TNT) -- -- -- -- -- -- -- 249
3.3.3.4. Chipset семейства Riva TNT2 -- -- -- -- -- -- -- 249
3.3.3.4.1. Riva TNT2-A -- -- -- -- -- -- -- -- 250
3.3.3.4.2. Riva TNT2 Ultra -- -- -- -- -- -- -- -- 250
3.3.3.4.3. Riva TNT2 Vanta -- -- -- -- -- -- -- -- 250
3.3.3.4.4. Riva TNT2 M64 -- -- -- -- -- -- -- -- 251
3.3.3.5. Chipset семейства GeForce -- -- -- -- -- -- -- 251
3.3.3.5.1. GeForce 256 -- -- -- -- -- -- -- -- 251
3.3.3.5.2. GeForce 2 GTS -- -- -- -- -- -- -- -- 251
3.3.4. Matrox -- -- -- -- -- -- -- -- -- 252
3.3.4.1. Matrox MGA G200 -- -- -- -- -- -- -- -- 252
3.3.4.2. Matrox MGA G400 -- -- -- -- -- -- -- -- 253
3.3.5. ATI -- -- -- -- -- -- -- -- -- -- 254
3.3.5.1. ATI Rage 128 -- -- -- -- -- -- -- -- 255
3.3.5.2. ATI Rage 128 Pro -- -- -- -- -- -- -- -- 255

3.3.5.3. ATI Rage Fury MAXX -- -- -- -- -- -- -- 255
3.3.5.4. ATI Radeon 256 -- -- -- -- -- -- -- -- 256
3.3.6. S3, Inc. -- -- -- -- -- -- -- -- -- 256
3.3.6.1. S3 Savage 3D -- -- -- -- -- -- -- -- 257
3.3.6.2. S3 Savage 4 -- -- -- -- -- -- -- -- -- 257
3.3.6.3. S3 Savage 2000 -- -- -- -- -- -- -- -- 258
3.3.7. 3Dlabs -- -- -- -- -- -- -- -- -- 259
3.3.8. Современные графические процессоры. -- -- -- -- -- 259
Контрольные вопросы -- -- -- -- -- -- -- 264
Приложение 1. Список WWW-адресов производителей видеоадаптеров -- **266**
Приложение 2. Список терминов и сокращений -- -- -- -- -- 268
Алфавитный указатель -- -- -- -- -- -- -- -- 281
Список литературы -- -- -- -- -- -- -- -- 291

Предисловие

Данное учебно-справочное пособие посвящено одной из важнейших систем персональных компьютеров - видеосистеме, а именно её основной составляющей - видеоадаптерам.

В первой главе приводятся сведения о структуре видеосистемы ПК, состоящей из двух основных компонентов - видеомонитора и видеоадаптера, о принципах ее функционирования в графическом и текстовом режимах, а также об этапах ее развития от монохромной системы MDA до видеосистемы мультимедийного ПК.

Во второй главе рассматривается устройство, регистровая архитектура и особенности работы стандартного видеоадаптера VGA, а также его системная поддержка на уровне VGA BIOS. Обсуждаются основные направления совершенствования видеосистемы: использование аппаратного ускорения графических функций, быстродействующей памяти, скоростных интерфейсов между видеоадаптером и материнской платой; унификация видеоадаптеров SVGA; реализация в видеоадаптерах мультимедийных функций и пр. Особое внимание во второй главе уделено рассмотрению особенностей работы с видеоадаптером в различных режимах из среды MS DOS, включая работу с использованием режима линейной адресации в адресном пространстве 4 Гбайт. Приводится широкий набор программ на языке Ассемблер, иллюстрирующих эти особенности.

Третья глава посвящена рассмотрению вопросов, связанных с использованием аппаратных средств ускорения трехмерной графики: основные технологии синтеза 3D-изображений; устройство и основные характеристики ускорителей 3D-графики; чипсеты для 3D-акселераторов.

Каждый логический раздел пособия оканчивается перечнем контрольных вопросов.

В конце пособия приводится перечень используемых терминов и сокращений, список использованной литературы, алфавитный указатель и WWW-адреса производителей видеоадаптеров.

Данное пособие является информационно-справочной поддержкой раздела курса лекций "Адаптеры и контроллеры ЭВМ", читаемого автором в Саровском государственном физико-техническом институте (СарФТИ) в рамках специальности "Вычислительные машины, комплексы, системы и сети" и других, родственных ей специальностей. Широкий спектр информации, представленной в пособии, позволяет использовать его и в ряде других курсов, связанных с использованием и изучением возможностей видеосистемы ПК.

Автор

ВВЕДЕНИЕ

В последнее время одним из наиболее модных и интригующих понятий, связанных с персональным компьютером (PC), стал термин *"мультимедиа"*. Сегодня при покупке компьютера трудно найти модель, которую бы продавцы не предлагали как *мультимедийный PC*. При этом необходимым и достаточным условием его «мультимедийности» до сих пор принято считать наличие в PC привода CD-ROM и звуковой карты с подключенной к ней акустической системой. Однако такой упрощенный подход не совсем справедлив, точнее, совсем не справедлив. Как часто случается, новые термины и понятия часто употребляют, до конца не понимая их смысла, особенно, если они иноязычного происхождения. Ярким примером такой ситуации служит и употребление термина «мультимедиа»: практически в любой книге или журнальной статье, посвященной PC, можно встретить этот термин, однако вам вряд ли удастся отыскать его четкое определение. В свою очередь, нечеткость терминологии заставляет пользователя каждый раз искать ответ на вопрос: какие именно аппаратные средства необходимы PC, чтобы его можно было назвать мультимедийным?

В соответствии с одной из возможных трактовок, термин «мультимедиа» *(multimedia)* означает совокупность множества сфер (сред) информационного взаимодействия человека с компьютером. Несколько витиеватое понятие «среда информационного взаимодействия» — не что иное, как материальная среда, воздействующая на органы чувств человека.

До недавнего времени диалог между человеком и компьютером осуществлялся исключительно визуально, при помощи символов на экране видеотерминала или на листе бумаги. Как только в состав PC были включены аппаратные средства цифровой обработки звука и компьютер «заговорил», появилась возможность воздействия на орган слуха человека. Собственно, с этого момента и возникло понятие мультимедиа, хотя в то время под «множеством сфер» подразумевались всего две: визуальная и акустическая.

В настоящее время наметились новые тенденции развития аппаратных средств мультимедиа:

- непрерывно увеличивается количество сфер информационного воздействия на человека: наряду с традиционными средствами — визуальными (средства отображения информации) и акустическими (средства цифровой обработки звука), — появились устройства, создающие тактильные ощущения, т. е. использующие способность человека осязать объекты;
- смещаются акценты в разработке новых аппаратных средств мультимедиа: помимо быстродействия, все большее внимание уделяется качеству создаваемых компьютером виртуальных образов (зрительных, слуховых, тактильных), иными словами, разработчики мультимедийных средств за счет использования различных эффектов (объемный звук, стереоскопическое изображение и т. п.) стремятся к тому, чтобы виртуальные образы были как можно больше похожи на реальные объекты окружающего мира.

В результате возникли качественно новые системы — *системы виртуальной реальности,* предназначенные для решения основной задачи мультимедиа: создание у человека иллюзии реальности искусственного, виртуального мира. Очевидно, что достичь этой цели можно лишь в том случае, если формировать виртуальные образы с учетом особенностей органов чувств человека, т. е. визуальные, звуковые и тактильные эффекты должны восприниматься человеком как реальные явления.

Наиболее полные сведения об окружающим мире человек получает визуально. Недаром существует поговорка: «Лучше один раз увидеть, чем сто раз услышать». Иными словами, с помощью зрения человек получает несоизмеримо больший объем информации, нежели с помощью любого другого органа чувств. Именно поэтому качество, реализм формируемого видеосистемой PC изображения играет решающую роль в создании *виртуальной реальности.*

Таким образом, видеосистема является важнейшим элементом современного мультимедийного PC. Ее конфигурация и настройка определяют не только быстродействие системы, но и степень проникновения в виртуальный мир.

В настоящее время стоимость видеосистемы составляет примерно 60% от общей стоимости мультимедийного PC. Поэтому каждый раз при модернизации или покупке PC приходится решать проблему экономически оправданного выбора видеосистемы с точки зрения ее соответствия решаемым задачам. Для оптимального выбора видеосистемы необходимо знать основные характеристики и четко представлять, какие из них являются наиболее значимыми для успешного решения поставленных задач.

Состав аппаратных средств видеосистемы современного мультимедийного PC отличается невероятным разнообразием. Если раньше под видеосистемой понимались традиционный монитор и сравнительно простой видеоадаптер, то сегодня в ее состав входят ускорители двумерной и трехмерной графики, устройства цифровой обработки и воспроизведения видеоданных, разнообразные по конструкции и возможностям устройства формирования трехмерных изображений, мультимедийные проекторы, устройства приема телевизионных сигналов (TV-тюнеры) и т. п.

Видеосистема современного PC характеризуется множеством параметров, среди которых пользователю без соответствующей подготовки или специального технического образования разобраться довольно трудно. Где же почерпнуть необходимые знания?

Во-первых, из книг, посвященных аппаратным средствам PC. Однако в таких книгах содержится, как правило, достаточно поверхностное описание видеосистемы и ее основных характеристик, позволяющее получить лишь общее представление о видеосистеме (обычно это один из разделов книги).

Во-вторых, можно воспользоваться специальной литературой, посвященной решению конкретных задач видеосистемы, но такая литература зачастую изобилует специфическими терминами и требует определенных знаний в области системного программирования и будет мало понятна широкому кругу пользователей.

Сведения о новейших достижениях в области видеосистемы мультимедийного PC (о трехмерной графике, стереоскопии, цифровой обработке и воспроизведении видеоданных и т. п.) можно найти в периодической печати, в частности в журнальных статях, или на сайтах в Интернет, посвященных компьютерной тематике. К сожалению, эта информация в большинстве случаев является субъективной и носит противоречивый характер.

Таким образом, достаточно полного и систематизированного описания видеосистемы современного мультимедийного PC в одной книге вам вряд ли удастся найти.

Именно частичному решению этой задачи посвящены материалы приведенные в этом пособии. Оно посвящено главной составляющей видеосистемы - видеоадаптерам мультимедийных ПК.

Видеосистема ПК — это цифровая система. Внутри нее циркулирует двоичная информация, которая отображается на экране монитора в виде символов или рисунков. Для того чтобы понять, как действует видеосистема, необходимо знать базовые принципы ее построения и функционирования. Этому посвящена *первая глава* книги. В этой главе содержится также краткий исторический обзор эволюции видеосистемы, начиная с видеосистем первых моделей IBM PC и заканчивая видеосистемой современного мощного мультимедийного компьютера.

Вторая глава посвящена видеоадаптеру. В этой главе рассмотрены устройство, принцип действия и основные характеристики видеоадаптера мультимедийного PC. Описана базовая архитектура стандартного видеоадаптера VGA и особенности современных видеоадаптеров Super VGA. Особое внимание уделено мультимедийным возможностям видеосистемы, в частности, возможностям использования в PC двух видеосистем и приема телевизионных программ.

Не секрет, что компьютер — это не только рабочий инструмент, но и средство развлечения. Игры особенно зрелищны тогда, когда на экране монитора формируются объекты с использованием эффектов, подчеркивающих их трехмерность (тени, освещение, перспектива и т. п.). Получить такое изображение в реальном масштабе времени можно только тогда, когда видеосистема PC снабжена специальным устройством — *ускорителем трехмерной графики,* или *3D-акселератором.* Не будет преувеличением сказать, что 3D-графика — одно из главных направлений развития PC на ближайшие несколько лет. Поэтому аппаратным средствам ускорения трехмерной графики посвящена отдельная, *третья глава* книги. В этой главе описана технология создания трехмерных изображений (так называемый 3D-*конвейер*), рассмотрен принцип действия графического процессора и дан обзор чипсетов (*Chipset* — набор микросхем) наиболее распространенных 3D-акселераторов.

Дополнительную информацию по видеосистемам можно найти в изданиях, посвященных аппаратным средствам современного мультимедийного PC, а именно «PC Magazine», «Компьютер ПРЕСС», «625», «Мир ПК», «Мультимедиа», «Byte», «Publish», «Hard'n'Soft», а также на серверах сети Интернет, особенно iXBT Hardware (http:// ixbt.com), содержащих наиболее полные сведения о новинках аппаратного обеспечения PC.

Контрольные вопросы.

1. Что понимается под термином "Мультимедиа"?
2. Какие тенденции наблюдаются в развитии аппаратных средств мультимедиа?
3. Что понимается под системой виртуальной реальности?
4. Что понимается под видеосистемой мультимедийного компьютера?

1. СТРУКТУРА ВИДЕОСИСТЕМЫ PC.

Под *видеосистемой* персонального компьютера (PC — *Personal Computer)* будем понимать совокупность устройств, выполняющих следующие функции:

- формирование и обработку цифрового изображения;
- преобразование цифрового изображения в видеосигналы, поступающие на устройство отображения;
- формирование изображения.

Первую из этих задач видеосистема решает совместно с основным элементом компьютера — центральным процессором (CPU — *Central Processing Unit).* Остальные — преобразование цифровых данных в визуальные образы: цифры, рисунки, фотографии, видеосюжеты и т. п. — видеосистема решает самостоятельно.

Основой видеосистемы и неотъемлемой частью PC является устройство отображения (визуализации) информации — монитор на основе электронно-лучевой трубки (ЭЛТ).

Как известно, изображение на экране ЭЛТ формируется путем последовательной прорисовки строк, образующих растр. Подобный метод синтеза изображения требует подачи на ЭЛТ аналогового сигнала, описывающего изменение яркости изображения вдоль каждой строки растра. С другой стороны, внутри компьютера вся информация (в том числе графическая) представлена двоичными числами, или цифровым кодом. Для преобразования цифровой информации в аналоговые электрические сигналы (т. е. для выполнения второй из названных выше функций) в состав видеосистемы входит специальное согласующее устройство. Это устройство называют *видеоадаптером* (видеокартой, видеоплатой). Оно выполняет роль интерфейса между центральной частью компьютера и устройством отображения (монитором).

Работа видеоадаптера осуществляется под управлением программного обеспечения (ПО). Зачастую именно некорректность работы программного обеспечения ограничивает потенциальные возможности аппаратных средств, составляющих видеосистему.

Таким образом, видеосистема PC включает в себя следующие обязательные элементы:

- устройство отображения информации;
- видеоадаптер;
- программное обеспечение.

В зависимости от назначения PC в состав видеосистемы могут входить и другие аппаратные средства, предназначенные для улучшения ее работы или для обеспечения новых функциональных возможностей компьютера (например, графический акселератор, средства оцифровки телевизионного и видеосигнала и др.). Как правило, эти устройства являются дополнительными, причем спектр их возможных конструктивных решений чрезвычайно широк: они могут быть выполнены в виде отдельных функциональных элементов или быть встроенными в видеоадаптер.

Чтобы уяснить для себя место и роль видеосистемы в составе персонального компьютера, рассмотрим архитектуру PC.

1.1. Архитектура PC

На рис. 1.1 изображена упрощенная архитектура персонального компьютера. Элементы, входящие в состав его видеосистемы, обведены сплошной линией. Помимо видеосистемы, PC имеет типичный набор аппаратных средств: CPU, контроллеры периферийных устройств и шин, оперативное запоминающее устройство и др. Структура видеосистемы, изображенная на рис. 1.1, во многом повторяет базовую архитектуру IBM PC.

Главным элементом PC является CPU. Прикладные программы и данные размещаются в оперативном запоминающем устройстве (ОЗУ), или RAM *(Random Access Memory* — память с произвольным доступом). CPU и RAM связаны друг с другом *системной шиной* — высокоскоростной шиной, содержащей линии (шины) адреса и данных. Разрядность системной шины определяется разрядностью внешних регистров данных CPU. При этом разрядность шины данных и тактовая частота шины определяют скорость обмена информацией между CPU и RAM, а разрядность шины адреса — максимальный объем адресуемой памяти, или размер *адресного пространства.* Например, процессор Intel 8088, на основе которого был создан первый персональный компьютер IBM PC, имел 8-разрядную шину данных и 20-разрядную шину адреса. Процессоры 286 и 386SX имеют 16-разрядную шину данных и 24-разрядную шину адреса (16/24), процессоры 386DX и 486 являются 32-разрядными (32/32), а процессоры Pentium и выше — 64-разрядными (64/32). Системная шина работает на довольно высокой тактовой частоте: до 66 МГц для процессоров Pentium и до 133 МГц и выше для процессоров Pentium II, III. IV и им подобным..

CPU и RAM составляют ядро (центральную часть) архитектуры IBM PC. Все остальные устройства PC по отношению к ним являются *внешними* (периферийными). Для связи внешних устройств с CPU используется *шина ввода/вывода.* Обычно она имеет меньшие, чем системная шина, разрядность и тактовую частоту. Для сопряжения шины ввода/вывода с системной шиной в состав IBM PC входит специальное устройство — *контроллер шины,* или *шинный арбитр.* Для компьютеров на основе процессоров 286 и 386 стандартной шиной ввода/вывода являлась шина ISA (*Industrial Standard Architecture* — стандартная

промышленная архитектура). В настоящее время стандартной шиной ввода/вывода PC является шина PCI *(Peripheral Component Interconnect* — соединение периферийных устройств) а для видеоадаптеров AGP (Accelerated Graphic Port — ускоренный графический порт).

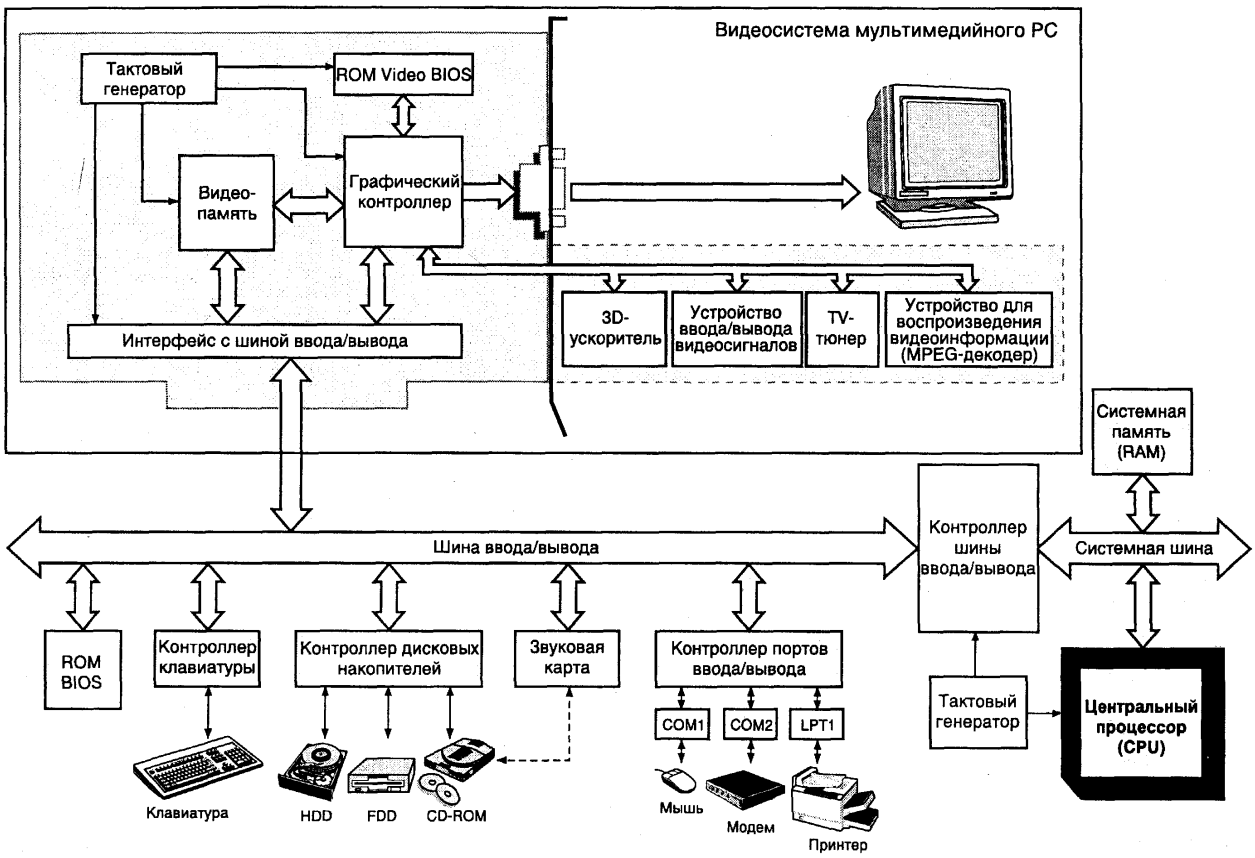


Рис.1.1. Архитектура PC.

Некоторые внешние устройства, такие как, например, контроллеры жесткого диска и дисководов, устанавливаются на материнской плате PC, другие — звуковая и сетевая карты, адаптер интерфейса SCSI и т. п. — обычно подключаются к шине ввода/вывода через специальные разъемы, так называемые *слоты расширения.* Последние еще называют *картами расширения,* так как они являются внешними относительно не только CPU, но и материнской платы и предназначены для расширения уже существующей конфигурации PC.

Для обмена данными с CPU каждое внешнее устройство имеет свой адрес в пределах общего адресного пространства PC. Доступ к внешним устройствам осуществляется центральным процессором так же, как и к ячейкам ОЗУ.

Все внешние устройства соединяются с шиной ввода/вывода через специальные устройства, называемые *адаптерами.* Адаптеры как правило выполняются на базе одного или нескольких специализированных устройств, называемых *контроллерами.* Управление контроллерами осуществляется командами центрального процессора. Получив команду, контроллер формирует последовательность собственных команд, «понятных» подключенному к нему устройству (например, изменяет положение магнитных головок в дисководе, инициирует передачу данных на модем или принтер и т. п.). Таким образом, контроллер, по сути, является специализированным информационно-управляющим устройством, имеющим свою собственную систему низкоуровневых команд.

При выполнении операций ввода/вывода на контроллеры IBM PC необходимо подавать стандартные последовательности команд — выполнять системные программы. Эти программы записываются в специальное *постоянное запоминающее устройство* (ПЗУ), называемое ROM BIOS *(Read Only Memory Base Input/Output System* — ПЗУ, содержащее *базовую систему ввода/вывода).* BIOS занимает не более 64 Кбайт памяти. Кроме программ по управлению вводом/выводом, в ROM BIOS содержатся программы начального тестирования оборудования компьютера, программа начальной загрузки, различные константы, определяющие текущую конфигурацию PC, и другая информация.

Видеосистема PC имеет аналогичный принцип построения. Основным элементом видеосистемы является видеоадаптер, а одним из основных элементов видеоадаптера — *графический контроллер.* В зависимости от типа видеоадаптера в качестве графического контроллера может быть использована как обычная комбинационная схема с жестким алгоритмом работы, так и сложный *графический процессор с* программируемыми функциями.

Графический контроллер получает команды от CPU по шине ввода/вывода. По этим командам он модифицирует цифровое изображение, хранящееся в специальном буфере — *видеопамяти.* В свою очередь *видеоконтроллер*, в нашем случае входящий в состав графического контроллера, преобразует содержимое видеопамяти в видеосигнал и подает на устройство отображения (монитор).

Видеопамять представляет собой специализированное быстродействующее ОЗУ, размещенное на плате видеоадаптера и предназначенное для хранения текущего цифрового изображения. Содержимое видеопамяти представляет собой цифровой образ того изображения, которое в данный момент отображается на экране монитора. Для ячеек видеопамяти зарезервирована также часть общего адресного пространства PC, поэтому CPU, используя шину ввода/вывода, может непосредственно записывать в них необходимую информацию.

Разрядность и тактовая частота графического контроллера могут отличаться от аналогичных характеристик шины ввода/вывода. Для согласования с шиной в состав видеоадаптера входит специальная *схема интерфейса с шиной ввода/вывода.*

Под управлением прикладной программы CPU создает в RAM цифровой образ изображения и пересылает его в видеопамять. Одновременно с цифровым изображением CPU передает в графический контроллер соответствующие команды управления. По этим командам из специального ПЗУ, которое, по аналогии с ROM BIOS, называется *ROM Video BIOS,* считываются микропрограммы управления графическим контроллером. Работа всех компонентов видеоадаптера синхронизируется сигналами *тактового генератора.*

Видеосистема мультимедийного PC может выполнять обширный набор дополнительных функций: аппаратное ускорение построения трехмерных изображений, оцифровку отдельных кадров и видеопоследовательностей, прием и отображение на экране монитора телевизионных программ, воспроизведение видеофильмов и др. Для реализации этих функций в состав видеосистемы дополнительно включают соответствующие аппаратные компоненты, которые на рис. 1.1 обведены пунктирной линией. Изначально эти устройства выполнялись в виде отдельных карт расширения, однако в настоящее время они все чаще устанавливаются на плате видеоадаптера, превращая последний в мощную универсальную графическую плату.

Видеосистема PC имеет ряд особенностей, благодаря которым и занимает особое место среди других подсистем компьютера.

1. Учет физиологических особенностей человека. Ни в какой другой подсистеме PC так полно не учитываются физиологические особенности оператора (в первую очередь его зрения), как в видеосистеме. Качество формируемого изображения и комфортные условия работы оператора являются одними из главных критериев, определяющих возможности PC для решения конкретной задачи. Например, с помощью высокопроизводительного компьютера, видеосистема которого обеспечивает низкое качество изображения, практически невозможно решить ни одну из современных задач мультимедиа.

2. Интенсивный обмен информацией между видеоадаптером и CPU. Для представления изображения в цифровом виде необходим очень большой массив данных. Время пересылки этих данных не должно быть больше, чем период кадровой развертки монитора. Если пропускная способность шины ввода/вывода не позволяет этого сделать, производительность PC снижается. В последние годы низкая скорость обмена данными между CPU и видеосистемой стала одним из главных факторов, ограничивающих быстродействие PC. Для решения этой проблемы разрабатываются новые и продолжают совершенствоваться уже существующие специализированные высокоскоростные шины, предназначенные для нужд видеосистемы: PCI и AGP.

3. Высокая загруженность CPU. Синтез цифрового изображения и пересылка его в видеоадаптер приводят к высокой загруженности CPU. Поэтому качественные показатели видеосистемы (в первую очередь, видеоадаптера) значительно влияют на общую производительность PC. Для снижения загрузки CPU в состав современных видеоадаптеров включают графический процессор, обеспечивающий аппаратное выполнение большинства функций, необходимых при построении изображения. При этом CPU решает лишь самые необходимые задачи (например, пересылает изображение из RAM в видеопамять), благодаря чему общая производительность PC заметно возрастает.

4. Высококачественное программное обеспечение. Для корректной работы видеосистемы особенно важно качество ПО. Если большинство других подсистем PC имеют стандартное ПО, содержащееся в ROM BIOS, то для работы каждой видеосистемы (видеоадаптера) требуется специальное ПО, или *драйвер.* Пользователь PC обычно не задумывается над тем, какие программы управляют работой его жесткого диска или порта ввода/вывода — эти программы автоматически загружаются из ROM BIOS при включении PC. Другое дело — видеосистема. Для реализации всех ее возможностей необходимо установить соответствующее программное обеспечение и произвести его настройку.

Все это свидетельствует об огромной роли видеосистемы как составной части PC и особенно — мультимедийного.

1.2. Принцип действия видеосистемы PC

1.2.1. Формирование изображения на экране монитора

Традиционным и самым распространенным устройством отображения информации является монитор на основе электронно-лучевой трубки. Поэтому описание принципа работы видеосистемы PC начнем с рассмотрения принципа работы ЭЛТ.

1.2.1.1. Конструкция ЭЛТ

ЭЛТ — это электронно-вакуумный прибор, предназначенный для преобразования электрического сигнала, формируемого видеоадаптером, в изображение. Это изображение создается за счет возвратно-поступательного движения электронного луча, создаваемого специальным устройством — *электронной пушкой,* или *электронным прожектором.* Экран ЭЛТ изнутри покрыт специальным веществом — *люминофором,* — которое обладает способностью светиться при попадании на него электронов. Чем интенсивнее поток электронов, тем ярче светится люминофор. В свою очередь, интенсивность электронного луча пропорциональна напряжению, поступающему от видеоадаптера на специальный управляющий электрод электронной пушки — *модулятор.*

ЭЛТ цветного и монохромного монитора отличаются друг от друга конструкцией. В монохромных мониторах используется ЭЛТ с одной-единственной электронной пушкой и сплошным люминофорным покрытием экрана. ЭЛТ цветного монитора создавалась с учетом психофизических свойств человеческого зрения. Как известно, на глазном дне, куда фокусируется изображение, имеется множество специальных микроскопических рецепторов — *палочек* и *колбочек.* Палочки реагируют на интенсивность света, а колбочки — на его цвет. Колбочки объединены в триады, причем каждая из них реагирует только на один из трех основных цветов: красный *(Red),* зеленый *(Green)* или синий *(Blue).* Размер триады колбочек определяет минимальный видимый размер точки, цвет которой человек еще может различить, т. е. цветовую разрешающую способность глаза. В зависимости от степени раздражения каждой из колбочек триады создается ощущение цвета точки.

В ЭЛТ цветных мониторов имеются три электронные пушки, а экран покрыт триадами зерен люминофора, каждое из которых при бомбардировке электронами светится своим цветом. В качестве цветов выбраны *основные* цвета видимого спектра. Луч каждой из электронных пушек попадает только на свои зерна триады.

Поскольку размер зерна люминофора весьма мал и соответствует разрешающей способности глаза, близкорасположенные зерна триады воспринимаются как одна точка, цвет которой определяется законом *пространственного смешения основных цветов.* Изменяя яркость свечения каждого из трех зерен люминофора, можно получить любой цвет.

Важным элементом цветной ЭЛТ является *цветоделительная маска.* Как следует из ее названия, маска обеспечивает попадание каждого из трех электронных лучей на зерна люминофора своего цвета. Цветоделительная маска должна с высокой точностью сохранять свою форму при нагреве, возникающем при ее бомбардировке электронами. Поэтому маску изготавливают из специального сплава — инвара — с очень низким коэффициентом температурного расширения. В зависимости от конструкции цветоделительной маски различают два типа ЭЛТ: ЭЛТ *с теневой маской*, и ЭЛТ со *щелевой маской.* В последнее время стали использовать ЭЛТ *с апертурной решеткой* типа *Trinitron.*

Более подробную информацию о принципах работы ЭЛТ различных типов можно найти в [1,2,7].

1.2.1.2. Формирование растра

Изображение (растр) на экране ЭЛТ формируется построчно, причем электронный луч движется по зигзагообразной траектории (рис. 1.2)— слева направо и сверху вниз. Электронный луч периодически

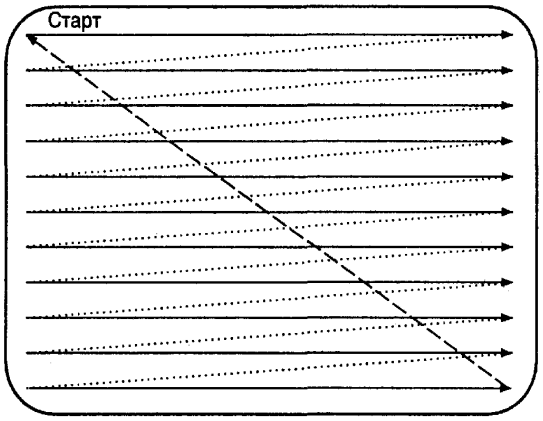


Рис. 1.2. Формирование растра на экране ЭЛТ

сканирует весь экран, образуя на нем близко расположенные строки развертки. Полный цикл движения луча, в течение которого на экране ЭЛТ оказываются прорисованными все строки изображения, образует один *кадр* изображения.

Прямой ход луча по горизонтали, в течение которого передается изображение и ток луча изменяется под действием поступающего на модуляторы ЭЛТ видеосигнала, осуществляется сигналом строчной (горизонтальной) развертки, а по вертикали — кадровой (вертикальной) развертки. Запуск генераторов строчной и кадровой разверток осуществляется специальными *синхронизирующими импульсами — строчными* и *кадровыми.* Совокупность строчных и кадровых синхроимпульсов называется *синхросигналом* или *синхросмесью.*

Перевод луча из крайней правой точки строки в крайнюю левую точку следующей строки (обратный ход луча по горизонтали) и из крайней правой позиции последней строки экрана в крайнюю левую позицию первой строки (обратный ход луча по вертикали) осуществляется специальными сигналами обратного хода. В течение обратного хода ток луча минимален. На рис. 1.2 прямой ход луча показан сплошной линией, а обратный — пунктирной.

Для предотвращения засветки люминофора лучом во время его обратного хода на модуляторы ЭЛТ подаются специальные *импульсы гашения,* или *импульсы обратного хода,* запирающие электронные пушки. Для гашения луча на время обратного хода по горизонтали используются *строчные гасящие импульсы,* а по вертикали — *кадровые гасящие импульсы.*

Стандартный телевизионный видеосигнал содержит в себе одновременно сигналы изображения (сигнал яркости и два цветоразностных сигнала), синхросигналы и гасящие импульсы. На монитор же подаются раздельные сигналы — сигналы *R, G, B* с импульсами гашения и синхросигнал. Приведенный на рис. 1.3 сигнал хотя и является телевизионным видеосигналом строки изображения последовательности разных по яркости вертикальных полос (градационного клина), недостаточно наглядно иллюстрирует зависимость яркости изображения на экране монитора от уровня видеосигнала.

Таким образом, одними из основных характеристик монитора являются частоты кадровой и строчной разверток, иными словами, *частота кадров и частота строк.*

Описанный выше способ формирования изображения применяется и в телевизорах. Здесь частота обновления *(Refresh Rate* — скорость обновления) изображения (частота кадров) составляет 25 Гц. Может показаться, что это очень низкая частота. Однако в телевидении для сокращения полосы частот спектра телевизионного сигнала применяется *чересстрочная развертка (Interlaced Mode),* т. е. полный растр формируется в два приема (рис. 1.4, *а).* Сначала за время, равное 1/50 с, воспроизводятся только четные строки: 2, 4 и т. д. Эта часть растра называется *полем четных строк* или *четным полукадром.*

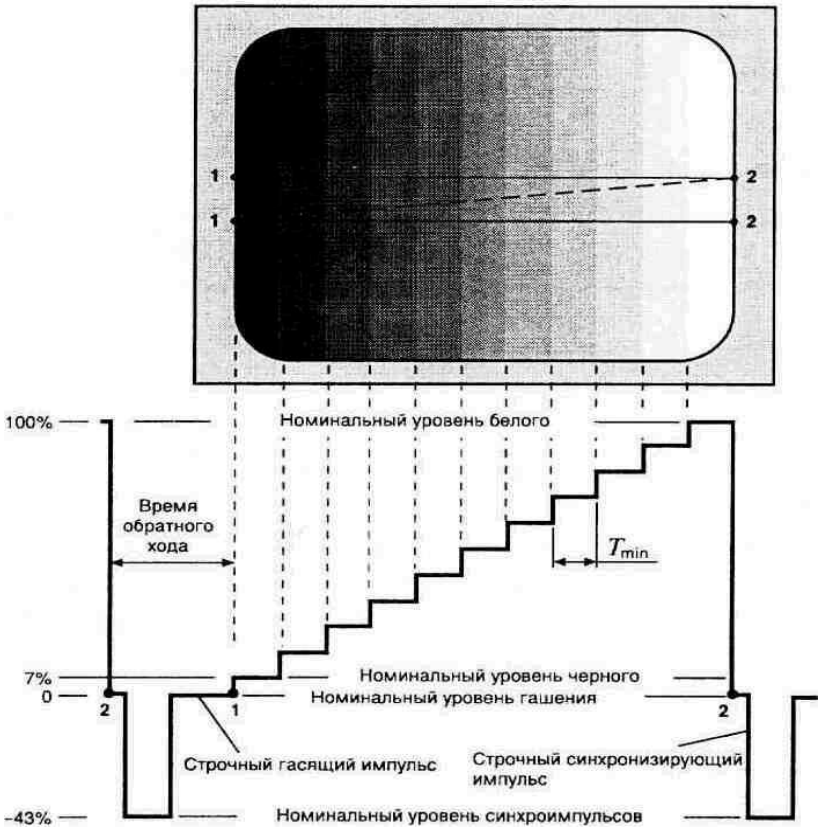


Рис. 1.3. Видеосигнал одной строки изображения градационного клина

Затем развертывающий электронный луч переводится от нижнего края экрана вверх и попадает в начало 1-й

(нечетной) строки. Далее луч прорисовывает все нечетные строки: 1, 3 и т. д. Так формируется *поле нечетных строк,* или *нечетный полукадр.* Если наложить оба полукадра друг на друга, то получится полный кадр изображения.

Применение данного способа формирования изображения как в мониторах, так и в телевизорах оказалось возможным благодаря двум особенностям, точнее, недостаткам зрения человека:

- инерционности восприятия световых раздражении;
- ограниченной разрешающей способности по перемещениям.

Первая особенность заключается в том, что возникновение и прекращение фотохимических реакций на сетчатке глаза (после начала и окончания воздействия импульса света) происходит не мгновенно, а с задержкой, характеризующей эту инерционность. Время нарастания зрительного ощущения составляет около 0,1 с, а время сохранения светового возбуждения после окончания действия светового раздражителя — 0,4—1,0 с. Благодаря такому свойству зрения, оказалось возможным производить поэлементную развертку изображения от одной строки к другой и от одного полукадра к другому (при чересстрочном способе формирования изображения), т. е. изображение представляется в виде быстро сменяющейся последовательности строк и кадров.

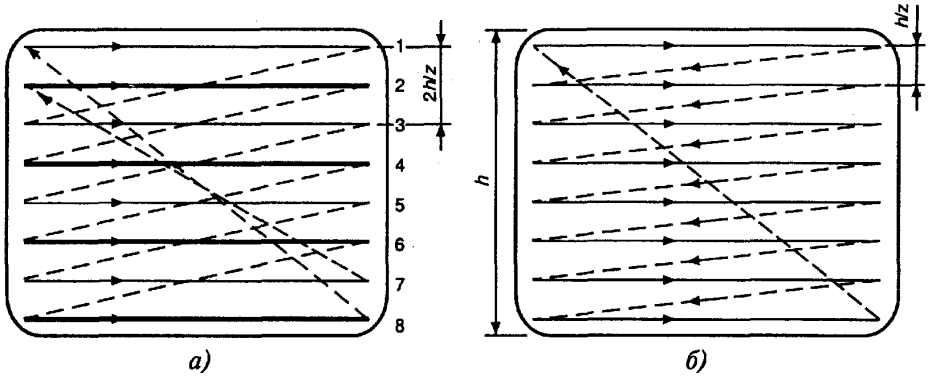


Рис. 1.4. Формирование растра при чересстрочной *(а)* и построчной *(б)* развертке

С учетом второй особенности зрения формируются изображения движущихся предметов на экране монитора или телевизора. Для того чтобы движения казались плавными, каждое изменение положения предметов должно быть передано небольшими «порциями», т. е. различия в картинках должны быть достаточно малыми (как в мультипликации). Движение передается путем покадрового воспроизведения отдельных, мало отличающихся друг от друга фаз движения.

Человеческий глаз воспринимает последовательность дискретных картинок как непрерывное динамическое изображение, если частота смены этих картинок не ниже 20—25 Гц. Исходя из этого выбиралась частота полей в телевидении. Для мониторов частота кадров имеет более важное значение, поскольку во многом определяет устойчивость изображения по вертикали (отсутствие мерцаний) и, как следствие, степень утомляемости глаз. Поэтому частоту кадров монитора PC стараются по возможности устанавливать как можно выше.

Современные мониторы поддерживают частоту кадров в диапазоне 60—120 Гц. Однако повышение частоты кадров требует увеличения частоты строчной развертки, так как уменьшается время, отводимое на формирование каждой точки растра. Частота строк примерно определяется произведением частоты вертикальной развертки на количество строк, содержащихся в одном кадре (разрешающая способность по вертикали).

1.2.1.3. Разрешение изображения

Качество изображения на экране ЭЛТ характеризуется в основном двумя параметрами: разрешением и цветностью.

Изображение на экране ЭЛТ является мозаичным, т. е. состоит из отдельных элементов. Минимальный элемент изображения называют *пикселом (Pixel— Picture Element).* Количество пикселов в строке называют *разрешением по горизонтали,* а количество пикселов по вертикали —*разрешением по вертикали.* Фактически разрешение по вертикали определяется количеством строк в растре. Под разрешением кадра (изображения) понимают общее количество содержащихся в нем пикселов и обозначают как произведение разрешений по горизонтали и вертикали, например 640×480 (640 пикселов в строке и 480 строк).

Очевидно, что пиксел не может быть меньше зерна люминофора (здесь и далее при упоминании термина «зерно люминофора» будем подразумевать триаду зерен). Поэтому *максимальное (физическое) разрешение,* которое может быть обеспечено монитором с конкретной ЭЛТ, определяется количеством зерен люминофора на экране этой ЭЛТ.

На практике размеры пиксела зависят от установленного разрешения, которое, в свою очередь,

определяется параметрами видеосигнала, поступающего на ЭЛТ. Размер по вертикали определяется количеством строк растра, а по горизонтали — *максимальной частотой спектра* видеосигнала (как известно, любой сложный сигнал может быть представлен как сумма простейших синусоидальных колебаний разной частоты и амплитуды — это и есть спектр данного сигнала). Именно высокочастотные спектральные составляющие видеосигнала определяют размер мелких деталей на экране монитора.

Оценим взаимосвязь между разрешением монитора, параметрами развертки и характеристиками видеосигнала. Для этого воспользуемся изображением в виде шахматного поля (рис. 1.5). Предположим, что разрешение этого изображения составляет $N \times M$, где N— количество пикселов (клеток) в строке (горизонтальное разрешение), а M— количество строк (вертикальное разрешение). Частоту строк обозначим $f_{стр}$, а частоту кадров — $f_{кадр}$.

Размер пиксела по горизонтали определяет минимальный период видеосигнала $T_{мин}$. Величина, обратная $T_{мин}$, т. е. количество периодов видеосигнала в единицу времени, является максимальной частотой спектра видеосигнала f_{max}. Один кадр изображения содержит $N \times M$ пикселов. Если частота кадров $f_{кадр}$, то участок видеосигнала длительностью 1 с должен содержать $f_{кадр} \times N \times M$ периодов, т. е. максимальная частота

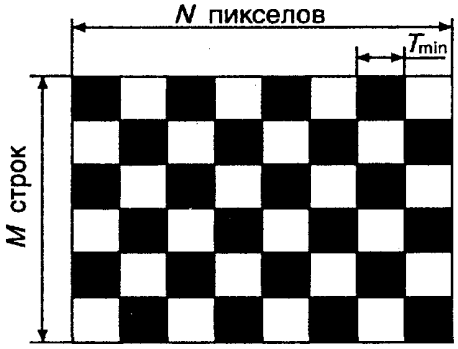


Рис. 1.5. Изображение в виде шахматного поля

спектра видеосигнала при разрешении $N \times M$ и частоте кадров $f_{кадр}$

$$f_{max} = f_{кадр} \times N \times M \text{ Гц}.$$

Например, при разрешении 640×480 и частоте кадров 60 Гц максимальная частота спектра видеосигнала составит около 20 МГц, а при разрешении 1024×768 и частоте кадров 75 Гц — уже 59 МГц.

Анализируя полученное соотношение, можно сделать три вывода:

1. Повышение разрешения изображения при неизменной частоте кадров сопровождается расширением спектра видеосигнала, т. е. увеличением его максимальной частоты спектра f_{max}.

2. Максимальная частота спектра видеосигнала определяет максимально возможное при заданной частоте кадров разрешение. На практике эта частота не может быть больше, чем *ширина полосы пропускания видеотракта монитора.* Напомним, что полосой пропускания устройства называют диапазон частот входного сигнала, в пределах которого коэффициент передачи данного устройства по напряжению составляет не менее 0,707 от максимального (рис. 1.6). Зависимость коэффициента передачи устройства от частоты сигнала называют *амплитудно-частотной характеристикой* (АЧХ).

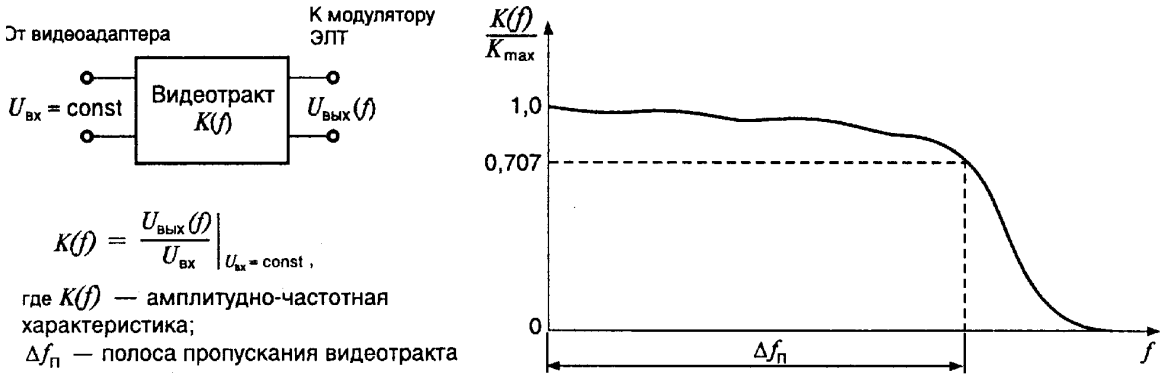


Рис. 1.6. Типичная АЧХ видеотракта монитора

Если максимальная частота (т. е. ширина) спектра видеосигнала превысит ширину полосы пропускания видеотракта монитора, то видеосигнал не сможет пройти через этот тракт без искажений. Повысить разрешение изображения при сохранении значения f_{max} *неизменным* можно только при снижении частоты кадров, что сделает мерцание изображения более заметным. Например, ширина полосы пропускания видеотракта стандартного монитора видеосистемы VGA (см. разд. 1.3) составляет около 25 МГц, что позволяет ему поддерживать разрешение 640×480 при частоте кадров 60 Гц или 720×400 при частоте кадров

70 Гц. Для работы с разрешением 800×600 необходимо снизить частоту кадров до 56 Гц, что приведет к появлению заметного мерцания экрана. Сохранение прежней частоты кадров при таком разрешении невозможно, поскольку максимальная частота спектра видеосигнала окажется за пределами полосы пропускания видеотракта монитора.

3. Для повышения разрешения изображения по вертикали при заданной частоте кадров необходимо не только расширить полосу пропускания видеотракта, но и увеличить частоту строк $f_{стр} = f_{кадр} \times M$. Даже в том случае, когда f_{max} попадает в полосу пропускания видеотракта монитора, изображение на его экране может быть неустойчивым или пропасть вовсе, если генератор строчной развертки монитора не в состоянии обеспечить требуемую частоту строк $f_{стр}$ (с учетом импульсов гашения требуемое значение частоты строк оказывается еще выше примерно на 10%). В частности, первым монитором, поддерживающим разрешение 1024×768, был монитор 8514 фирмы IBM. Он имел достаточно широкую полосу пропускания видеотракта (45—47 МГц), чтобы обеспечить при данном разрешении и построчной развертке частоту кадров около 56 Гц. Однако предельная частота генератора строчной развертки у этого монитора не превышала 36 КГц, тогда как для обеспечения указанной частоты кадров требовалось как минимум 48 КГц. В результате обеспечить разрешение 1024×768 этот монитор мог только при чересстрочной развертке с частотой кадров 43,8 Гц (частотой полей около 88 Гц) и частотой строк 35 КГц.

Таким образом, максимальное разрешение монитора определяется не только размером зерна люминофора, но и шириной полосы пропускания видеотракта и максимальной частотой строчной развертки. Значения всех этих характеристик монитора должны быть согласованы друг с другом, т. е. повышение разрешения возможно только при одновременном их улучшении.

1.2.1.4. Количество отображаемых цветов

Качество цветного изображения на экране монитора определяется количеством цветов, которые он в состоянии воспроизвести.

Отметим одну терминологическую тонкость. Если количество цветов невелико (до 16), то будем говорить о *цветах.* При этом будем иметь в виду не только три основных цвета (красный, зеленый и синий), но и получаемые путем их смешения *неосновные цвета.* Если же количество цветов составляет сотни, тысячи и даже миллионы, то правильнее говорить об *оттенках цветов.* Иногда вместо термина «количество цветов» (оттенков) употребляют термины *палитра* или *размер палитры.*

Далее, если конкретное количество *оттенков цветов* не оговаривается, для краткости будем использовать термин «количество цветов».

Количество цветов, формируемых монитором, определяется типом видеосигнала — цифровой он или аналоговый.

Цифровой видеосигнал может принимать только два значения: высокое (уровень логической единицы) и низкое (уровень логического нуля) (рис. 1.7, *а).*

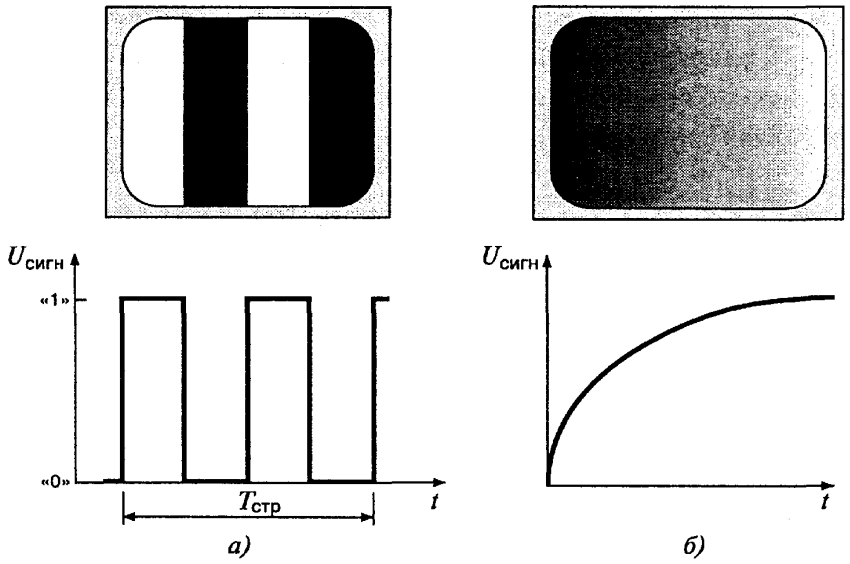


Рис. 1.7. Цифровой *(а)* и аналоговый *(б)* видеосигналы

Если на модулятор электронной пушки поступает видеосигнал высокого уровня, то формируется луч номинальной интенсивности, а если низкого — пушка заперта. Несложно подсчитать, что возможное число различных состояний трех пушек составляет $2^3 = 8$, т. е. монитор может сформировать только 8 цветов. Другими словами, смешение основных цветов производится с использованием только целых весовых коэффициентов — либо 0, либо 1.

Мониторы, управление которыми осуществляется цифровым видеосигналом, называют *цифровыми* или

TTL-мониторами. Последнее название объясняется тем, что аналогичные уровни логического нуля и логической единицы используются в цифровых схемах на основе транзисторно-транзисторной логики (TTL — *Transistor-Transistor Logic).*

Главный недостаток цифровых мониторов — скудная палитра. Для увеличения количества отображаемых цветов, помимо трех основных сигналов (R, G и B), используются дополнительные линии. Так, в мониторе видеосистемы CGA используется дополнительный модулятор яркости *(I — Intensity),* общий для всех трех пушек (см. разд. 1.3). В общем случае, чтобы получить палитру размером N цветов, необходимо $\log_2 N$ проводов для передачи двоичного кода цвета. Кроме того, необходимо изменить конструкцию модулятора каждой электронной пушки таким образом, чтобы ею можно было управлять не только основным, но и дополнительным сигналом.

Очевидно, что значительно увеличить палитру таким способом невозможно. Ведь соединительный кабель между монитором и видеоадаптером может вместить лишь ограниченное количество проводов. Максимальное количество (6 проводов) использовалось в мониторах, разработанных фирмой IBM для видеосистемы EGA (см. разд. 1.3). Каждая электронная пушка этого монитора имела два модулятора: основной (R, G, B) и дополнительный (r, g, b). Размер палитры монитора составлял $2^6 = 64$ оттенка цвета. В настоящее время цифровые мониторы практически не используются.

Сделать палитру монитора неограниченной можно, если подавать на ЭЛТ *аналоговые* (непрерывные) видеосигналы (рис. .1.7, б).

Мониторы, управление которыми осуществляется непрерывным видеосигналом, называются *аналоговыми.* В настоящее время именно они используются в составе PC.

Принцип действия аналогового монитора такой же, как и цветного телевизора. При изменении аналоговых видеосигналов на модуляторах ЭЛТ интенсивность электронных лучей будет изменяться плавно. Также плавно, а не дискретно (как в цифровых мониторах), будет изменяться яркость свечения соответствующих зерен люминофора. Это позволит выполнять смешение трех основных цветов в любой пропорции. Поскольку множество значений непрерывной величины бесконечно, бесконечным будет и число возможных сочетаний яркости трех основных цветов. Поэтому цветовая палитра аналоговых мониторов не ограничена, т. е. с помощью таких мониторов можно получить практически любой оттенок цвета.

На практике палитра видеосистемы PC достаточно велика, но все же ограничена (до 16,7 млн оттенков). Это обусловлено не свойствами монитора, а особенностями работы видеоадаптера, формирующего видеосигнал.

1.2.2. Формирование видеосигнала

Видеосигнал, подаваемый на модуляторы ЭЛТ, а также импульсы строчной и кадровой синхронизации, подаваемые в блок разверток монитора, формируются вторым элементом видеосистемы — видеоадаптером — на основе цифрового образа формируемого изображения — матрицы $N \times M$ n-разрядных чисел. Эти данные заносятся центральным процессором PC в память видеоадаптера, или *видеопамять.* Участок видеопамяти, отведенный для хранения цифрового образа текущего изображения (кадра), называется *кадровым буфером* или *фрейм-буфером* (от *англ. frame buffer*— кадровый буфер). Видеоадаптер последовательно считывает (сканирует) содержимое ячеек кадрового буфера и формирует на своем выходе видеосигнал, уровень которого пропорционален значениям, хранящимся в каждой из ячеек. Сканирование видеопамяти осуществляется синхронно с перемещением электронного луча по экрану ЭЛТ. В результате яркость каждого пиксела на экране монитора оказывается пропорциональной значению, содержащемуся в соответствующей ячейке памяти видеоадаптера. В начале и по окончании просмотра ячеек, соответствующих одной строке растра, видеоадаптер формирует строчные импульсы гашения и на отдельном выходе — импульсы строчной синхронизации. По окончании сканирования кадрового буфера формируются кадровые гасящие и синхронизирующие импульсы соответственно. Таким образом, частоты строчной и кадровой разверток монитора определяются скоростью сканирования содержимого видеопамяти. Очевидно, что блок разверток монитора должен поддерживать эти частоты. В противном случае изображение на экране монитора будет нестабильным или вовсе отсутствовать.

В зависимости от режима работы видеоадаптера видеосистема PC может обеспечивать различные разрешение и палитру. Совокупность параметров, характеризующих режим работы видеоадаптера (разрешение, палитра, частоты строчной и кадровой разверток, способ адресации участков экрана и др.), будем называть *видеорежимом.* Если указаны только некоторые параметры, например разрешение, то будем говорить просто о *режиме* работы, не уточняя значений опущенных параметров. При таком подходе термин «режим» может включать в себя несколько различных видеорежимов с одним общим параметром.

Все видеорежимы делятся на *графические* и *текстовые.* Различие в режимах существенно только для видеоадаптеров, поскольку в каждом из них используются различные механизмы формирования видеосигнала. Что же касается монитора, то в обоих режимах он работает одинаково.

Исторически первым в видеосистеме PC использовался текстовый режим, так как его реализация требовала меньших затрат по сравнению с графическим. Однако принцип работы видеоадаптера в графическом режиме проще, чем в текстовом, поэтому описание этих режимов начнем с описания графического режима.

1.2.2.1. Графический режим

Графический режим является основным режимом работы видеосистемы мультимедийного PC. В этом режиме на экран монитора может быть выведено произвольное изображение: текст, рисунок, фотография, анимация или видеосюжет. В частности, в таком режиме работает видеосистема PC под управлением операционной системы Windows. Эта операционная система предлагает пользователю удобный и наглядный графический интерфейс, благодаря которому работа на компьютере доставляет истинное удовольствие.

Однако для эффективной работы в графическом режиме требуется значительный объем видеопамяти и высокопроизводительный компьютер, поэтому данный режим появился позднее текстового и стал основным только с появлением PC на основе CPU Intel 80386 и 80486.

В графическом режиме содержимое каждой ячейки кадрового буфера (матрицы $N \times M$ n-разрядных чисел) является кодом цвета соответствующего пиксела экрана. Разрешение экрана при этом также равно $N \times M$. Адресуемым элементом экрана является минимальный элемент изображения — пиксел. По этой причине графический режим называют также режимом APA *(All Point Addressable* — все точки адресуемы).

Принцип действия видеоадаптера в графическом режиме проиллюстрируем с помощью рис. 1.5. Предположим, на экране монитора — монохромное изображение шахматного поля размером $N \times M$ клеток, причем каждая клетка представляет собой один пиксел изображения. Содержимое кадрового буфера, соответствующее такому изображению, будет представлять собой матрицу $N \times M$ одноразрядных двоичных чисел. Ячейка, соответствующая светлому пикселу экрана, будет содержать единицу, а черному — ноль. В процессе считывания содержимого ячеек видеопамяти видеоадаптер сформирует двухуровневый (двоичный) видеосигнал.

Если изображение цветное, то для кодирования цвета каждого пиксела отводится не один, а n двоичных разрядов. Иногда число n называют *глубиной цвета или глубиной пиксела.* При этом количество одновременно отображаемых цветов равно T, а размер кадрового буфера, необходимый для хранения цветного изображения с разрешением $N \times M$ и глубиной цвета (пиксела) n, составит $N \times M \times n$ бит. В частности, для хранения изображения с разрешением 800×600 и глубиной цвета 24 бит (2^{24}=16,7 млн цветов) необходимо уже 800×600×24 = 11520000 бит = 1,37 Мбайт памяти. Это значит, что видеоадаптер должен иметь как минимум 2 Мбайт видеопамяти. Именно объем видеопамяти определяет размер цветовой палитры изображения.

Частота, с которой происходит опрос ячеек кадрового буфера, содержащих коды цвета пикселов, определяет скорость прорисовки изображения на экране монитора. Эта частота называется *dot rate* или *pixel rate* — скорость вывода пикселов на экран. Очевидно, что она равна верхней граничной частоте видеосигнала. В видеоадаптере имеется специальный генератор, формирующий сигнал этой частоты, — *dot clock.* Современные видеоадаптеры содержат несколько таких генераторов, причем частота генерируемых сигналов может быть изменена программным путем.

1.2.2.2. Текстовый режим

В текстовом (символьном) режиме, так же, как и в графическом, изображение на экране монитора представляет собой множество пикселов и характеризуется разрешением $N \times M$. Однако все пикселы разбиты на группы, называемые *знакоместами* или *символьными позициями (character positions* или *character boxes* — символьные ячейки) размером $p \times q$. В каждом из знакомест может быть отображен один из 256 символов. Таким образом, на экране умещается $M/q = M_t$ символьных строк по $N/p = N_t$ символов в каждой (рис. 1.8, *а*). Типичным текстовым режимом является режим 80×25 символов.

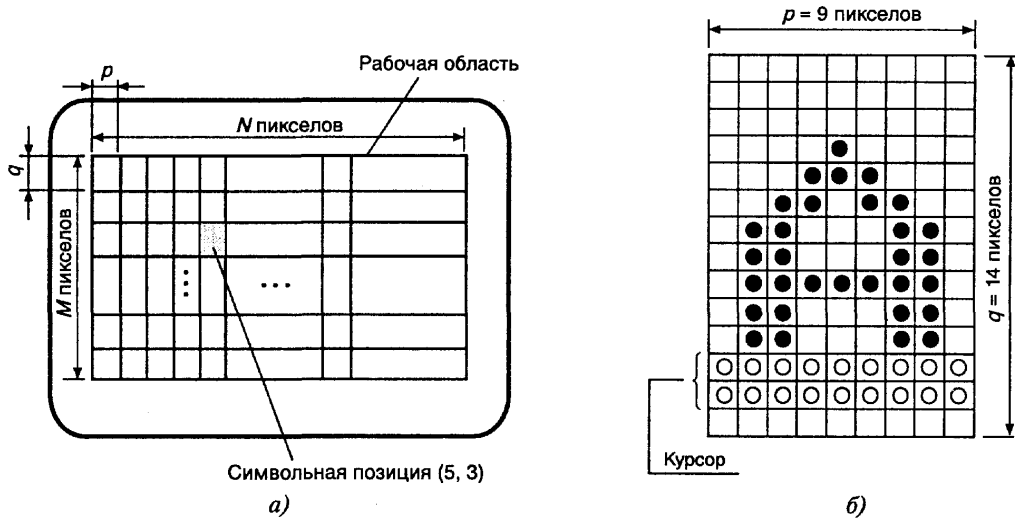
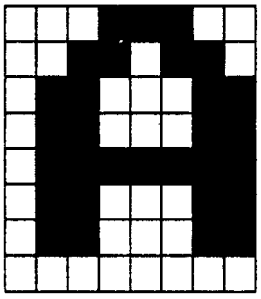


Рис. 1.8. Разбиение экрана на знакоместа в текстовом режиме работы видеосистемы

Изображение символа в пределах каждого знакоместа задается точечной матрицей *(dot matrix).* Размер матрицы зависит от типа видеоадаптера и текущего видеорежима. Чем больше точек используется для отображения символа, тем выше качество изображения и лучше читаемость текста. Точки матрицы, образующие изображение символа, называют передним планом *(foreground),* остальные — задним планом или фоном *(background).* В качестве примера на рис. 1.8, *б* изображена символьная матрица 9×14 пикселов.

Если считать, что темной клетке соответствует логическая единица, а светлой — логический ноль, то каждую строку символьной матрицы можно представить в виде двоичного числа (рис. 1.9). Следовательно, графическое изображение символа можно хранить в виде последовательности двоичных чисел. Для этой цели используется специальное ПЗУ, размещенное на плате видеоадаптера. Такое ПЗУ называется *аппаратным знакогенератором (hardware character generator),* а совокупность изображений 256 символов — *шрифтом.* В частности, аппаратный знакогенератор хранит шрифт, который автоматически используется видеоадаптером сразу же после включения компьютера (обычно это буквы английского алфавита и набор специальных символов). В качестве адреса ячейки знакогенератора используется порядковый номер символа.



0	0	0	1	1	1	0	0	1Ch
0	0	1	1	0	1	1	0	36h
0	1	1	0	0	0	1	1	63h
0	1	1	0	0	0	1	1	63h
0	1	1	1	1	1	1	1	7Fh
0	1	1	0	0	0	1	1	63h
0	1	1	0	0	0	1	1	63h
0	0	0	0	0	0	0	0	00h

Рис. 1.9. Кодирование изображения символьной матрицы 8×8 пикселов последовательностью двоичных чисел

В современных видеоадаптерах содержимое знакогенератора копируется в специально отведенную область видеопамяти, в которой затем можно корректировать данные, изменяя тем самым начертание символов. На этом принципе основана технология, позволяющая загружать шрифты национального алфавита. Программа, выполняющая загрузку требуемого шрифта, называется *программным знакогенератором (software character generator).* К таким программам относятся широко распространенные русификаторы, такие как KEYRUS.COM, CFD.EXE, VIDEORUS.EXE и др.

Работа видеоадаптера в текстовом режиме имеет свои особенности:

- на экране монитора нельзя создать произвольное изображение — оно может состоять только из ограниченного набора символов, начертание которых определяется шрифтом используемого знакогенератора;
- поскольку изображение символа известно, нет необходимости хранить его в кадровом буфере целиком — достаточно поместить туда лишь номер символа;
- минимальным адресуемым элементом экрана является не пиксел (как в графическом режиме), а знакоместо, поэтому в текстовом режиме нельзя изменять значение каждого отдельного пиксела.

Как следует из названия, текстовый режим предназначен в основном для отображения текстовой информации. Поскольку количество символов ограничено (256), возможности видеосистемы с точки зрения разнообразия изображения невелики. Обычно это буквы двух алфавитов (английского и национального), цифры, набор символов псевдографики (отрезки различных линий), а также ограниченный набор специальных управляющих символов. Символы нумеруются от 0 до 255. Номер каждого символа представляет собой 8-разрядное двоичное число (1 байт) и обычно записывается не в двоичной, а в шестнадцатеричной системе счисления. Например, порядковый номер английской буквы «S» (83) в двоичном коде запишется как 01010011, а в шестнадцатеричном — 53h. Буква «h» указывает на то, что число записано в шестнадцатеричной системе счисления.

Из 256 символов первые 128 (с номерами 0—127) всегда являются буквами английского алфавита, цифрами и специальными управляющими символами. Их нумерация стандартизирована и неизменна, поэтому английский алфавит всегда доступен на любом компьютере. В частности, в PC широко используются две схемы нумерации символов:

- ASCII *(American Standard Code for Information Interpretation)* — американский стандартный код; используется в операционной системе MS DOS;
- ANSI *(American National Standardization Institute)* — код, разработанной американским национальным институтом стандартизации для представления символьной информации в среде MS Windows.

Вторая половина символьного набора (с номерами 128—255) отводится под буквы национального

алфавита и является уникальной для каждой страны. Совокупность всех символов для конкретной страны называется *кодовой страницей.* Каждой кодовой странице фирма Microsoft присвоила уникальный номер. Например, кодовая страница России имеет номер 866 для DOS (ASCII) и 1251 для Windows (ANSI).

Для описания изображения на экране в пределах каждого знакоместа в каждом буфере отводится по два байта. Байт с четным адресом содержит *код символа,* т. е. его порядковый номер, второй байт, с нечетным большим адресом, — код *атрибутов символа,* т. е. информацию о способе представления символа: цвет и яркость символа и фона, подчеркивание или мерцание символа, наличие курсора и др. Обычно информация, содержащаяся в первых (младших) четырех разрядах байта атрибутов, описывает собственно символ, т. е. передний план символьной матрицы *(foreground),* а в четырех старших разрядах — задний план, т. е. параметры фона.

Поскольку для описания атрибутов символа используется всего 4 разряда, в текстовом режиме одновременно может быть отображено не более 2^4=16 цветов.

Если на экране имеется $N_t \times M_t$ знакомест, то объем видеопамяти, необходимый для хранения изображения, составит $N_t \times M_t \times 2$ байт. Эту область видеопамяти называют *видеостраницей (video page).* Видеостраница является аналогом кадрового буфера в графическом режиме, но имеет значительно меньший объем. В наиболее распространенном текстовом режиме 80×25 символов размер видеостраницы составляет 4000 байт, в режиме 40×25 — 2000 байт. На практике для удобства адресации в режимах 80×25 и 40×25 под видеостраницу отводится целое число килобайт: 4 Кбайт = 4096 байт и 2 Кбайт = 2048 байт соответственно, при этом «лишние» байты (96 и 48) не используются.

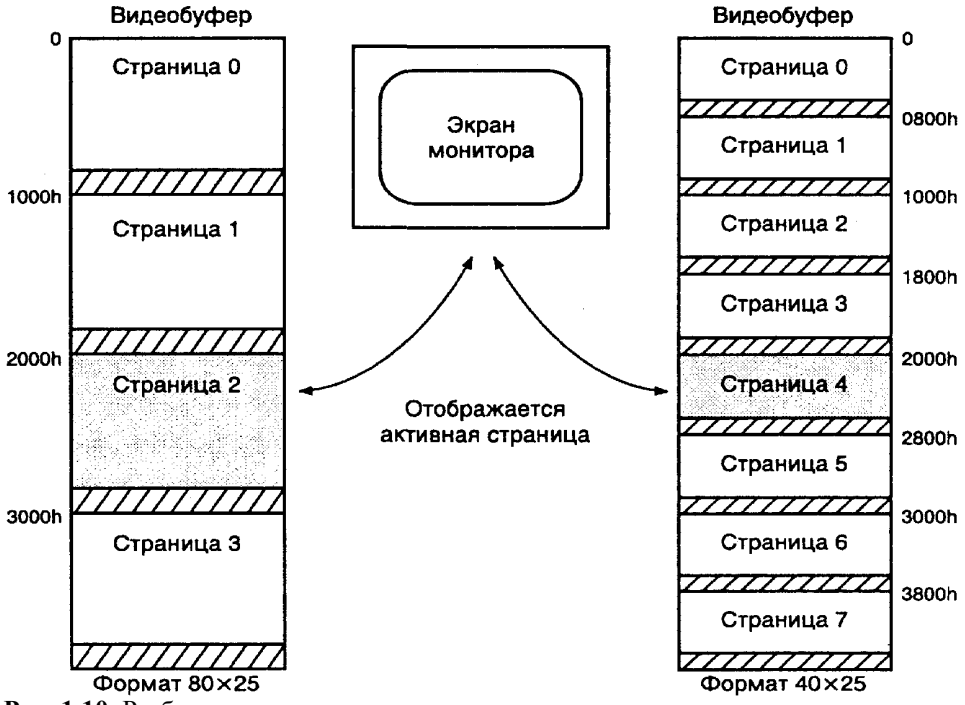


Рис. 1.10. Разбиение видеопамяти на видеостраницы

Поскольку размер видеостраницы обычно намного меньше емкости видеопамяти, в ней одновременно размещают несколько видеостраниц (рис. 1.10). Видеостраница, содержимое которой в данный момент выводится на экран, называется *активной* (используются также синонимы *текущая* и *визуальная страница).*

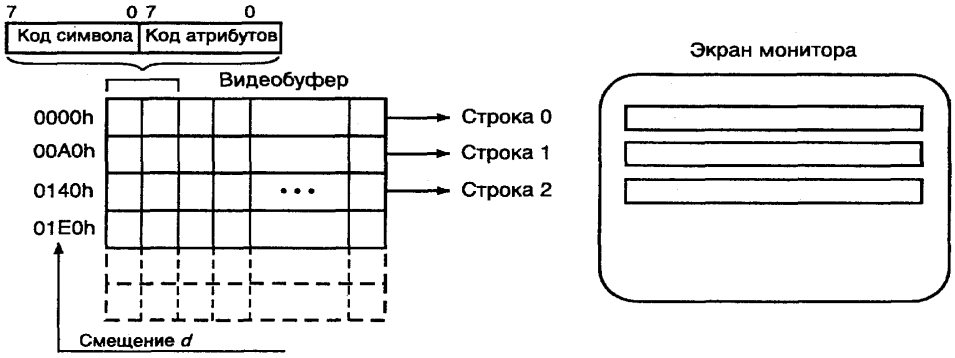


Рис. 1.11. Отображение активной видеостраницы на экран монитора в текстовом режиме 80×25 символов

Многостраничный механизм позволяет центральному процессору заранее обновлять содержимое неактивных видеостраниц, т.е. - пока видеоадаптер формирует изображение активной страницы.

Для обновления экрана достаточно переключить активную страницу — смена изображения произойдет практически мгновенно. Видеосигнал в текстовом режиме формируется так же, как и в графическом: видеоадаптер последовательно опрашивает ячейки памяти активной видеостраницы. Адрес ячейки, соответствующей знакоместу экрана с координатами *(x, y),* задается *смещением d,* которое определяется как

$$d = (x + y \times n) \times 2,$$

где x — номер символа в строке (номер столбца); y — номер строки; n — количество символов в строке.

Другими словами, используется принцип линейного отображения содержимого видеостраницы на экран (рис. 1.11).

На рис. 1.12 представлена структурная схема, а на рис. 1.13 - механизм формирования видеосигнала в текстовом режиме и кодировка символов в знакогенераторе.

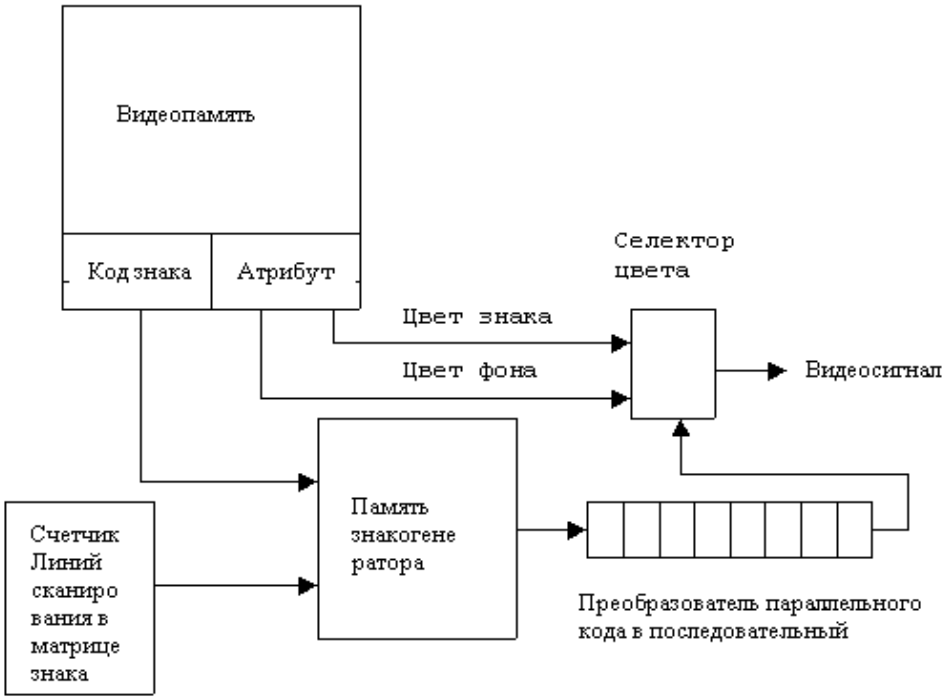


Рис. 1.12. Структурная схема формирования видеосигнала в текстовом режиме

Содержащиеся в видеопамяти числа являются кодами символов и одновременно адресами ячеек ПЗУ знакогенератора, в которых содержатся символьные матрицы $p \times q$ пикселов (нулем в знакогенераторе кодируется цвет фона, а единицей - цвет знака). Считав коды символов одной строки, видеоадаптер формирует q периодов строчной развертки. В течение i-го периода, где i = 1,2, .., q, видеоадаптер последовательно считывает i-ю строку каждой символьной матрицы и преобразует ее содержимое в видеосигнал, поступающий на монитор. Считывание содержимого символьной матрицы осуществляется с частотой *pixel rate* (см. графический режим). При этом учитываются атрибуты каждого знакоместа.

По окончании формирования видеосигнала одной символьной строки видеоадаптер считывает коды символов следующей строки и т. д. Механизм формирования изображения в текстовом режиме на примере символьной матрицы 8x8 пикселов изображен на рис. 1.33.

Таким образом, для работы видеоадаптера в текстовом режиме требуется значительно меньше ресурсов, нежели в графическом, а именно:

- для кадрового буфера необходим небольшой объем видеопамяти; например, если в графическом режиме 640×480 при 256 оттенках цветов необходимо 300 Кбайт видеопамяти (видеоадаптер должен иметь, как минимум, 512 Кбайт видеопамяти), то в текстовом режиме 80×25 символов, в котором используется близкое к упомянутому разрешение экрана 720×400, активная видеостраница занимает всего лишь 4 Кбайт;
- резко снижается загрузка центрального процессора и шины ввода/вывода за счет значительного сокращения объема пересылаемых данных;
- быстрое обновление экрана осуществляется путем изменения номера активной видеостраницы.

Низкие затраты ресурсов PC на реализацию текстового режима были существенным преимуществом последнего в тот период, когда общая производительность PC была невысока. Именно по этой причине

видеосистема первого персонального компьютера IBM PC могла работать только в текстовом режиме. Легендарная программная оболочка Norton Commander для MS DOS также работает в текстовом режиме, благодаря чему затраты ресурсов PC на ее функционирование минимальны. Напротив, операционная среда Windows работала на PC первых поколений (286, 386) очень медленно, так как центральный процессор был в основном занят решением задачи создания графического интерфейса пользователя, т. е. заполнением внушительного по размеру (около 1 Мбайт) кадрового буфера.

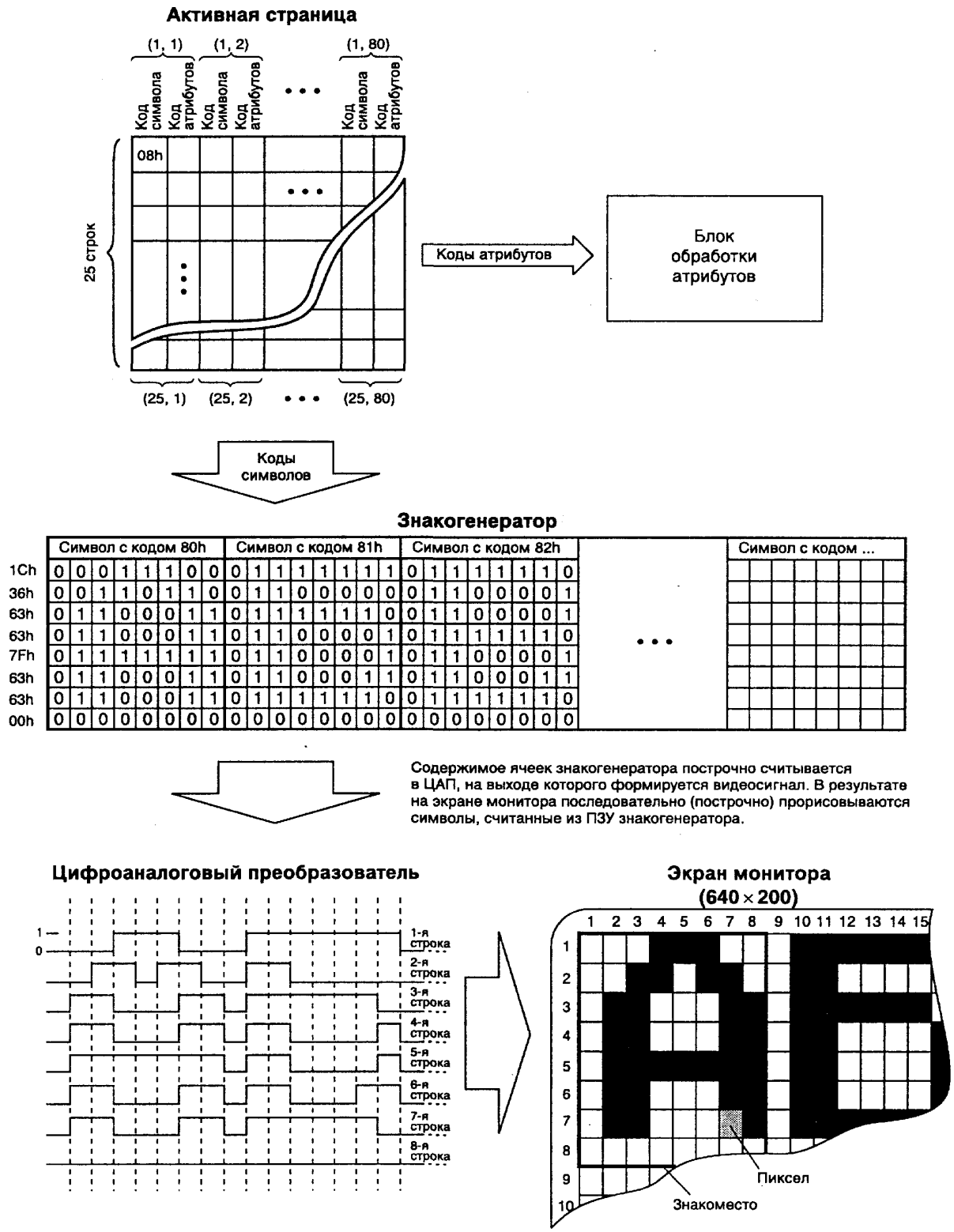


Рис. 1.13. Механизм формирования видеосигнала в текстовом режиме

В настоящее время производительность CPU и видеоадаптера, а также пропускная способность шины ввода/вывода PC настолько возросли, что работа в графическом режиме практически не сказывается на быстродействии компьютера. В результате текстовый режим работы видеоадаптера стал для

мультимедийного PC вспомогательным, а графический, обладающий гораздо большими возможностями, — основным. В современном PC видеосистема работает в текстовом режиме лишь на стадии инициализации и загрузки операционной системы, а также в режиме эмуляции MS DOS (в Windows 95/98).

Программирование адаптеров для работы в определенных режимах работы осуществляет BIOS видеосистемы через прерывание Int 10h.

Выдача символов на экран осуществляется через функции BIOS с помощью функции 09h и условно показана на рис. 1.14.

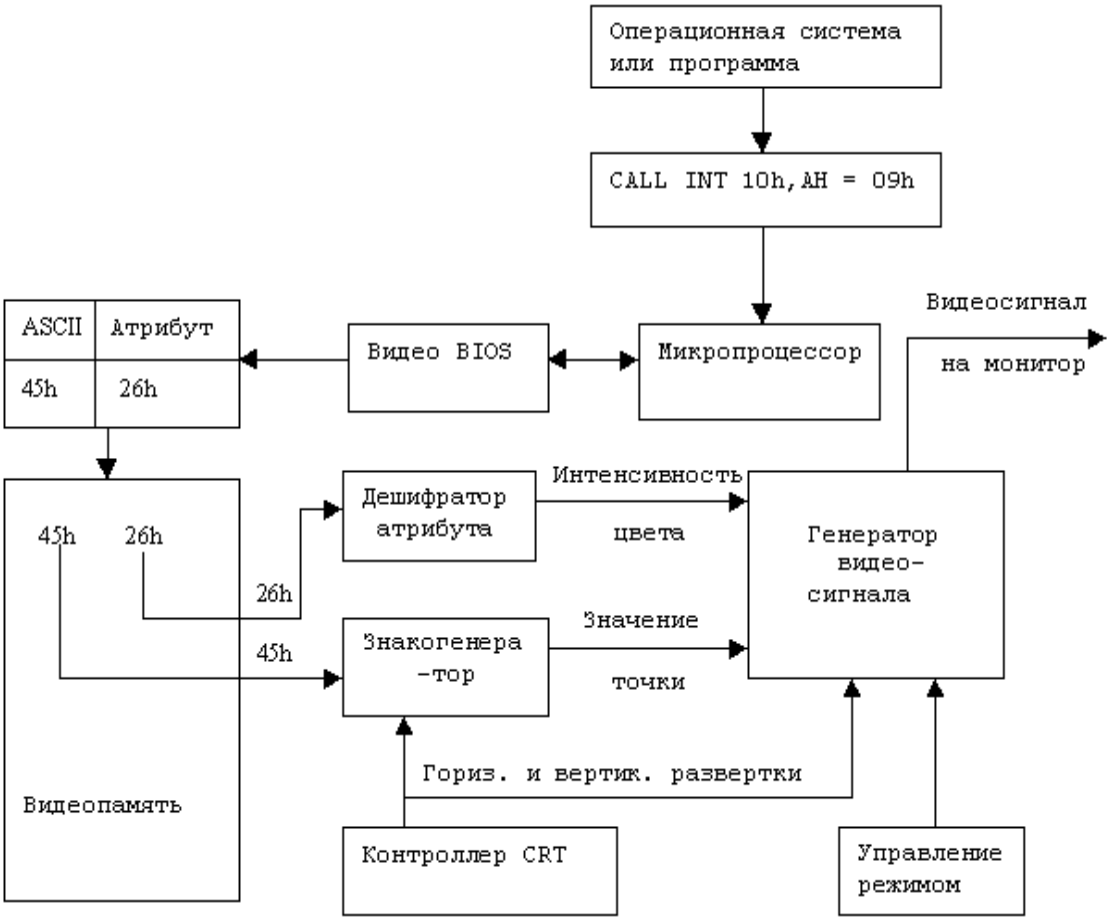


Рис. 1.14. Структурная схема выдача символов на экран

1.3. Эволюция видеосистемы PC

Выпуск фирмой IBM в 1981 г. первого персонального компьютера типа IBM PC ознаменовал собой начало массового производства компьютеров, обладающих удобным и наглядным интерфейсом. Эти компьютеры были более комфортны в работе, чем имевшиеся в то время большие ЭВМ, так как обеспечивали компактное размещение всех подсистем «под рукой» у пользователя. Особенно удобным было наличие собственной видеосистемы, позволяющей реализовать оперативный диалог пользователя с компьютером. Качество изображения, создаваемого видеосистемой PC, было значительно выше, чем изображение на экране дисплея больших ЭВМ. С момента появления и до наших дней у PC сменилось несколько типов видеосистем. Рассмотрим основные из них, поскольку именно они оказали наибольшее влияние на облик видеосистемы современного мультимедийного PC.

1.3.1. MDA

Первые модели IBM PC были оснащены монохромным дисплеем с люминофором зеленого свечения. Для связи этого дисплея с компьютером использовался видеоадаптер типа MDA *(Monochrome Display Adapter* — адаптер монохромного дисплея). Он работал только в текстовом режиме 80×25 символов. Размер символьной матрицы (знакоместа) составлял 9×14 пикселов, поэтому разрешение, поддерживаемое монитором MDA, составляло 720×350 пикселов. Размер самого символа составлял 7×9 пикселов. Емкость видеопамяти видеоадаптера MDA была минимальной и достаточной для размещения только одной видеостраницы размером 4 Кбайт.

Основным функциональным узлом адаптера является контроллер CRT, который реализован на БИС Motorola 6845 или совместимом контроллере.

BIOS MDA находится в системной ROM на материнской плате (не устанавливается на адаптерной плате).

Порты ввода-вывода MDA и соответствующие регистры приведены в табл. 1.1.

Программирование контроллера 6845 осуществляется через индексный регистр 3B4h. Данные в регистры записываются по адресу 3B5h.

Таблица. 1.1. Порты ввода-вывода MDA

Адрес ввода-вывода	Чтение/ запись	Индекс	Назначение
3B4h	Чтение/ запись	-	Индексный регистр
3B5h	Запись	00h	Общая по горизонтали (Horizontal Total)
3B5h	Запись	01h	Конец отображения по горизонтали (Horizontal Displayed)
3B5h	Запись	02h	Начало горизонтального бланкирования (Horizontal Sync Position)
3B5h	Запись	03h	Конец горизонтального бланкирования (Horizontal Sync Pulse Width)
3B5h	Запись	04h	Начало обратного хода по горизонтали (Vertical Total)
3B5h	Запись	05h	Конец обратного хода по горизонтали (Vertical Total Adjust)
3B5h	Запись	06h	Общая по вертикали (Vertical Displayed)
3B5h	Запись	07h	Дополнение (Vertical Sync Position)
3B5h	Запись	08h	Предустановка линии сканирования (Interlace Mode)
3B5h	Запись	09h	Максимальная линия сканирования Maximum Scan Line
3B5h	Запись	0Ah	Начальная позиция курсора (Cursor Start)
3B5h	Запись	0Bh	Конечная позиция курсора (Cursor End)
3B5h	Запись	0Ch	Начальный адрес (старший байт) (Start Address Height)
3B5h	Запись	0Dh	Начальный адрес (младший байт) (Start Address Low)
3B5h	Чтение/ запись	0Eh	Адрес курсора (старший байт) (Cursor Location Height)
3B5h	Чтение/ запись	0Fh	Адрес курсора (младший байт) (Cursor Location Low)
3B5h	Чтение/ запись	10h	Световое перо (старший байт) (Light Pen Height)
3B5h	Чтение/ запись	11h	Световое перо (младший байт) (Light Pen Low)
3B8h	Чтение/ запись	-	Регистр управления режимом (Mode Control). [0] - бит включения адаптера, [3] - бит включения видеосигнала, [5] - бит включения атрибута мерцания.
3BAh	Чтение	-	Регистр текущего состояния адаптера (Input Status). [0] - идентификатор обратного хода луча, [3] - бит видеосигнала.

Видеосистема MDA обеспечивала довольно хорошую читаемость символов на экране монитора и вполне соответствовала требованиям приложений, связанных с обработкой текстовой информации, однако имела и два серьезных недостатка:

- видеосистема формировала только монохромное изображение, тогда как более естественным и информативным является цветное изображение;
- многие приложения требовали вывода на экран не только текстовой, но и графической информации, т. е. работа с этими приложениями должна была осуществляться в графическом режиме, который видеосистема MDA не поддерживала.

С целью устранения недостатков видеосистемы MDA буквально через несколько месяцев после выпуска IBM PC фирма IBM предложила новую видеосистему — CGA, обладающую более широкими возможностями, но уже в составе улучшенной модели компьютера — IBM PC XT *(eXtended Technology).*

1.3.2. CGA

Видеосистема CGA включала в себя цветной TTL-монитор (цифровой) и видеоадаптер CGA *(Color Graphics Adapter* — цветной графический адаптер). Главные отличия этой видеосистемы от видеосистемы MDA отражены в ее названии, т. е. она обеспечивала:

- цветное изображение (от 4 до 16 цветов);
- несколько графических режимов работы видеоадаптера.

Максимальное разрешение монитора CGA составляло 640×200. Такое разрешение использовалось либо в текстовом режиме 80×25 (при размере знакоместа 8×8), либо в монохромном графическом режиме. В последнем случае для хранения цифрового образа экрана требовался кадровый буфер размером 640×480×1 = 128000 бит = 15,625 Кбайт. Поэтому объем видеопамяти видеоадаптера CGA составлял 16 Кбайт.

При работе в графическом режиме с более низким разрешением, например 320×200, для кодирования цвета каждого пиксела использовалось 2 бита, благодаря чему обеспечивалось одновременное отображение 4 цветов, а при разрешении 200×160 -16 цветов. В текстовом режиме были доступны все 16 цветов.

Поскольку разрешение монитора CGA по сравнению с монитором MDA было снижено (640×200 против 720×350), пришлось уменьшить и размер знакоместа (8×8 вместо 9×14), чтобы в текстовом режиме уместить на экране все те же 80×25 = 2000 символов. Непосредственно изображение символа формировалось матрицей 7×7 пикселов. Меньшая детальность прорисовки символа и малое межсимвольное расстояние заметно ухудшили различимость текста, поэтому качество изображения в текстовом режиме работы видеосистемы CGA было ниже, чем у MDA. Главными достоинствами новой видеосистемы были возможность работы в графическом режиме и возможность формирования цветного изображения.

Основным функциональным узлом адаптера являлся контроллер CRT, который, как и MDA был реализован на БИС Motorola 6845 или совместимом контроллере.

BIOS CGA находится в системной ROM на материнской плате (не устанавливается на адаптерной плате).

Порты ввода-вывода CGA и соответствующие регистры приведены в табл. 1.2.

Таблица. 1.2. Порты ввода-вывода CGA

Адрес ввода-вывода	Чтение/ запись	Индекс	Назначение
3D4h	Чтение/ запись	-	Индексный регистр (регистр адреса), служит для адресации всех остальных регистров контроллера
3D5h	Запись	00h	Общая по горизонтали (Horizontal Total)
3D5h	Запись	01h	Конец отображения по горизонтали (Horizontal Displayed)
3D5h	Запись	02h	Начало горизонтального бланкирования (Horizontal Sync Position)
3D5h	Запись	03h	Конец горизонтального бланкирования (Horizontal Sync Pulse
3D5h	Запись	04h	Начало обратного хода по горизонтали (Vertical Total)
3D5h	Запись	05h	Конец обратного хода по горизонтали (Vertical Total Adjust)
3D5h	Запись	06h	Общая по вертикали (Vertical Displayed)
3D5h	Запись	07h	Дополнение (Vertical Sync Position)
3D5h	Запись	08h	Предустановка линии сканирования (Interlace Mode)
3D5h	Запись	09h	Максимальная линия сканирования Maximum Scan Line
3D5h	Запись	0Ah	Начальная позиция курсора (Cursor Start)
3D5h	Запись	0Bh	Конечная позиция курсора (Cursor End)
3D5h	Запись	0Ch	Начальный адрес (старший байт) (Start Address Height)
3D5h	Запись	0Dh	Начальный адрес (младший байт) (Start Address Low)
3D5h	Чтение/ запись	0Eh	Адрес курсора (старший байт) (Cursor Location Height)
3D5h	Чтение/ запись	0Fh	Адрес курсора (младший байт) (Cursor Location Low)
3D5h	Чтение/ запись	10h	Световое перо (старший байт) (Light Pen Height)
3D5h	Чтение/ запись	11h	Световое перо (младший байт) (Light Pen Low)
3D8h	Чтение/ запись	-	Регистр управления режимом (Mode Control): [0] - ширина экрана в текстовом режиме: 1 - 80 символов, 0 - 40 символов. [1] - ширина экрана в графическом режиме: 1 - 320 пикселов, 0 - другая. [2] - бит отключения составляющей интенсивности цвета; [3] - бит включения видеосигнала; [4] - ширина экрана в графическом режиме: 1 - 640 пикселов, 0 - другая; [5] - бит включения атрибута мерцания; [6, 7] - резерв.

Таблица. 1.2. Порты ввода-вывода CGA (продолжение)

3D9h	Чтение/ запись	-	Регистр палитры (Color Select); [0-3] - цвет изображения в режиме наивысшего разрешения (640×200); [4] - бит переключения цвета; [5, 7] - резерв.
3DAh	Чтение	-	Регистр текущего состояния адаптера (Input Status): [0] - текущее состояние видеосигнала: 1 - гашение сигнала, 0 - сигнал активен; [1] - включение светового пера [2] - текущее состояние кнопки светового пера: 0 - кнопка нажата, 1 - кнопка отжата; [3] - вертикальный обратный ход луча; [4, 5] - биты диагностики; [6, 7] - резерв.

Монитор CGA был цифровым, т. е. на модуляторы ЭЛТ подавались двоичные сигналы. ЭЛТ этого монитора имела три электронные пушки, на модуляторы которых подавались двоичные сигналы *RGB,* а также дополнительный общий модулятор, на который подавался двоичный сигнал яркости *I (Intensity).* Способ формирования цвета, реализованный в мониторах CGA, называют IRGB. В табл. 1.3 приведены номера и схемы формирования всех 16 цветов.

Полностью цветовая палитра использовалась видеоадаптером либо в текстовом режиме, либо в графическом с минимальным разрешением 200×160.

Таблица 1.3. Кодирование цветов по схеме IRGB

Номер цвета	Цвет	Способ получения	Значение видеосигнала			
			I	R	G	B
0	Черный (Black)	—	0	0	0	0
1	Синий (Blue)	B	0	0	0	1
2	Зеленый (Green)	G	0	0	1	0
3	Голубой (Cyan)	G+B	0	0	1	1
4	Красный (Red)	R	0.	1	0	0
5	Вишневый (Magenta)	R+B	0	1	0	1
6	Коричневый (Brown)	R+G	0	1	1	0
7	Белый (White)	R+G+B	0	1	1	1
8	Серый (Gray)	I	1	0	0	0
9	Ярко-синий (Light Blue)	I+B	1	0	0	1
10	Ярко-зеленый (Light Green)	I+G	1	0	1	0
11	Ярко-голубой (Light Cyan)	I+B+G	1	0	1	1
12	Ярко-красный (Light Red)	I+R	1	1	0	0
13	Ярко-вишневый (Light Magenta)	I+R+B	1	1	0	1
14	Желтый (Yellow)	I+R+G	1	1	1	0
15	Ярко-белый (Light White)	I+R+G+B	1	1	1	1

Частота строк монитора CGA составляла около 15 КГц, как у бытовых телевизоров, что позволяло использовать обычный телевизор в качестве устройства отображения. Для этого видеоадаптер CGA был оснащен специальным кодирующим устройством, которое из четырех двоичных сигналов *I, R, G, B* и сигналов синхронизации формировало композитный (совмещенный) *полный цветной телевизионный сигнал* (ПЦТС). Этот сигнал мог подаваться на телевизор двумя способами:

- непосредственно, через специальный разъем типа RCA ("колокольчик") (коаксиальный разъем, широко применяемый в аудио- и видеотехнике) видеоадаптера (при этом телевизор должен иметь аналогичный видеовход, иногда называемый низкочастотным видеовходом);
- через антенный вход телевизора (для этого необходим дополнительный блок, преобразующий телевизионный видеосигнал с выхода видеоадаптера в радиочастотный сигнал одного из телевизионных каналов).

Поскольку разрешение экрана телевизора ниже, чем экрана монитора, то и различимость текста на нем значительно хуже. Поэтому в видеоадаптере CGA для работы с телевизором был предусмотрен специальный текстовый режим (40×25 символов), в котором буквы были крупнее и легче читались на экране. Но качество

изображения все равно оставалось весьма низким.

Возможность формирования ПЦТС позволяла использовать совместно с видеоадаптером CGA не только бытовой телевизор, но и специальный *композитный монитор* (он подключался к видеоадаптеру через разъем RCA). Композитные мониторы, ставшие сегодня музейной редкостью, обеспечивали более высокое, чем телевизоры, разрешение, поэтому их можно было использовать в любом текстовом режиме (40×25 и 80×25 символов). Однако вокруг символов на экране такого монитора возникала цветная окантовка, сильно мешающая при работе. Для устранения этого недостатка в видеоадаптере CGA были предусмотрены дополнительные видеорежимы, в которых вместо ПЦТС формировался только сигнал яркости (монохромное изображение), имеющий 16 градаций в текстовом режиме и 4 — в графическом. Однако и композитные мониторы значительно уступали по качеству изображения более современным TTL-мониторам. Поэтому последующие модели видеоадаптеров перестали оснащать схемами формирования композитного видеосигнала.

Впрочем, история повторяется. С началом эры мультимедиа появились так называемые VGA—TV-конверторы, преобразующие сигналы RGB в телевизионный видеосигнал. Однако при этом требуется установка разрешения и частот синхронизации, совпадающих с телевизионным стандартом, что не всегда возможно. Существуют и более дорогие устройства с буферной памятью, формирующие стандартный ПЦТС независимо от режима работы видеоадаптера. Большинство современных видеоадаптеров оснащаются выходом ПЦТС. В этом случае говорят о наличии у него функции TV-out. Очевидно, что данная функция полезна при использовании PC на презентациях, для просмотра видео сюжетов в большой аудитории, для игр и т. п.

Как уже отмечалось, для реализации графического режима необходимо, чтобы все подсистемы компьютера имели довольно высокое быстродействие. В то же время IBM PC XT, в составе которого впервые появилась видеосистема CGA, высоким быстродействием явно не отличался. Центральный процессор не успевал за период кадровой развертки полностью обновить кадровый буфер, поэтому в некоторых ячейках видеопамяти оставались неправильные, случайные значения. В результате на изображении появлялась характерная рябь, напоминавшая снег. Эти искажения назвали «снегом CGA» *(CGA snow).*

Появление видеоадаптера CGA послужило толчком к созданию первых графических игровых программ для IBM PC. Строго говоря, именно они и стали предками современных мультимедийных игр, окунувших человечество в мир виртуальной реальности. Однако низкое разрешение и неестественная цветовая гамма видеосистемы CGA не только сильно портили общее впечатление от графики, но и негативно влияли на зрение пользователя. Возникла необходимость в кардинальном повышении качества изображения (особенно в графическом режиме), и в 1984 г. фирма IBM выпустила очередной тип видеосистемы.

1.3.3. EGA

Новая видеосистема EGA *(Enhanced Graphics Adapter* — улучшенный графический адаптер) обеспечивала более высокое разрешение по вертикали и большее количество отображаемых цветов и имела более высокое быстродействие. Видеосистема EGA появилась в составе более мощного компьютера — IBM PC AT *(Advanced Technology).* Этот PC был оснащен новым 16-разрядным центральным процессором Intel 80286 с тактовой частотой 8 МГц и 16-разрядной шиной ISA. В результате быстродействие IBM PC AT стало значительно выше, чем IBM PC XT, соответственно увеличилась и скорость работы видеосистемы.

Максимальное разрешение, обеспечиваемое видеосистемой EGA, составило 640×350, что позволило повысить качество изображения в текстовом и графическом режимах работы.

В текстовом режиме размер знакоместа был увеличен до 8×14 пикселов (размер символа составил 7×9). Благодаря увеличению расстояния между символами и их более детальной, по сравнению с CGA, прорисовке, была значительно улучшена различимость текста.

Для улучшения работы в графическом режиме были усовершенствованы видеоадаптер и монитор.

Во-первых, был значительно увеличен объем видеопамяти. Размер видеопамяти видеоадаптеров EGA первых моделей составлял 64 Кбайт, в дальнейшем он был увеличен до 128 Кбайт. Это позволило обеспечить 16-цветное изображение при максимальном разрешении 640×350 (для сравнения напомним, что видеоадаптер CGA при максимальном разрешении 640×200 мог обеспечить только монохромное изображение).

Во-вторых, существенной доработке подвергся монитор. Он остался цифровым, но каждая электронная пушка ЭЛТ, помимо основного, была оборудована дополнительным модулятором. Поэтому вместо четырех сигналов (как в мониторе CGA) на ЭЛТ монитора EGA подавалось шесть — по два на каждую электронную пушку. За счет этого каждый из трех основных цветов имел четыре градации яркости, а общая палитра монитора составила 64 оттенка цвета. Была увеличена также частота строчной и кадровой синхронизации, благодаря чему мерцание экрана стало менее заметным.

Для обеспечения новых режимов работы видеосистемы были расширены функции BIOS. ПЗУ с реализацией этих расширенных функций помещалась на плате видеоадаптера EGA.

Видеоадаптер EGA (рис. 1.15) имел еще одну важную особенность, существенную для неанглийских пользователей PC: наряду с аппаратным знакогенератором, он позволял использовать и программный, т. е. загружаемые шрифты. Это значительно облегчило поддержку национальных языков (в частности,

русификацию).

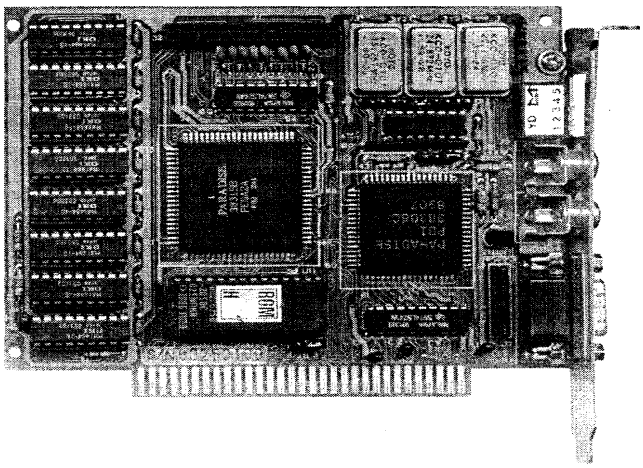


Рис. 1.15. Видеоадаптер EGA

Видеоадаптер EGA обеспечивал хорошую совместимость с программным обеспечением, так как поддерживал видеорежимы всех предыдущих адаптеров.

Хотя видеосистема EGA была заметно лучше, чем CGA, качество формируемого ею графического изображения все равно нельзя было считать удовлетворительным. Причина заключалась не столько в низком разрешении, сколько в ограниченном количестве одновременно отображаемых цветов (16), совершенно не сопоставимом с той палитрой, которую способен воспринимать человек, — около 240 000 оттенков цветов. Поэтому судьба видеосистемы EGA была предрешена. Ее не спасло даже появление расширения стандарта EGA — видеосистемы EGA-плюс, которая обеспечивала более высокое разрешение — 800×600 и 16 цветов. Эта новинка просто не успела получить широкого распространения, так как вскоре появилось принципиально новое техническое решение, позволившее радикально улучшить качество изображения на экране монитора PC.

Рассмотрим структурную схему, принципы работы и регистровую архитектуру адаптера EGA.

1.3.3.1. Структурная схема адаптера EGA

Структурная схема адаптера EGA приведена на рис. 1.16.

Основными функциональными узлами адаптера являются *контроллер CRT, синхронизатор, графический контроллер, контроллер атрибутов,* а также *буферная память, состоящая из 4 плоскостей*.

Контроллер CRT формирует сигналы кадровой и строчной разверток для управления дисплеем и сигнал бланкирования видеосигнала по обратному ходу, формирует адрес для доступа к буферной памяти и адреса строк символов в алфавитно-цифровом режиме, а также синхросигналы курсора и подчеркивания.

При адресации буфера в режиме графики контроллер CRT выполняет преобразование адресов для обеспечения совместимости с цветным графическим адаптером, так как в улучшенном адаптере отсутствует смещение в 8 килобайт между четными и нечетными линиями и идет сквозная адресация памяти.

Аппаратура контроллера CRT обеспечивает режим расщепления экрана. Если счетчик линий сканирования достигает запрограммированного значения, адресные цепи буфера сбрасываются в нуль и регенерация буфера экрана начинается с нулевого адреса. Это позволяет отображать на экране несмежные области буфера.

Контроллер CRT обеспечивает также аппаратную поддержку вертикального и горизонтального панорамирования точек.

Синхронизатор формирует синхросигналы памяти и синхросигналы символов для регенерации изображения, а также обеспечивает доступ процессора при записи или считывании информации к буферу во время активного отображения. Регистр маски, входящий в синхронизатор, защищает отдельные плоскости памяти от случайной записи в них процессором. Плоскость знакогенератора в алфавитно-цифровом режиме открывается для записи только на время загрузки знакогенератора.

Графический контроллер обеспечивает работу с буфером в режиме регенерации и при обращении к нему процессора по записи или считыванию. Графический контроллер передает данные из процессора в видеопамять при записи, направляет данные из памяти на шину данных процессора при считывании памяти и в контроллер атрибута при регенерации экрана. В алфавитно-цифровом режиме данные из памяти поступают на контроллер атрибута и на плоскость генератора знаков, минуя графический контроллер.

Графический контроллер обеспечивает два режима считывания данных процессором и три режима записи в буферную память, причем при записи могут выполняться логические операции И, ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ над исходной информацией и информацией, считанной из адресуемой ячейки памяти.

Контроллер атрибутов предназначен для формирования сигналов цветности в соответствии с

разрядами атрибута или кодом растровой точки в режиме графики (четыре одноименных разряда в четырех плоскостях памяти). Разряды атрибута посылаются в массив из 16 6-разрядных регистров палитры как индекс. Перепрограммированием этих регистров можно менять цветовую палитру. Содержимое выбранного регистра палитры подается на контакты дисплея. Цвет, который будет виден на экране дисплея, зависит от содержимого регистра палитры, а не от значения разрядов атрибута. Если все регистры палитры установлены в нуль, экран будет черным независимо от содержимого разрядов регистра атрибутов.

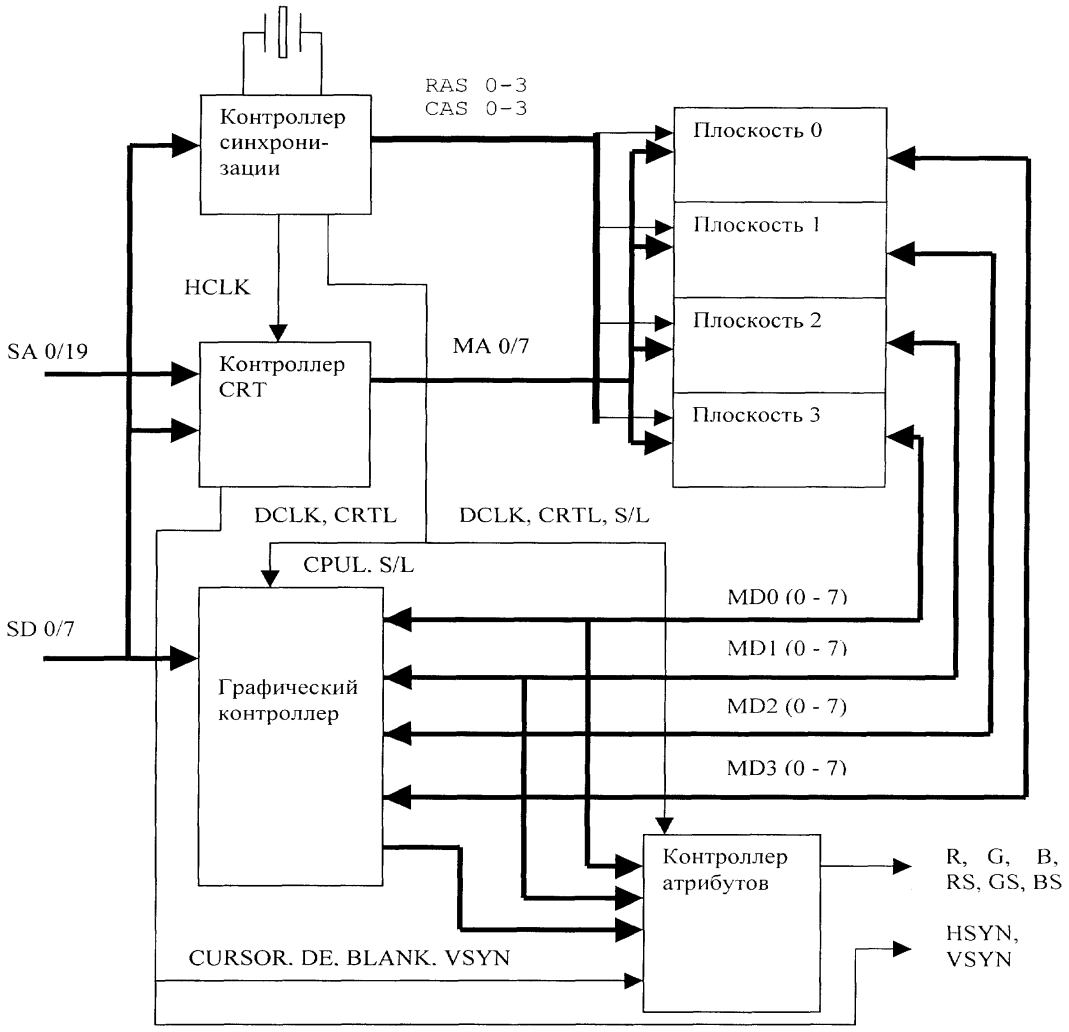


Рис. 1.16. Структурная схема адаптера EGA

Организация буферной памяти дисплея принципиально отличается от организации памяти в адаптерах монохромного и цветного дисплеев. Память буфера разбивается логически на 4 параллельные плоскости, которые занимают в адресном пространстве процессора одни и те же адреса, а необходимый объем памяти определяется исходя из максимального количества цветов (16) в графическом режиме высокого разрешения 640×350). В этом случае для отображения 224 000 (640×350) точек каждая плоскость должна содержать 28 000 байт (224 000 : 8), а четыре плоскости - 128 К байт, что обеспечивает хранение только одной дисплейной страницы в режиме графики высокого разрешения.

Для работы со страницами в графическом режиме необходимо иметь емкость буфера не менее 256 Кбайт. При наличии памяти 128 Кбайт отсутствует возможность работы со страницами в графике высокого разрешения, а при минимально необходимом объеме 64 Кбайт цветовая палитра в графике высокого разрешения уменьшается до 4 цветов.

Страничная организация буферной памяти означает, что для получения 4 плоскостей, т.е. информационного потока в 32 разряда, необходима соответствующая организация элементов памяти (видеопамять). Оптимальными по потребляемой мощности и по занимаемой площади для установки микросхем являются элементы с организацией динамической памяти 64 Кбит × 4.

В зависимости от режима работы плоскости памяти адресуются по-разному и выполняют различные функции. В режиме графики высокого разрешения при достаточном объеме памяти все 4 плоскости занимают одно и то же адресное пространство процессорной памяти и могут адресоваться одновременно. В этом же режиме при объеме памяти 64 Кбайт устанавливается режим цепочки. Плоскость 1 помещается за плоскостью 0, плоскость 3 за плоскостью 2, что удваивает область адресов доступных процессору.

В режиме графики 320×200 в регистре режима памяти синхронизатора устанавливается четно-нечетный выбор памяти. Четные адреса поступают на плоскости 0 и 2, нечетные - на плоскости 1 и 3, а при регенерации экрана четные адреса поступают с четных карт памяти, нечетные - с нечетных карт. Таким образом эмулируется режим графики с двумя смежными разрядами в байте на точку.

В алфавитно-цифровом режиме в плоскость 0 записывается байт символа, в плоскость 1 - байт атрибута, а плоскость 2 используется в качестве знакогенератора. При размере плоскости 64 Кбайт можно загрузить до 4 различных наборов знаков, а их динамическое переключение предоставляет большие возможности для работы в алфавитно-цифровом режиме с использованием псевдографики.

1.3.3.2. Формирование изображения

Структура формирования видеосигнала приведена на рис. 1.17.

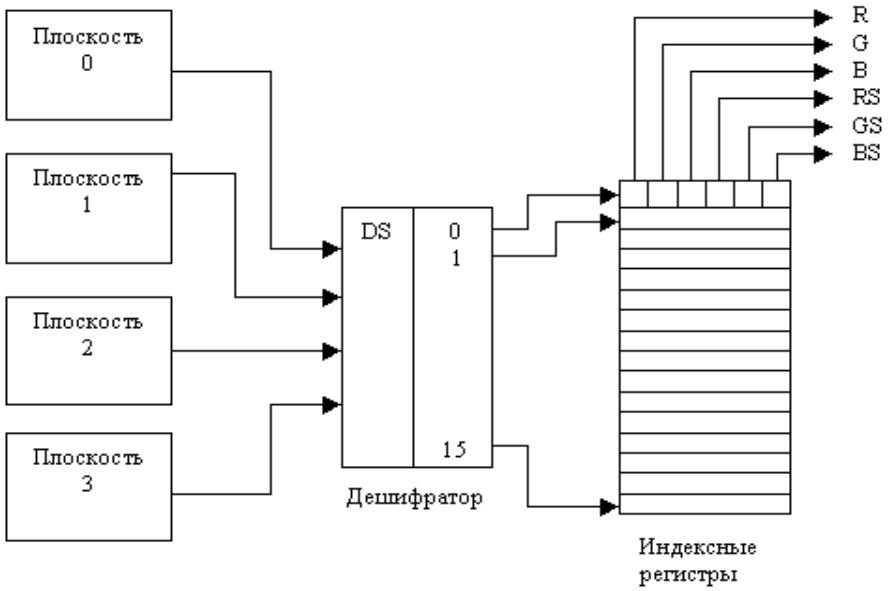


Рис. 1.17. Структура формирования видеосигнала

В графическом режиме каждый пиксел экрана кодируется одним разрядом в каждой плоскости видеопамяти. Таким образом, цвет каждой точки кодируется в памяти четырьмя разрядами и может принимать значение кода одного из шестнадцати цветов. Но получаемое сочетание еще не является идентификатором цвета на экране. Цвет точки на экране получается путем индексации четырехразрядного кода цвета точки через шестиразрядные регистры палитры контроллера атрибутов. Четырехразрядный код цвета выбирает один из шестнадцати шестиразрядных регистров палитры, что обеспечивает одновременное отображение 16 цветов из палитры 64 цветов. Для выбора других 16 цветов необходимо перепрограммировать регистры палитры.

В алфавитно-цифровом режиме в нулевой плоскости хранится код символа, в первой - атрибут знака и фона, во второй - четыре знакогенератора, из которых два могут быть выбраны одновременно, что позволяет расширить набор одновременно используемых символов с 256 до 512. Третья плоскость в этом режиме не используется.

Карта распределения знакогенераторов второй плоскости видеопамяти приведена рис. 18.

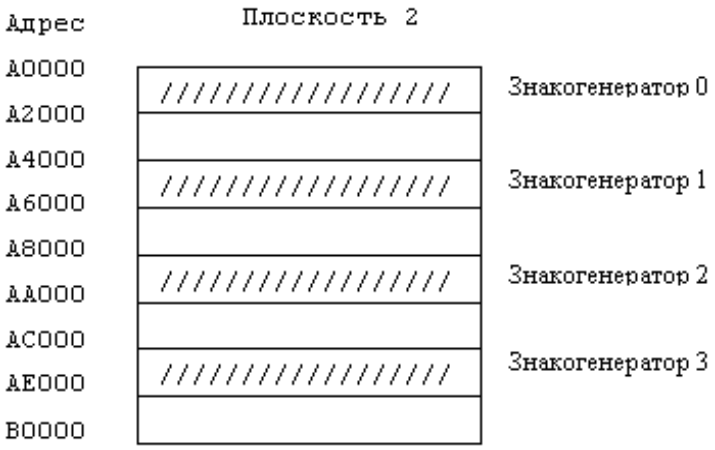


Рис. 18. Карта распределения знакогенераторов второй плоскости видеопамяти.

1.3.3.3. Регистры видеоадаптера EGA.

1.3.3.3.1. Общие регистры.

Адаптер EGA использует для управления режимами работы и контроля его текущего состояния один внешний управляющий регистр "Регистр смешанного вывода (Miscellaneous Output)" и два регистра состояния: "Нулевой регистр входного состояния (Input Status 0)" и " Первый регистр входного состояния (Input Status 1)".

Обращение к регистрам осуществляется через указанные при описании порты.

Общие регистры			
Адрес ввода-вывода	Чтение/ запись	Индекс	Назначение
3C2h По данным [3]	Запись	-	Регистр смешанного вывода (Miscellaneous Output) [0] - не используется; [1] - разрешение обращения процессора к буферу дисплея: 0 - буфер заблокирован для записи (считывания) процессором; 1 - обращение к буферу разрешено. [3, 2] - имеют двойную функцию: выбор частоты задающего генератора: 00- выбирается частота 14 МГц; 01- выбирается частота 16 МГц / выбор считываемого разряда переключателя режимов: 00 - 1-й разряд переключателя; 01 - 2-й разряд переключателя; 10 - 3-й разряд переключателя; 11 - 4-й разряд переключателя. [4, 5] - не используются; [6] - полярность строчного синхроимпульса: 0 - полярность положительная; 1 - полярность отрицательная. [7] - полярность кадрового синхроимпульса: 0 - полярность положительная; 1 - полярность отрицательная
3C2h По данным [2].	Запись	-	Регистр смешанного вывода (Miscellaneous Output). [0] - бит выбора адресного пространства для обращения к регистрам адаптера: 0 - адресное пространство для монохромного режима адаптера (3BXh); 1 - адресное пространство для цветного режима адаптера (3DXh); [1] - бит разрешения доступа *к* ОЗУ адаптера: 0 - запрет; 1 - разрешение [3, 2] - выбор H-такта адаптера: 00 - 640 точек (320 в режиме повторного сканировании) на линию; 01 - 720 точек на линию; 10 - внешний такт; [4] - бит отключения видеосигнала: 0 - управление от EGA; 1 - управление от соединившегося. [5] - бит выбора режима адресации: 0 - выбор младшей страницы, плоскость 0; 1 - выбор старшей страницы, плоскость 1. [7, 6] - выбор V-такта адаптера: 00 - 200 линий; 01 - 350 линий; 10 - ошибка; 11 - ошибка.

Продолжение

3C2h По данным [3].	Чтение	-	Нулевой регистр входного состояния (Input Status 0) [0-3] - не используются; [4] - состояние четырехразрядного переключателя режимов (номер считываемого разряда переключателя задан разрядами 2, 3 порта вывода 3C2h); [5, 6] - не используются; [7] - прерывание по обратному ходу луча по кадру: 0 - прерывание сброшено; 1 - запрос на прерывание
3C2h По данным [2].	Чтение	-	Нулевой регистр входного состояния (Input Status 0). [0-6] - зарезервированы. [7] - прерывание по обратному ходу луча по кадру: 0 - прерывание сброшено; 1 - запрос на прерывание
3BAh - моно-хромный; 3DAh - цвет-ной	Чтение	-	Первый регистр входного состояния (Input Status 1) [0] - отражает текущее состояние видеосигнала: 1 - гашение сигнала; 0 - сигнал активен. [1] - бит включения светового пера. [2] - текущее состояние кнопки светового пера: 0 - кнопка нажата; 1 - кнопка отпущена. [3] - идентификация вертикального обратного хода луча. [5, 4] - биты диагностики. [7, 6] - не используются.

1.3.3.3.2. Регистры контроллера синхронизации (генератора последовательности)

Контроллер синхронизации EGA имеет пять внутренних 8-битовых регистров данных и один адресный (индексный) регистр. Обращение к адресному регистру осуществляется через порт 3C4h, к регистрам данных - через порт 3C5h.

Для обращения к регистру данных в регистр адреса следует записать номер требуемого регистра.

Регистры контроллера синхронизации			
Адрес ввода-вывода	Чтение/ запись	Индекс	Назначение
3C4h	Чтение/ запись	-	Индексный регистр, содержимое которого определяет адрес регистра данных.
3C5h	Запись	00h	Регистр сброса (Reset): [0] - асинхронный сброс (если равен 0); [1] - синхронный сброс (если равен 0)
3C5h По данным [3].	Запись	0lh	Регистр режима синхронизации (Clock Mode): [1] - период синхросигнала для CRT: 0 - равен 8 периодам задающего генератора; 1 - равен 16 периодам задающего генератора; [3] - частота синхросигналов для регистров сдвига ГК и КА, 0 - равна частоте задающего генератора; 1 - равна частоте задающего генератора, деленной на 2
3C5h По данным [2].	Запись	0lh	Регистр режима синхронизации (Clock Mode): [0] - определение длительности символьной единицы: 1 - 8 битов на символ; 0 - 9 битов на символ. [1] - определяет количество циклов с разрешонным доступом процессора к видеопамяти: 1 - 3 цикла из 5 (используется в режиме низкого разрешения); 0 - 1 цикл из 5 (используется в режиме высокого разрешения).

Продолжение

3C5h	Запись	02h	Регистр маски плоскости при записи ОЗУ (Color Plane Write Enable). Нулевое значение разрядов запрещает запись в соответствующую плоскость: [0] - маска плоскости 0; [1] - маска плоскости 1; [2] - маска плоскости 2; [3] - маска плоскости 3
3C5h	Запись	03h	Регистр выбора номера знакогенератора в алфавитно-цифровом режиме (Character Generator Select): [0, 1] - карта B (выбирается один из четырех знакогенераторов при разряде 3 байта атрибута, равном нулю); [2, 3] - карта A (выбирается один из четырех знакогенераторов при разряде 3 байта атрибута, равном единице).
3C5h	Запись	04h	Режим работы памяти: [0] - 1 - алфавитно-цифровой, 0 - графический; [2] - 1 - последовательная адресация, 0 - режим чет (нечет) - четными адресами обращаются к плоскости 0, нечетными - к плоскости 1.

1.3.3.3.3. Регистры видеоконтроллера (контроллера электронно-лучевой трубки).

Видеоконтроллер адаптера EGA отличается от рассмотренного ранее видеоконтроллера Motorola 6845. Его программный интерфейс дозволяет эмулировать видеоконтроллер адаптеров MDA, CGA и HGC, но назначения и форматы соответствующих регистров отличаются, поэтому использование программы, оперирующей непосредственно регистрами видеоконтроллера 6845 (не через прерывания), может привести к ошибке.

Видеоконтроллер EGA имеет 25 внутренних 8-битовых регистра данных и один адресный (индексный) регистр. Обращение к адресному регистру для монохромного режима осуществляется через порт 3B4h, для цветного - через порт 3D4h. К регистрам данных - через порты 3B5h и 3D5h соответственно.

Для обращения к регистру данных необходимо в регистр адреса записать номер требуемого регистра.

Регистры контроллера CRT			
Адрес ввода-вывода	Чтение/ запись	Индекс	Назначение
* 3?4h/3?4h	Чтение/ запись	-	Индексный регистр. Служит для адресации регистров данных
* 3?5h/3?5h	Запись	00h	Регистр общего количества символов в строке. (Horizontal Total) Определяет период строчного синхроимпульса. Содержит общее количество символов в строке сканирования, включая обратный ход по строке
* 3?5h/3?5h	Запись	0lh	Регистр конца отображения по строке. (Horizontal Displayed Enable End). Определяет число символов, отображаемых в строке
* 3?5h/3?5h	Запись	02h	Регистр начала гашения по строке (Start Horizontal Blanking). Определяет момент установки сигнала гашения по строке
*3?5h/3?5h	Запись	03h	Регистр конца гашения по строке (End Horizontal Blanking). [0-4] - определяют момент сброса сигнала гашения по строке; [5, 6] - управляют смещением сигнала разрешения отображения вправо: [6] [5] Смещение в символах 0 0 0 0 1 1 1 0 2 1 1 3

Продолжение

* 3?5h/3?5h	Запись	04h	Регистр начала строчного синхроимпульса (Start Horizontal Retrace). Определяет момент установки синхросигнала строчной развертки
* 3?5h/3?5h	Запись	05h	Регистр конца строчного синхроимпульса (End Horizontal Retrace). [0-4] - определяют момент сброса строчного синхроимпульса; [5, 6] - управляют задержкой строчного синхроимпульса: [6] [5] Задержка в символах 0 0 0 0 1 1 1 0 2 1 1 3 [7] - должен быть нулевым во всех режимах
* 3?5h/3?5h	Запись	06h	Регистр общего количества линий сканирования в кадре (Horizontal Total). Этот девятиразрядный регистр (см. разряд 0 регистра 3D5.07) определяет период кадрового синхроимпульса. Содержит общее количество линий сканирования в кадре, включая обратный ход по кадру
* 3?5h/3?5h	Запись	07h	Регистр дополнения (Overflow). Содержит девятые разряды некоторых других управляющих регистров: [0] - разряд 8 регистра 3D5.06, [1] - разряд 8 регистра 3D5.12, [2] - разряд 8 регистра 3D5.10, [3] - разряд 8 регистра 3D5.15, [4] - разряд 8 регистра 3D5.18.
*3?5h/3?5h	Запись	08h	Регистр предварительной установки линии сканирования в строке (Preset Row Scan). [0-4] - в текстовом режиме определяет номер линии в строке, с которой начинается отображение строки (используется для обеспечения плавной прокрутки текста; [5-7] - зарезервированы.
* 3?5h/3?5h	Запись	09h	Регистр максимальной линии сканирования в строке (Maximum Scan Line Address). [0-4] - число линий в символьной строке минус 1; [5-7] - зарезервированы.
* 3?5h/3?5h	Запись	0Ah	Регистр начала курсора (Cursor Start). [0-4] - определяет линию сканирования в строке, где начинается отображение курсора; [5-7] - зарезервированы.
* 3?5h/3?5h	Запись	0Bh	Регистр конца курсора (Cursor End). [0-4] - определяют линию сканирования в строке, где кончается отображение курсора; [5, 6] - управляют сдвигом курсора вправо: [6] [5] Сдвиг курсора в символах 0 0 0 0 1 1 1 0 2 1 1 3
* 3?5h/3?5h	Чтение/ запись	0Ch	Регистр начального адреса (старший байт) (Start Address Height). Вместе с содержимым регистра 3D5.0D определяет начальный адрес регенерации изображения относительно начала видеобуфера.

Продолжение

* 3?5h/3?5h	Чтение/ запись	0Dh	Регистр начального адреса (младший байт) (Start Address Low). Вместе с содержимым регистра 3D5.0C определяет начальный адрес регенерации изображения относительно начала видеобуфера.
* 3?5h/3?5h	Чтение/ запись	0Eh	Регистр адреса курсора (старший байт) (Cursor Location Height). Вместе с содержимым регистра 3D5.0F определяет позицию курсора относительно начала видеобуфера.
*3?5h/3?5h	Чтение/ запись	0Fh	Регистр адреса курсора (младший байт) (Cursor Location Low). Вместе с содержимым регистра 3D5.0E определяет позицию курсора.
* 3?5h/3?5h	Запись	10h	Регистр начала синхроимпульса кадровой развертки (Vertical Retrace Start). Этот девятиразрядный регистр (см. разряд 2 регистра 3D5.07) определяет момент установки синхроимпульса кадровой развертки и обратного хода луча по кадру.
* 3?5h/3?5h	Чтение	10h	Регистр светового пера (старший байт) (Light Pen Height). В регистре содержится старший байт смещения позиции светового пера относительно начала видеобуфера.
* 3?5h/3?5h	Запись	11h	Регистр конца синхроимпульса кадровой развертки (Vertical Retrace End): [0-3] - определяют момент сброса кадрового синхроимпульса. В регистре программируется величина, равная значению, содержащемуся в регистре 3D5.10, плюс длительность кадрового синхроимпульса (в линиях сканирования); [4] - сброс прерывания по обратному ходу кадра. Единица сбрасывает прерывание; [5] - разрешение прерывания по обратному ходу кадра. Единица разрешает выдачу прерывания.
* 3?5h/3?5h	Чтение	11h	Регистр светового пера (младший байт) (Light Pen Low). Вместе с регистром 3D5.10 светового пера содержит 16-разрядный адрес позиции светового пера.
* 3?5h/3?5h	Запись	12h	Регистр конца отображения по кадру (Vertical Display Enable End). Девятиразрядный регистр (см. разряд 1 регистра 3D5.07) определяет количество отображаемых линий сканирования в кадре.
* 3?5h/3?5h	Запись	13h	Регистр смещения (Offset (Logical Line Width)). Задает логическую ширину экрана в словах. С помощью этого регистра определяется начальный адрес каждой следующей строки экрана во время отображения (прямого хода по кадру)
*3?5h/3?5h	Запись	14h	Регистр позиции подчеркивания (Underline Location). [0-4] - определяет линию сканирования в строке, где возможно подчеркивание символа.
* 3?5h/3?5h	Запись	15h	Регистр начала гашения по кадру (Start Vertical Blanking). Этот девятиразрядный регистр (см. разряд 3 регистра 3D5.07) определяет момент установки сигнала гашения по кадру в линейных единицах.
* 3?5h/3?5h	Запись	16h	Регистр конца гашения по кадру (End Vertical Blanking). [0-4] - определяет момент сброса сигнала гашения по кадру в линейных единицах.

Продолжение

* 3?5h/3?5h	Запись	17h	Регистр управления режимом (Mode Control): [0] - режим совместимости с CGA. При наличии логического нуля в этом разряде значение разряда 0 счетчика линий сканирования в строке заменяет значение разряда 13 адреса регенерации , что обеспечивает адресацию нечетных линий сканирования через 8 Кбайт в области буфера дисплея. При логической единице такая замена не осуществляется; [1] - режим совместимости с платой графики "Геркулес" (HGC). При наличии логического нуля в этом разряде значение разряда 1 счетчика линий сканирования в строке заменяет значение разряда 14 адреса регенерации , что обеспечивает адресацию каждой второй и третьей линий сканирования через 16 Кбайт в области буфера дисплея. При логической единице такая замена не осуществляется; [2] - удвоение разрешающей способности по кадру. Девятиразрядный регистр 3D5.06 обеспечивает максимальное разрешение, равное 512 линиям сканирования в кадре. Если счетчик линий в кадре синхронизируется с частотой, меньшей вдвое, то максимальное число линий сканирования в кадре удваивается (1024). [2] = 0 - синхронизация счетчика линий в кадре осуществляется по каждой линии сканирования, [2] = 1 - синхронизация счетчика линий в кадре осуществляется по каждой второй линии сканирования. Системный сброс устанавливает данный разряд в нуль; [3]-синхронизация счетчика адреса регенерации. [3] = 0 - счетчик синхронизируется от CLK, [3] = 1 - счетчик синхронизируется от CLK, деленного на 2; [4] - управление выходами: [4] = 1 - на выходах синхронизации формируется состояние высокого импеданса, [4] = 0 - формирование нормальных выходов. Системный сброс устанавливает данный разряд в нуль; [5] - преобразование адреса регенерации изображения . В режиме байта (3D5.17 [6] = 1) преобразование адреса не происходит. В режиме слова (3D5.17 [6] = 0) адрес регенерации изображения формируется следующим образом: если [5] = 0, то MA0=MA13; если [5]=1, то MA0=MA15 (см. 3D5.17, разряд 6); [6] - режим слова или режим байта: [6]=0 - режим слова; [6]=1 - режим байта. В режиме слова выходы счетчика адреса регенерации MA сдвигаются вниз на один разряд (MA1 = MA0, MA2 = MA1 и т.д.), а старший значащий разряд счетчика (MA13 или MA15) в зависимости от значения разряда 5 становится младшим значащим выходом адреса регенерации изображения MA0; [7] - аппаратурный сброс. Логический нуль сбрасывает HSYN и VSYN. Логическая единица разрешает формирование HSYN и VSYN. Системный сброс устанавливает данный разряд в нуль
*3?5h/3?5h	Запись	18h	Регистр сравнения линий (Line Compare). Этот девятиразрядный регистр (см. разряд 4 регистра 3D5.07) определяет линию сканирования, где экран разделяется на два окна. Информация, отображаемая в нижнем окне/ всегда начинается с адреса 0000 (начало буфера)

1.3.3.3.4. Регистры контроллера атрибутов

Контроллер атрибутов адаптера EGA имеет 21 внутренний 8 -битовый регистр. Обращение к регистрам осуществляется через порт 3C0H.

Для обращения к конкретному регистру необходимо в порт 3C0h записать индекс (адрес) регистра, следующей операцией записи в порт 3C0h устанавливается состояние самого регистра (flip -flop технология обращения). Используя такой способ обращения, необходимо следить за последовательностью операций с регистрами. Операции записи в порт должны быть парными. Если текущее состояние внутреннего переключателя неизвестно, то можно сбросить его в исходное состояние. Для этого достаточно прочитать регистр Input Status 1 (обращение через порт 3BAh/3DAh).

Регистры контроллера атрибутов			
Адрес ввода-вывода	Чтение/ запись	Индекс	Назначение
3DAh	Чтение		Регистр состояния (Input Status 1): [0] - разрешение отображения информации; [3] - обратный ход по кадру; [4, 5] - диагностирование видеовыходов (в зависимости от значения разрядов 4, 5 регистра 3C0.13. Чтение регистра состояния сбрасывает внутренний переключатель в исходное состояние, соответствующее вводу по адресу 3C0h адреса регистра
3C0h	Запись		Индексный регистр адреса, указывает регистр данных, в который записывается информация. По этому же адресу записывается и информация для выбранного регистра. После записи индекса внутренний переключатель настраивает порт на прием данных для выбранного регистра. [0-4] - номер регистра; [5] - бит запрета модификации палитры: 0 - палитра может быть модифицирована, экран погашен; 1 - палитра не может быть модифицирована, экран в режиме отображения.
3C0h	Запись	00h-0Fh	Регистры палитры с 0 по F, определяют цвет отображаемой точки: [0] - голубой (B) ; [1] - зеленый (G) ; [2] - красный (R) ; [3] - голубой интенсивный (BS); [4] - зеленый интенсивный (GS); [5] - красный интенсивный (RS); [6, 7] - зарезервированы. В режиме совместимости с CGA: [0] - голубой (B) ; [1] - зеленый (G) ; [2] - красный (R) ; [3, 4] - интенсивность; [5-7] - зарезервированы. В монохромном режиме: [3] - видеосигнал; [4] - интенсивность; [0-2] и [5-7] зарезервированы.
3C0h	Запись	10h	Управление режимом (Mode Control): [0] - определяет способ декодирования атрибутов: 1 - как четыре бита графики, 0 - как байт для текстового режима. [1] - определяет вид интерпретации атрибутов: 0 - как атрибут цвета; 1 - как атрибут для монохромного монитора. [2] - в текстовом режиме для монохромного монитора при ширине символьной матрицы 9 бит: 0 - 9-я строка символьной матрицы имеет цвет фона; 1 - 9-я строка символьной матрицы дублирует 8-ю строку, что позволяет использовать непрерывные горизонтальные линии псевдографики; [3] - в текстовом режиме определяет интерпретацию7-го бита атрибута: 0 - бит расширения цвета фона; 1 - бит включения режима мерцания, в графическом режиме включает режим мерцания; [4-7] - зарезервированы.

Продолжение

<table>
<tr><td>3C0h</td><td>Запись</td><td>11h</td><td>Цвет границы экрана (Screen Border Color):
[0-3] - значение цвета рамки, [4-7] - зарезервированы.</td></tr>
<tr><td>3C0h</td><td>Запись</td><td>12h</td><td>Разрешение плоскости цвета (Color Plane Enable):
[0-3]: 1 - разрешение соответствующих плоскостей памяти для отображения, 0 - запрет отображения;
[4, 5] - мультиплексирование состояния видеовыходов КА.
Состояние разрядов порта 3DA в зависимости от состояния разрядов 4, 5 регистра 3C0.12:
<table>
<tr><td colspan="2">Разряды регистра 3C0.12</td><td colspan="2">Разряды порта состояния (3DA)</td></tr>
<tr><td>[5]</td><td>[4]</td><td>[5]</td><td>[4]</td></tr>
<tr><td>0</td><td>0</td><td>R</td><td>B</td></tr>
<tr><td>0</td><td>1</td><td>BS</td><td>G</td></tr>
<tr><td>1</td><td>0</td><td>RS</td><td>GS</td></tr>
<tr><td>1</td><td>1</td><td>-</td><td>-</td></tr>
</table></td></tr>
<tr><td>3C0h</td><td>Запись</td><td>13h</td><td>Панорамирование изображения по горизонтали (Horizontal Panning): [0-3] - число элементов для сдвига видеоданных влево. При константе панорамирования больше 7 сдвиг видеоданных влево не происходит.</td></tr>
<tr><td>3C0h</td><td>Запись</td><td>14h</td><td>Выбор цвета по умолчанию (Color Select)</td></tr>
</table>

1.3.3.3.5. Регистры графического контроллера.

Графический контроллер EGA имеет девять внутренних 8-битовых регистров данных и один адресный (индексный) регистр. Обращение к адресному регистру осуществляется через порт 3CEh, к регистрам данных - через порт 3CFh.

Для обращения к регистру данных в регистр адреса следует записать номер требуемого регистра.

<table>
<tr><td colspan="4">Регистры графического контроллера</td></tr>
<tr><td>Адрес ввода-вывода</td><td>Чтение/ запись</td><td>Индекс</td><td>Назначение</td></tr>
<tr><td>3CEh</td><td>Запись</td><td>-</td><td>Индексный регистр. Используется для выбора внутренних регистров контроллера</td></tr>
<tr><td>3CFh</td><td>Запись</td><td>00h</td><td>Установка (сброс) (Set/Reset). Разряды 0-3 регистра записываются соответственно в плоскости 0-3, если выбран режим записи 0 и разрешен режим установки (сброса) для соответствующей плоскости регистром разрешения установки (сброса) (см. 3CF.01)</td></tr>
<tr><td>3CFh</td><td>Запись</td><td>01h</td><td>Разрешение установки (сброса) (Set/Reset Annabel). Этот регистр работает совместно с 3CF.00. Единичное состояние разрядов 0-3 разрешает действие регистра установки (сброса) на соответствующую плоскость. При нулевом значении любого разряда 3CF.01 соответствующая плоскость загружается данными с шины данных процессора в режиме записи 0</td></tr>
<tr><td>3CFh</td><td>Запись</td><td>02h</td><td>Сравнение цвета (Color Compare). Если регистр режима установлен в режим считывания 1, данные, считанные из плоскостей памяти 0-3, сравниваются с разрядами 0-3 регистра сравнения цвета. Результатом сравнения является логическая единица в соответствующем разряде шины данных процессора, например:
<table>
<tr><td>[0]</td><td>1</td><td>1 1 1 1 1 1 1 1</td><td>Плоскость 0</td></tr>
<tr><td>[1]</td><td>1</td><td>0 0 0 0 0 0 0 1</td><td>Плоскость 1</td></tr>
<tr><td>[2]</td><td>0</td><td>1 1 1 1 1 1 1 0</td><td>Плоскость 2</td></tr>
<tr><td>[3]</td><td>0</td><td>0 0 0 0 0 0 0 0</td><td>Плоскость 3</td></tr>
<tr><td></td><td></td><td>0 0 0 0 0 0 0 1</td><td>Результат</td></tr>
</table></td></tr>
</table>

Продолжение

3CFh	Запись	03h	Циклический сдвиг данных и выбор функции (Data Rotate /Function Select): [0-2] - счетчик сдвига влево данных процессора. Используется в режиме записи 0; [3, 4] - выбор функции для режимов записи 0, 2. Разрешает логические операции между данными, считанными из буфера в контроллер (ГК), и данными, записываемыми в буфер: 00 - данные не изменяются; 01 - ИЛИ; 10 - И; 11 - ИСКЛЮЧАЮЩЕЕ ИЛИ
3CFh	Запись	04h	Выбор плоскости считывания (Read Plane Select). [0-2] - закодированный номер плоскости, считываемой в процессор. Используется только в режиме считывания 0
3CFh	Запись	05h	Регистр режима (Mode). [0, 1] режимы записи: 00 - режим записи 0, каждая из четырех плоскостей загружаются данными из процессора, сдвинутыми согласно счетчику в 3CF.03, если для этих плоскостей нет разрешения установки (сброса). В случае разрешения установки (сброса) плоскости загружаются данными из регистра 3CF.00; 10 - режим записи 1, каждая из четырех плоскостей загружаются данными, считанными в предыдущем цикле; 01 - режим записи 2, плоскости 0-3 заполняются значениями шины данных процессора D0-D3 соответственно; 11 - не используется; [2] - условие проверки. Единичное состояние разряда переводит в третье состояние разряды регистра 3CF.06; [3] режимы считывания: 1- режим считывания 1, процессор считывает результат логического сравнения между данными 4 плоскостей памяти и содержимым регистра сравнения цвета; 0- режим считывания 0, процессор считывает данные из плоскости памяти. Плоскость памяти выбирает регистр 3CF.04; [4] режимы адресации: 1 - режим чет (нечет), 0 - режим последовательной адресации; [5]: 1 -режим CGA, 0 -режим EGA
3CFh	Запись	06h	Смешанный регистр (Miscellaneous): [0]: 1 - режим графики, 0 - алфавитно-цифровой режим; [1]: 1- цепочка: четыре плоскости памяти по 16 К каждая делится на две плоскости по 32 К и нечетная плоскость идет после четной, за нулевой -первая, и за второй - третья, 0 - нормальный режим; [2, 3] - адресация буфера в адресном пространстве процессора: 00 - A0000-BFFFF для 128 Кбайт, 10 - A0000-AFFFF для 64 Кбайт (графика с высоким разрешением), 01 - не используется, 11 - B8000-BFFFF для эмуляции режима 640×200
3CFh	Запись	07h	Регистр блокировки сравнения цвета (Color Don't Care). Работает только в режиме считывания 1. Если разряды 0-3 равны нулю, результатом сравнения в этом случае будут логические единицы во всех разрядах шины данных процессора независимо от считываемой информации, т.е. цвет не обслуживается

Продолжение

3CFh	Запись	08h	Регистр маски разрядов 0-7 (Mask). Любой разряд в этом регистре, установленный в нуль, запрещает запись соответствующего разряда в каждой из четырех плоскостей памяти. Записываемые в память данные, в этом случае будут соответствовать данным, считанным в предыдущем цикле и записанным в контроллере. Любой разряд регистра, установленный в единицу, разрешает различные действия в соответствующем разряде в каждой из четырех плоскостей памяти. Маска разряда подается одновременно на все четыре плоскости

1.3.4. VGA

Почему при худшей, чем у монитора PC, разрешающей способности телевизионный приемник позволяет создать весьма реалистичную картину? Потому, что в телевизоре на кинескоп подается аналоговый видеосигнал, в результате чего количество отображаемых цветовых оттенков практически неограниченно, а все рассмотренные видеосистемы (CGA, EGA) оснащены цифровыми мониторами, имеющими крайне скудную палитру (см. разд. 1.1).

Учет данного обстоятельства привел к тому, что в 1987 г. фирма IBM начала выпуск новой видеосистемы с аналоговым монитором, видеоадаптер которой формировал, соответственно, не цифровой, а аналоговый RGB-сигнал. Эта видеосистема получила название VGA и впервые была реализована в компьютере IBM PS/2 *(Personal System).* Видеосистема VGA включала в себя видеоадаптер VGA, установленный на материнской плате этого компьютера.

Существует несколько вариантов расшифровки аббревиатуры VGA. Первоначально она обозначала название сверхбольшой интегральной схемы, в которой были реализованы основные узлы видеоадаптера: *Video Gate Array*— вентильная матрица для формирования видеосигнала. Аналогичным образом именовалась упрощенная версия видеоадаптера VGA — видеоадаптер MCGA *(Memory Controller Gate Array* — контроллер памяти на основе вентильной матрицы). Однако в дальнейшем нюансы внутреннего устройства этих адаптеров отошли на второй план, а расшифровка аббревиатур стала отражать их функциональные особенности. Была также учтена созвучность аббревиатур с названиями предыдущих видеоадаптеров (CGA, EGA), в которых сочетание GA означало *Graphics Adapter*— графический адаптер. В итоге VGA стали расшифровывать сначала как *Video Graphics Array* — графическая матрица для формирования видеосигнала, а затем как *Video Graphics Adapter*— графический адаптер, формирующий видеосигнал (т. е. аналоговый сигнал). Аналогично название MCGA стали расшифровывать как *Multi-Color Graphics Adapter* — многоцветный видеоадаптер. Очень скоро видеоадаптер VGA стали выпускать в виде самостоятельной платы, устанавливаемой в 16-разрядный слот шины ISA (рис. 1.19).

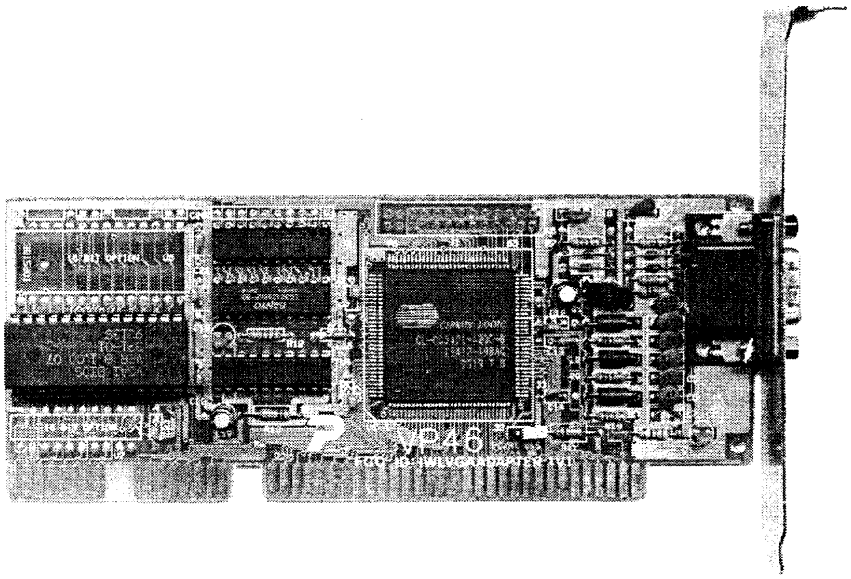


Рис. 1.19. Видеоадаптер VGA

В состав видеоадаптера VGA была включена специальная микросхема — RAMDAC *(Random Access Memory Digital-to-Analog Converter* — цифро-аналоговый преобразователь — ЦАП — данных, хранимых в ОЗУ). RAMDAC представлял собой быстродействующий трехканальный ЦАП с 256 регистрами, образующими его собственное маленькое ОЗУ — RAM (этим и объясняется название данного элемента:

RAMDAC, а не просто DAC). RAMDAC предназначен для преобразования двоичных чисел, содержащихся в ячейках видеопамяти, в три непрерывных RGB-сигнала, уровень которых пропорционален яркости каждого из трех основных цветов. Если для кодирования цвета пиксела использовать *n* двоичных разрядов, то RAMDAC сможет сформировать 2^n различных комбинаций из трех напряжений на выходе видеоадаптера. Каждая из этих комбинаций соответствует определенному оттенку цвета. Увеличивая *n,* можно получить любое количество цветовых оттенков на экране монитора.

Нетрудно догадаться, что основным фактором, ограничивающим максимальное разрешение и количество цветовых оттенков, является объем видеопамяти.

Видеоадаптер VGA имел 256 Кбайт видеопамяти. Это обеспечило поддержку графических режимов 640×480/16 (640×480 при 16 цветах) и 320×200/256. Использование последнего видеорежима впервые позволило получить хоть и немного размытое, но весьма реалистичное по цветовой гамме изображение. Как и прежде, видеоадаптер VGA поддерживал все видеорежимы предыдущих видеоадаптеров.

Максимальный размер палитры определялся разрядностью регистров RAMDAC. В стандартном видеоадаптере VGA каждый из трех каналов RAMDAC являлся шестиразрядным, что обеспечивало палитру из $2^{6\times3}$ = 262144 оттенка цветов. Такая палитра соответствует возможностям органа зрения человека по цветоощущению. Однако из-за ограниченного объема видеопамяти для кодирования цвета каждого пиксела изображения может быть отведено не более 8 бит, поэтому видеоадаптер мог отображать лишь 2^8 = 256 оттенков из указанной палитры.

В видеоадаптере VGA, так же, как и в видеоадаптере EGA, используется косвенное кодирование цвета. RAMDAC видеоадаптера VGA имеет 256 18-разрядных регистров, образующих его собственное ОЗУ. В этих регистрах хранится *текущая палитра,* т. е. 256 комбинаций из 262 144 возможных. Извлекаемая из видеопамяти n-разрядная (n=8) кодовая комбинация задает номер регистра RAMDAC, содержимое которого поступает в ЦАП и определяет в дальнейшем цвет соответствующего пиксела на экране монитора (рис. 1.20).

Регистры RAMDAC, содержащие текущую палитру, имеют свои адреса и доступны для CPU. Это позволяет изменять их содержимое программным путем, т. е. пользователь может оперативно менять текущую палитру.

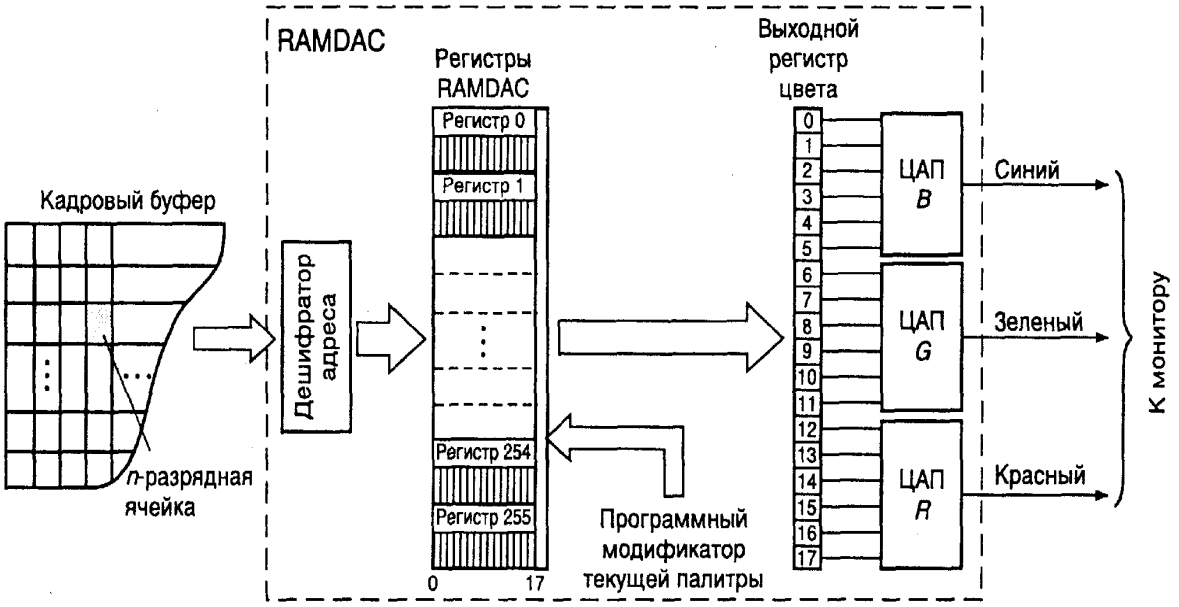


Рис. 1.20. Схема кодирования цвета пиксела в видеоадаптере VGA

Монитор VGA также сильно изменился по сравнению со своими предшественниками. Во-первых, как уже отмечалось, он стал аналоговым. Это позволило существенно упростить его электрическую схему, так как отпала необходимость в использовании сложных схем дешифрации цифрового сигнала. Во-вторых, значительно увеличены полоса пропускания видеотракта (она составила около 25 МГц) и частота строчной развертки (31,5 КГц). Это позволило повысить одновременно и разрешение по вертикали, и частоту кадров. В результате видеосистема VGA смогла обеспечить разрешение 640×480 при частоте кадров 60 Гц (в графическом режиме) и 720×400 при частоте кадров 70 Гц (в текстовом режиме). В последнем случае мерцание экрана практически незаметно, что снижает утомляемость глаз оператора. Использование более высокого разрешения по вертикали позволило увеличить размер знакоместа до 9×16 пикселов. В результате детальность прорисовки символов на экране оказалась выше, чем в видеосистеме MDA, считавшейся до этого эталоном качества изображения в текстовом режиме.

Изменение вида выходного сигнала видеоадаптера привело к необходимости использования другого соединительного кабеля между видеоадаптером и монитором. Новый кабель был оснащен 15-контактным D-образным разъемом и используется до настоящего времени. На первый взгляд, увеличение количества кон-

тактов (кабели для видеосистем EGA и CGA имели 9 контактов) при использовании аналогового видеосигнала выглядит странным, поскольку для его передачи теоретически достаточно четырех проводов: три провода для передачи сигналов R, G и B и один — общий (низкопотенциальный, или нулевой).

Увеличение количества проводов нового кабеля обусловлено следующими причинами.

Во-первых, поскольку видеосигнал, формируемый адаптером VGA, имеет более широкий спектр, чем видеосигналы предыдущих видеоадаптеров, при передаче его по обычному кабелю он испытывает сильное затухание. Для передачи такого сигнала используются два скрученных провода, называемых *витой парой.* В кабеле монитора VGA имеются три витых пары для сигналов R, G и B, т. е. для передачи каждого из этих сигналов используется отдельный нулевой провод, обычно называемый *линией возврата (return).* Такой способ передачи RGB-сигналов позволяет не только избавиться от взаимных помех, обусловленных использованием общего провода, но и устранить затухание высокочастотных составляющих видеосигнала.

Во-вторых, в результате увеличения количества жил кабеля появилось несколько резервных проводов, не задействованных для передачи видеосигнала и синхроимпульсов. Эти провода могут использоваться, например, для обмена служебной информацией между монитором и видеоадаптером. Так, в устаревших моделях PC они использовались для автоматического определения типа монитора, а в современных — также и для реализации специального канала DDC *(Display Data Channel*— канал данных монитора), по которому происходит обмен данными между монитором и видеоадаптером (более подробно об этом см. в [1, 2]).

Увеличение разрешения видеосистемы VGA до 640×480 позволило не только повысить качество изображения, но и значительно упростить процедуру построения на экране различных графических объектов, поскольку пиксел стал квадратным. В отличие от всех предыдущих видеосистем, у монитора VGA отношение разрешения по горизонтали к разрешению по вертикали (640/480 = 4/3) равно отношению ширины экрана к его высоте. Это отношение называется *aspect ratio.* Для большинства устройств отображения (монитор PC, телевизор) параметр aspect ratio имеет оптимальное значение 4/3. Следовательно, количество пикселов на единицу длины по вертикали и горизонтали на экране монитора VGA одинаково.

Квадратный пиксел избавил программистов от необходимости коррекции создаваемых изображений, которую ранее приходилось выполнять на мониторах с прямоугольным пикселом. Например, при программировании окружности на экране монитора видеосистемы EGA будет отображаться эллипс. Напротив, чтобы получить на экране монитора окружность, программировать надо эллипс.

При подключении монохромного дисплея видеоадаптер VGA автоматически определяет его тип и переходит в режим передачи монохромного сигнала с 2^6 = 64 градациями серого цвета (в соответствии с разрядностью одного канала RAMDAC).

Более подробно устройство и особенности работы видеоадаптера VGA будут рассмотрены в гл. 2.

Таким образом, создание видеосистемы VGA явилось качественным скачком в развитии PC вообще и его видеосистемы в частности. Очень скоро эта видеосистема стала промышленным стандартом. Все видеорежимы видеоадаптера VGA, в состав которых входили и режимы его предшественников, получили стандартные номера (табл. 1.4). В состав Video BIOS были включены команды инициализации этих режимов. (Расширение стандартной BIOS - ROM Video BIOS устанавливается на самом видеоадаптере).В частности, для этой цели использовалось программное прерывание INT 10h, при вызове которого в качестве параметра указывался номер видеорежима. Благодаря этому удалось обеспечить хорошую совместимость видеосистем сверху вниз, т. е. программы, написанные для CGA и EGA, корректно работали с адаптером VGA. Это также благоприятно сказалось на быстром распространении стандарта VGA.

Заметим, что в графическом режиме видеоадаптер также может отображать текст с использованием шрифта знакогенератора, встроенного в Video BIOS, однако размеры символьной матрицы при этом будут немного отличаться от размеров матрицы, используемой в текстовом режиме. Поэтому для каждого видеорежима, независимо от его типа (текстовый он или графический), в табл. 1.4 указаны разрешение, количество знакомест и размеры символьной матрицы.

Отметим одну особенность, связанную с нумерацией видеорежимов. Для видеоадаптеров EGA и VGA предусмотрены три пары одинаковых видеорежимов: 00h и 0lh, 02h и 03h, 04h и 05h. Это сделано для обеспечения совместимости этих видеоадаптеров с программами, ориентированными на работу с видеосистемой CGA. Напомним, что в составе этой видеосистемы мог использоваться композитный монитор, для работы с которым видеоадаптер CGA должен был формировать монохромный видеосигнал. Такой сигнал формируется в режимах 00h, 02h и 04h, а в режимах 0lh, 03h и 05h — стандартный сигнал для цветного TTL-монитора. Поскольку стандартные видеоадаптеры EGA и VGA не имеют композитного выхода, для них указанные режимы идентичны.

Эйфория, вызванная широкими возможностями видеоадаптера VGA, длилась недолго. Вскоре появились приложения (например, графические редакторы изображений), для работы которых 256 оттенков цветов при разрешении 320×200 явно недостаточно. Возникла потребность в дальнейшем увеличении разрешения и количества цветов. В первую очередь, это было необходимо для решения задач мультимедиа, нацеленных на создание максимально реалистичного изображения.

Особенности видеосистемы VGA позволили это сделать практически без изменения ее архитектуры. Все усовершенствования носили в основном количественный характер. В частности, были увеличены частоты синхронизации и расширена полоса пропускания монитора, уменьшен размер зерна люминофора, а также увеличены объем и быстродействие памяти видеоадаптера. Благодаря этому удалось повысить

разрешение и увеличить размер палитры. Так появились многочисленные расширения базового стандарта, получившие общее название *Super VGA* или *SVGA*.

Таблица 1.4. Номера и характеристики стандартных видеорежимов

Видеорежим		Тип видеоадаптера	Разрешение экрана, пикселы	Количество знакомест	Размер символьной матрицы, пикселы	Количество цветов или оттенков серого цвета
Номер	Тип					
00h	Текстовый	CGA	320×200	40×25	8×8	16 оттенков серого
		EGA	320×350	40×25	8×14	16
		VGA	360×400	40×25	9×16	16
01h	Текстовый	CGA	320×200	40×25	8×8	16
		EGA	320×350	40×25	8×14	16
		VGA	360×400	40×25	9×16	16
02h	Текстовый	CGA	640×200	80×25	8×8	16 оттенков серого
		EGA	640×350	80×25	8×14	16
		VGA	720×400	80×25	9×16	16
03h	Текстовый	CGA	640×200	80×25	8×8	16
		EGA	640×350	80×25	8×14	16
		VGA	720×400	80×25	9×16	16
04h	Графический	CGA	320×200	40×25	8×8	4 оттенка серого
		EGA/VGA	320×200	40×25	8×8	4
05h	Графический	CGA/EGA/VGA	320×200	40×25	8×8	4
06h	Графический	CGA/EGA/VGA	640×200	80×25	8×8	Моно
07h	Текстовый	MDA	720×350	80×25	9×14	Моно
		VGA	720×400	80×25	9×16	Моно
0Dh	Графический	EGA/VGA	320×200	40×25	8×8	16
0Eh	Графический	EGA/VGA	640×200	80×25	8×8	16
0Fh	Графический	EGA/VGA	640×350	80×25	8×14	Моно
10h	Графический	EGA/VGA	640×350	80×25	8×14	16
11h	Графический	VGA	640×480	80×25	8×16	Моно
13h	Графический	VGA	320×200	40×25	8×8	256

1.3.5. Super VGA

Поначалу совершенствование видеоадаптера VGA осуществлялось в основном путем увеличения объема его видеопамяти — сначала до 512 Кбайт, а затем и до 1 Мбайт. Появились видеоадаптеры, поддерживающие режимы 800×600, 1024×768 при одновременном отображении 256 оттенков цветов. Для работы с такими видеоадаптерами использовались модернизированные мониторы с уменьшенным размером зерна люминофора, повышенными частотами синхронизации и более широкой полосой пропускания видеотракта. Возникло понятие видеосистемы Super VGA, под которым поначалу понималось любое расширение возможностей стандартного адаптера VGA. Первой фирмой, изготовившей видеоадаптер, поддерживающий режим 800×600/256, была фирма NEC, а пионером среди видеоадаптеров, поддерживающих разрешение 1024×768, стал видеоадаптер 8514/A фирмы IBM.

Однако очень скоро возникла проблема совместимости адаптеров SVGA, изготовленных разными фирмами, с программным обеспечением. Такие адаптеры устойчиво работали только с теми программами, которые были написаны специально для них. При работе с другими программами видеосистема работала неустойчиво или не работала вовсе. Причина заключалась в том, что расширенные режимы работы видеоадаптера не имели стандартных команд инициализации. Каждая из фирм-производителей использовала свою нумерацию и команды инициализации для расширенных относительно VGA видеорежимов и оснащала видеоадаптеры собственной версией Video BIOS. Например, режим работы 800×600/256 видеоадаптеров фирмы Trident Microsystems имеет номер 5Eh, видеоадаптеров фирмы Realtek — 27h, а видеоадаптеров фирмы Tseng Labs — 30h. Следовательно, для установки режима 800×600/256 центральный процессор должен послать каждому из этих видеоадаптеров различные команды.

По этой причине долгое время нельзя было говорить о каком-либо общепринятом стандарте Super VGA. Более того, не был до конца ясен смысл самого термина SVGA. Многие фирмы, торгующие компьютерами, рекламировали свои PC, оснащенные обычными адаптерами VGA как SVGA-системы, основываясь на способности этих систем работать в режиме 800×600/16. При этом скромно умалчивалось то обстоятельство,

что необходимый для обеспечения такого режима объем видеопамяти составляет менее 256 Кбайт, т. е. видеоадаптер VGA со стандартным объемом памяти (256 Кбайт) в любом случае будет поддерживать данный режим.

Внести ясность в данный вопрос попыталась Ассоциация стандартов по видеоэлектронике VESA *(Video Electronics Standard Association),* предложив стандарт на нумерацию и способ инициализации расширенных относительно VGA видеорежимов. Упомянутый выше режим 800×600/16 было предложено отнести к стандартным, а SVGA-режимами считать те, которые требуют для своей реализации не менее 512 Кбайт видеопамяти. При этом автоматически становилось ясно, какой из видеоадаптеров можно причислить к семейству SVGA (рис. 1.21).

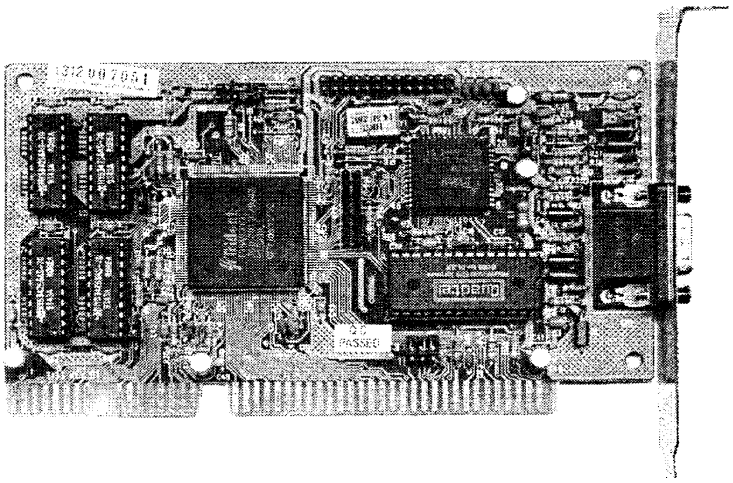


Рис. 1.21. Видеоадаптер SVGA

Для инициализации SVGA-режимов VESA предложила включить в состав Video BIOS ряд новых процедур (команд). В частности, для этой цели было предложено использовать все то же программное прерывание INT l0h с указанием параметра, соответствующего номеру режима (именно поэтому так важна единая нумерация видеорежимов!). Прикладная программа, которая должна установить такой видеорежим, запрашивает прерывание l0h и указывает в качестве параметра номер видеорежима в соответствии с табл. 1.5, предложенной VESA в качестве стандарта.

Таблица 1.5. Стандартные SVGA-режимы (согласно спецификации VESA версии 1.0)

Видеорежим		Разрешение	Количество цветов
Номер	Тип		
100h	Графический	640×400	256
101h	Графический	640×480	256
102h	Графический	800×600	16
103h	Графический	800×600	256
104h	Графический	1024×768	16
105h	Графический	1024×768	256
106h	Графический	1280×1024	16
107h	Графический	1280×1024	256
108h	Текстовый	80×60	16
109h	Текстовый	132×25	16
10Ah	Текстовый	132×43	16
10Bh	Текстовый	132×50	16
10Ch	Текстовый	132×60	16

Стандарт VESA имел несколько версий, появление которых отражало эволюцию возможностей видеосистемы. Данные, приведенные в табл. 1.5, соответствуют версии 1.0. В настоящее время используется стандарт версии 2.0, в который включены режимы с разрешением до 1600×1200 (см. разд. 2.2.4). Кроме того, в адаптерах SVGA начали использоваться 24-разрядные RAMDAC и прямая кодировка цвета пиксела, благодаря чему появились такие режимы, как High Color (16 бит на пиксел, или 65 536 оттенков цветов) и True Color (24 бит на пиксел, или 16 777 216 оттенков цветов).

Программы, дополняющие Video BIOS видеоадаптера SVGA и обеспечивающие поддержку спецификации VESA, получили название VBE *(VESA BIOS Extension).* Первоначально они использовались в виде драйверов и резидентных программ, загружаемых в память по мере необходимости. В дальнейшем производители видеоадаптеров стали включать эти программы в состав ROM Video BIOS, благодаря чему

совместимость со спецификацией VESA стала обеспечиваться автоматически.

Установить, поддерживает ли видеоадаптер спецификацию VESA, довольно легко. Для этого можно воспользоваться какой-либо тестовой программой, например Microsoft Diagnostic (MSD.EXE) из состава утилит MS DOS или Sysinfo из популярного пакета Norton Utilities, считывающей заголовок Video BIOS, в котором содержатся сведения о модели видеоадаптера, фирме-изготовителе и поддерживаемой версии спецификации VESA. Если видеоадаптер удовлетворяет этой спецификации, то в строке «тип видеоадаптера» будет сообщение «VESA-compatible» — VESA-совместимый с указанием версии стандарта (рис. 1.22).

```
------------------------------ Video ---------------------------------+
     Video Adapter Type: VGA                                            |
           Manufacturer: Unknown                                        |
                  Model:                                                |
           Display Type: VGA Color                                      |
             Video Mode: 3                                              |
      Number of Columns: 80                                             |
         Number of Rows: 25                                             |
     Video BIOS Version: S3 86C375/86C385 Video BIOS. Version 1.01.03   |
                         3 86C375/86C385 Video BIOS. Version 1.01.03    |
        Video BIOS Date: 12/13/96                                       |
 VESA Support Installed: Yes                                            |
           VESA Version: 1.02                                           |
          VESA OEM Name: S3 Incorporated. 86C375/86C385                 |
      Secondary Adapter: None                                           |
+-----------------------------------------------------------------------|
|                                OK   _                                 |
|                                ______                                 |
+-----------------------------------------------------------------------+
```

Рис. 1.22. Окно диалога программы Microsoft Diagnostic

Переход к более высокому разрешению и большей глубине цвета привел к резкому увеличению загрузки центрального процессора PC и шины ввода/вывода. Ведь сам по себе видеоадаптер VGA является пассивным устройством: он только преобразует в видеосигнал содержимое кадрового буфера. А вот заполнить этот буфер за время между двумя кадрами должен центральный процессор, используя для передачи информации шину ввода/вывода. Например, если установлен режим True Color (24 бит на пиксел), разрешение 1024×768 и частота кадров 70 Гц, то скорость передачи данных по шине ввода/вывода должна быть 1024×768×70×24 = 157,5 Мбайт/с. Полученная величина многократно превышает пропускную способность стандартной шины ISA. Да и CPU в этом случае будет занят исключительно обработкой и пересылкой видеоданных.

Для того чтобы разгрузить центральный процессор, ряд задач по построению изображения (заполнению кадрового буфера) решили возложить на специализированный набор микросхем *(Chipset)* видеоадаптера, получивший название *графического ускорителя (акселератора).* В результате скорость работы видеосистемы и PC в целом значительно повысилась. Акселератор на аппаратном уровне должен выполнять ряд функций по построению изображения, которые до него CPU был вынужден выполнять программным путем (т. е. значительно медленнее). Если акселератор допускал гибкое программирование своих функций, его называли *графическим сопроцессором.* Примерами таких систем стали видеоадаптеры TIGA *(Texas Instruments Graphics Architecture)* фирмы Texas Instruments на основе специализированных графических сопроцессоров TMS34010 и TMS34020 производства этой же фирмы.

Одновременно совершенствовалась и шина ввода/вывода. Появились скоростные 32-разрядные шины: локальная шина VLB *(VESA Local Bus)* и шина PCI *(Peripheral Component Interconnect—* соединение периферийных устройств), ставшая впоследствии стандартной шиной PC.

В результате этой эволюции облик видеосистемы VGA существенно изменился. Между тем основной принцип преобразования цифровых данных в изображение на экране монитора остался прежним. Более подробно устройство и принцип работы стандартного видеоадаптера VGA, а также его современных «могучих потомков» будет рассмотрено в гл. 2.

1.3.6. Видеосистема мультимедийного PC

В последние 7—10 лет стала очень популярной компьютерная обработка изображений. До недавнего времени такие задачи решались только при помощи специализированных графических станций (наиболее известны в этой области компьютеры фирмы Silicon Graphics). Однако стремительный рост производительности PC и повышение качества его видеосистемы сделали возможным цифровую обработку изображений в домашних условиях. Начался очередной бум мультимедиа, связанный с цифровой обработкой видеосигнала на PC. Видеосистема мультимедийного PC стала дополняться новыми компонентами, позволяющими значительно расширить ее функциональные возможности.

Появились платы захвата (оцифровки) статических изображений, а также устройства ввода/вывода

видеосигнала. Эти устройства позволяют получать цифровые изображения, используя в качестве источника аналоговый сигнал видеокамеры, видеомагнитофона или телевизора. Компьютеры стали оснащаться телевизионными тюнерами, выполненными в виде отдельных или встроенных в видеоадаптер карт расширения. В результате появилась возможность просмотра телевизионных программ на экране монитора PC и сохранения отдельных кадров и даже целых видеофрагментов в виде файлов на жестком диске.

Успехи в технологии сжатия видеоизображений позволили создать удобные и компактные носители цифровой видеоинформации — компакт-диски, — наибольшее распространение среди которых получили диски форматов Video CD и CD-I, неуклонно растет популярность DVD-дисков. Это позволило использовать PC, оснащенный приводом CD-ROM, в качестве проигрывателя видеодисков. Сочетание технологии цифровой обработки видеосигнала с возможностями сети Internet сделали обычным делом проведение видеоконференций. Домашний компьютер, оснащенный такими средствами, превратился в небольшую цифровую видеостудию.

Для обеспечения указанных возможностей потребовались устройства, выполняющие непривычные для PC функции: аналого-цифровое преобразование, компрессию и декомпрессию цифрового видеосигнала, преобразование сигнала из одной цветовой модели в другую и многое другое.

Первоначально эти задачи решались отдельными устройствами, подключаемыми к шине ввода/вывода. Однако в настоящее время большинство из них решается видеоадаптером, в состав которого входят необходимые элементы. В результате видеоадаптер мультимедийного PC из сравнительно простого и пассивного устройства превратился в миниатюрный, но весьма мощный графический процессор.

До настоящего момента речь шла о традиционной, двумерной, или 2D (2-*Dimentional* — двумерный), графике. Практически любое изображение, которое мы до недавнего времени видели на экране монитора, выглядело плоским. Его характеристиками являлись только разрешение и количество цветовых оттенков. Однако реальность, пусть даже и виртуальная, должна быть трехмерной, объемной! Чтобы создать такую виртуальную реальность, пришлось вновь совершенствовать видеосистему.

Построение объемных, трехмерных изображений — одна из последних тенденций развития видеосистемы PC. Переход к работе с трехмерной, или 3D-графикой *(3-Dimentional*— трехмерный), в очередной раз радикально изменил свойства и характеристики видеосистемы PC и мультимедийного, в частности. Появились ускорители трехмерной графики (3D-акселераторы), специальные устройства отображения трехмерных изображений и целый ряд других аппаратных средств. Устройства для отображения 3D-графики рассмотрены в [1, 2], а другие аппаратные средства 3D — в гл. 3.

На рис. 1.23 *а, б* и *в* приведены диаграммы, отражающие эволюцию видеосистемы PC с момента его появления и до 1995г. Поскольку в качестве количественных показателей выбраны традиционные параметры видеоадаптера, на данных графиках не выделена в качестве самостоятельного элемента видеосистема мультимедийного PC.

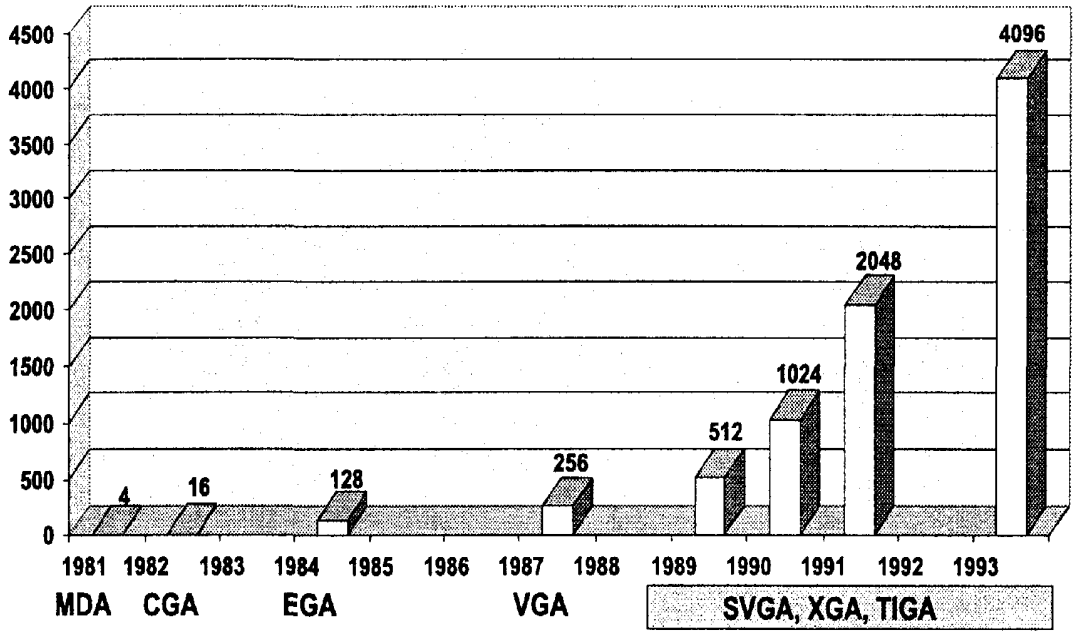


Рис. 123 *а*). Объем видеопамяти. Кбайт

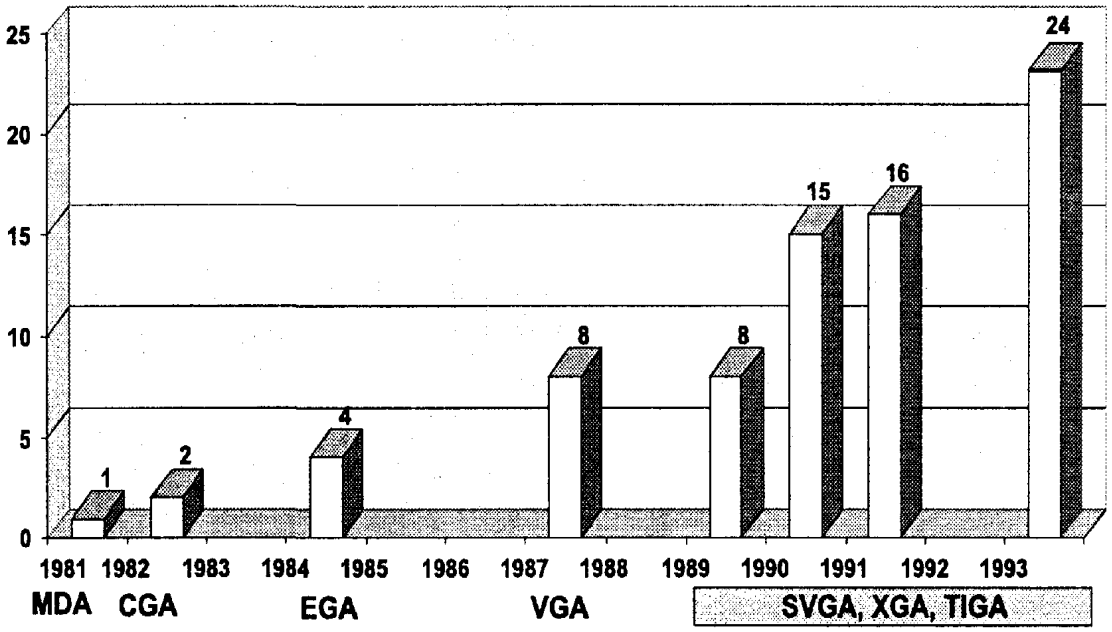


Рис. 1.23 *б*). Глубина представления цвета, бит/пиксел

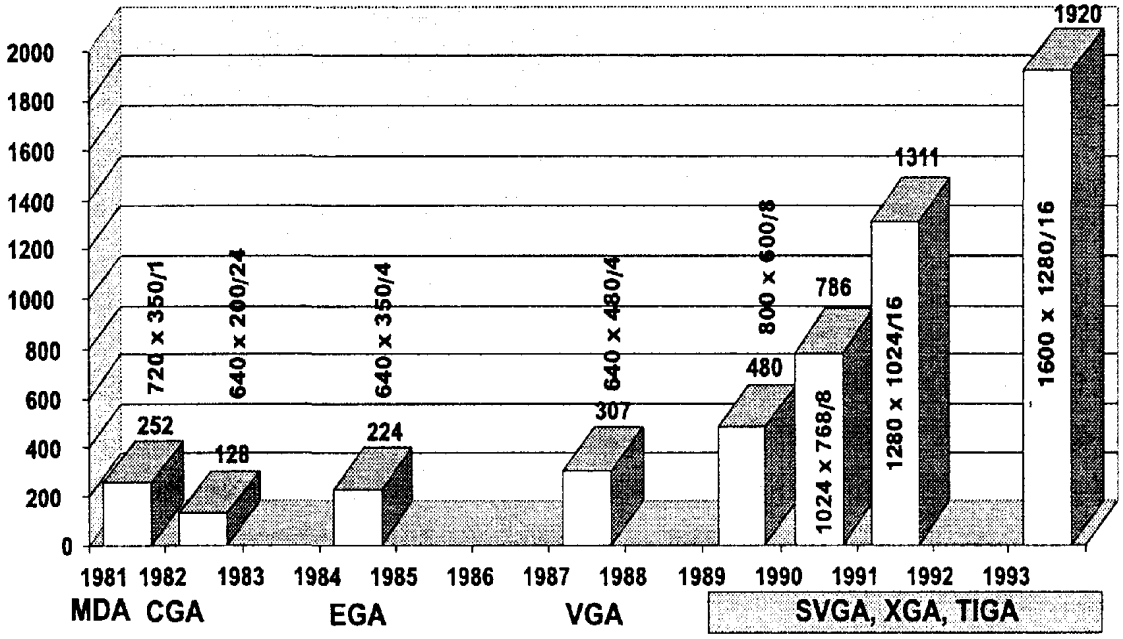


Рис. 1.23 *в*). Число пикселов растра при максимальном разрешении, тыс.

С 1995 года по настоящее время наблюдается значительный рост производительности как центральных компонентов PC (процессоры и память), его внутренних и внешних коммуникационных систем (системные шины, шины ввода-вывода, внешние интерфейсы), так и его видео системы: видеоадаптеры (видеоплаты) и устройства отображения. Основным тонусом этого развития являются мультимедийные приложения, особенно 3D-приложения и 3D-игровые программы.

Современная видеоплата по своей сложности соизмерима с современными материнскими платами, а ее центральное звено - графический процессор по своей архитектуре представляет собой даже более сложную систему, чем современные процессоры. Подтверждением последнему утверждению может являться сравнение числа транзисторов, входящих в их состав. Так процессор Petium 4 с ядром Norfhwood содержит 55 млн. транзисторов (январь 2002 г.), а графический процессор модельного ряда NVIDIA GeForct4 Ti - 63 млн. транзисторов (февраль 2002 г.).

В таблице 1.6 представлены основные параметры некоторых видеоплат, выполненных на базе различных графических процессоров (по состоянию на начало 2002 года). Более расширенную информацию можно найти в периодических изданиях (например, в [8]) и на соответствующих Web-узлах (см. табл. 1.6).

Таблица 1.6. Современные видеоплаты (2002 год).

Параметры	Gainward GeForce4 PowerPack! Ultra/ 750 XP Golden Sample	PowerColor AR2TO-B3 (Radeon 8500 Evil Master 11 275 MHz Pro Edition)	Hercules 3D Prophet 4500 TV/Out	AOpen GF3T1500-DV64	ASUS V817BSE/T
Ориентировочная розничная цена на середину апреля 2002 г., долл.	420	155	80	350	135
Графический процессор	GeForce4 Ti 4600	RADEON 8500	KYRO II	GeForce3 Ti 500	GeForce4 MX 420
Объем, Мбайт, и тип видеопамяти	128, DOR SDRAM	64, DDR SDRAM	64, SOR SDRAM	64, DDR SDRAM	64, SDR SDRAM
Версия BIOS	4.25.00.21.01	BK7.2.0 VR000.000. 007.002.000.001. 001 Ijfyy	Rev 01.107	3.20.0018	4.17.00.24
Тактовая частота графического процессора, МГц	306	275	175	240	250
Тактовая частота видеопамяти, МГц	662	549	175	501	167
Максимальная частота регенерации изображения в зависимости от разрешения при 8/16/32-бит глубине цвета, Гц	1024×768: 240/240/200; 1280×1024: 170/170/150; 1600×1200: 120/120/100; 1920×1440; 85/85/75; 2048×1536: 75/75/60	1024×768: 200/200/200; 1280×1024: 160/160/160; 1600×1200: 120/120/120; 1920×1440: 85/85/85; 2048×1536: 75/75/75	1024×768: 150/150/150: 1280×1024; 100/100/100; 1600×1200; 75/75/75; 1920×1440: 75/75/60	1024×768: 120/120/120; 1280×1024: 120/120/100; 1600×1200: 100/100/85; 1920×1440: 75/75/60; 2048×1536: 60/60/Н/П	1024×768: 240/240/200; 1280×1024: 170/170/170; 1600×1200: 120/120/100; 1920×1440: 85/85/75; 2048×1536: 75/75/75
Возможные режимы AGP	1X 2X 4X	1X 2X 4X	1X2X	1X 2X 4X	1X 2X 4X
Драйвер с Web-узла изготовителя	NVIDIA Detonator XP 28.32	ATI Radeon 6.13.10.6015 (Driver: 6.13. 10.6015)	Hercules 3D Prophet 4500 TV Out 15.20	AOpen GeForce3 Ti 500 27.50	ASUS V8170 SE V28.80
Изготовитель видеоплаты, адрес Web-узла	*Gainward, www.gainward. com*	*C.P. Technology, www.powercolor. com.tw*	*Hercules, www.hercules. com*	AOpen, *www.aopen. com*	ASUSTeK, *www.asus.com .tw*

Н/П — неприменимо.

Таблица 1.6. Современные видеоплаты (2002 год) (продолжение).

Параметры	ABIT Slluro GF3 Ti200	Gigabyte AR64S-H	Matrox Millennium G550	MSI G4T14600-VTD (MS-8872)	MSI G4MX460-VT (MS-8863)
Ориентировочная розничная цена на середину апреля 2002 г., долл.	155	80	110	395	155
Графический процессор	GeForce3 Ti 200	RADEON 7500LE	G550	GeForce4 Ti 4600	GeForce4 MX 460
Объем, Мбайт, и тип видеопамяти	128, DDR SDRAM	64, SDR SDRAM	32, DDR SDRAM	128, DDR SDRAM	64, DDR SDRAM
Версия BIOS	3.20.00.28	BK7.6.0 VR000.000. 007.006.000. 001.001 wfgb8	1.4b14	4.25.00.25.20	4.17.00.24.44
Тактовая частота графического процессора, МГц	175	230	133	300	300
Тактовая частота видеопамяти, МГц	401	170	333	662	551
Максимальная частота регенерации изображения в зависимости от разрешения при 8/16/32-бит глубине цвета, Гц	1024x768: 240/240/200; 1280x1024: 170/170/150; 1600x1200: 120/120/100; 1920x1440; 85/85/75; 2048x1536: 75/75/60	1024x768: 200/200/200 1280x1024; 160/160/160 1600x1200: 120/120/120 1920x1440; 85/85/85; 2048x1536: 75/75/75	1024x768: 160 и 160[1]; 1280x1024: 120 и 120; 1600x1200: 100 и 85; 1920x1440: 85 и Н/П; 2048x1536: 85 и Н/П	1024x768: 120/120/120; 1280x1024: 100/100/100; 1600x1200: 85/85/85: 1920x1200: 75/75/75; 2048x1536: 60/60/60	1024x768: 120/120/120; 1280x1024: 100/100/100; 1600x1200: 85/85/85; 1920x1200: 75/75/75; 2048x1536; 60/60/60
Возможные режимы AGP	1X 2X 4X	1X 2X 4X	1X 2X 4X	1X 2X 4X	1X 2X 4X
Драйвер с Web-узла изготовителя	ABIT Siluro GF3 Ti200 21.83	Gigabyte Radeon Ver 1.31 (Driver: 6.13.10.6015)	Matrox G550 AGP 5.82.018	MS-StarForce GeForce4 Ti 4600 Series 27.50	MS-StarForce GeForce4 MX 460 Series 27.50
Изготовитель видеоплаты, адрес Web-узла	A8IT, *www.abit.com. tw*	*Gigabyte, www.gigabyte. com.tw*	*Matrox, www.matrox. com*	*Micro-Star, www.msi.com. tw*	*Micro-Star, www.msi.com. tw*

[1] Для первого и второго мониторов при глубине цвета 16 бит.
Н/П — неприменимо.

В качестве примера на рис. 24 показана видеоплата ASUS V8460 Ti4600 GeForce 4 (2002г.), построенная на основе чипа графического процессора NVIDIA GeForce4 Ti4600 известной тайваньской фирмой ASUSTeK,.

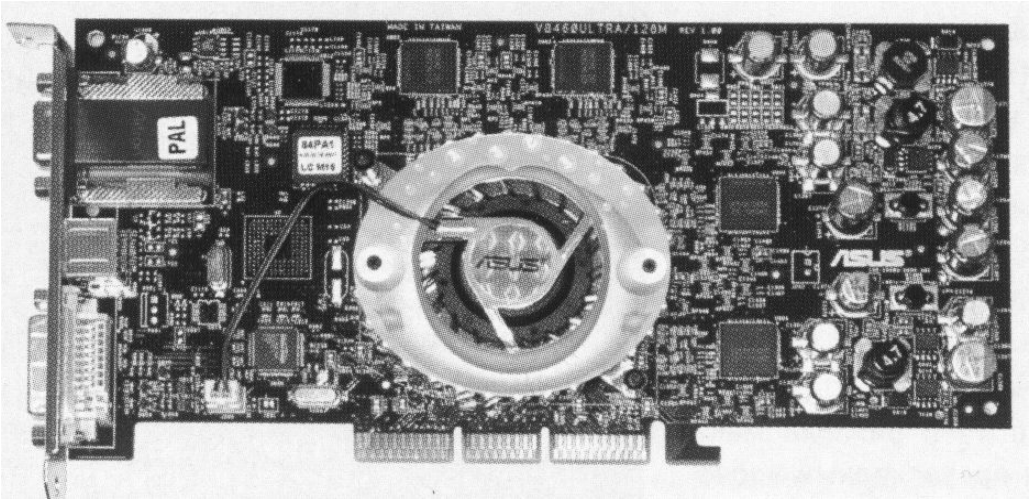


Рис.1.24. Видеоплата ASUS V8460 Ti4600 GeForce4.

Более подробно принципы работы и параметры современных видеоплат будут рассмотрены в главе 2, а графических процессоров - в главе 3.

Контрольные вопросы.

1. Что понимают под видеосистемой PC, и какие элементы входят в ее состав?
2. Перечислите основные архитектурные компоненты PC и дайте их краткую характеристику.
3. Охарактеризуйте основные компоненты видеосистемы PC.
4. Какие дополнительные компоненты включены в состав видеосистемы мультимедийного компьютера, и каково их назначение?
5. Перечислите и охарактеризуйте особенности видеосистемы ПК, благодаря которым она занимает особое место среди других подсистем ПК.
6. Какие компоненты входят в состав электронно-лучевой трубки?
7. Опишите принцип работы ЭЛТ.
8. Для чего нужна гамма-коррекция в ЭЛТ?
9. Почему телевизионные ЭЛТ называют кинескопами?
10. Какие особенности человеческого зрения учитывались при создании цветных ЭЛТ?
11. В чем заключается закон пространственного смешения цветов?
12. Каковы особенности конструкции цветной ЭЛТ?
13. Охарактеризуйте ЭЛТ с теневой маской.
14. Охарактеризуйте ЭЛТ со щелевой маской.
15. Охарактеризуйте ЭЛТ с апертурной решеткой.
16. Опишите процесс формирования растра изображения.
17. Какие особенности зрения учитываются при формировании растровых изображений движущихся предметов?
18. Что понимают под пикселом (пикселем)?
19. Чем определяется максимальное разрешение изображения на экране цветной ЭЛТ?
20. Какова связь между реальным разрешением монитора, параметрами развертки и характеристиками видеосигнала?
21. Когда применяется понятие "количество цветов", а когда - "количество оттенков цветов"?
22. Какие мониторы называются цифровыми или TTL-мониторами, и каковы их особенности?
23. Какие мониторы называются аналоговыми, и каковы их особенности?
24. Что понимается под фрейм-буфером видеоадаптера?
25. Что понимается под режимом работы видеоадаптера, а что под видеорежимом?
26. Какие типы видеорежимов может поддерживать видеоадаптер?
27. Что понимается под глубиной пиксела, и как она связана с размером цветовой палитры?
28. Почему графический режим иногда называют режимом APA?
29. Что понимается под dot rate (pixel rate) и dot clock?
30. Что понимается под знакоместом?
31. Что называется аппаратным знакогенератором, а что - шрифтом?
32. Что называется программным знакогенератором?
33. Каковы особенности работы видеоадаптера в текстовом режиме?
34. Чем идентифицируются символы в кодовых таблицах шрифтов?
35. Какова особенность первых 128 символов кодовых таблиц и символьного набора с номерами 128 -

255?

36. Что понимается под кодовой страницей и активной кодовой таблицей; каковы номера кодовых страниц России для DOS и Windows?
37. Охарактеризуйте кратко процесс формирования текстовой строки на экране видеомонитора.
38. Когда используется текстовый видеорежим в современных ПК?
39. Охарактеризуйте видеосистему MDA.
40. Кратко охарактеризуйте видеосистему CGA.
41. Как можно было подключить цветной телевизор к выходу видеоадаптера CGA?
42. Какие изменения были произведены в видеосистеме EGA по сравнению с CGA?
43. Как кодируется цвет в графическом режиме EGA?
44. Какую палитру может воспринимать человек?
45. Чем EGA отличается от EGA -плюс?
46. Когда были разработаны системы MDA, CGA, EGA и VGA?
47. Как расшифровывается аббревиатура VGA?
48. Какие дополнительные компоненты вошли в состав видеоадаптера VGA?
49. Как формируется палитра в VGA?
50. Какова отличительная особенность VGA-мониторов?
51. Что такое DDC?
52. Что такое "aspect ratio" и "квадратный пиксел"?
53. Что понимается под SVGA?
54. В связи с чем появилась возможность работы адаптеров SVGA в режимах High Color и True Color?
55. Как определить поддержку видеоадаптером спецификации VESA для SVGA?
56. Что понимается под графическим ускорителем и графическим сопроцессором?
57. Что характерно для видеосистемы мультимедийного компьютера?
58. Охарактеризуйте эволюцию видеосистемы ПК.

В данной главе были использованы материалы из [1, 3, 4, 8]. Достаточно подробное описание принципов работы видеомониторов, выполненных на базе электронно-лучевых трубок и плоских экранов можно найти в [1, 2, 7]. Информацию по видеосистемам HGC, HGC+ и MCGA, которые не рассматриваются в данном пособии, можно найти в [3, 15].

2. ВИДЕОАДАПТЕР МУЛЬТИМЕДИЙНОГО PC

Как было отмечено в разд. 1.1, вторым после монитора основным компонентом видеосистемы PC является видеоадаптер (видеокарта, видеоплата).

До недавнего времени термин «видеоадаптер» имел однозначное толкование: устройство, выполняющее единственную функцию — преобразование цифрового изображения, записанного в кадровый буфер, в аналоговый видеосигнал, подаваемый на монитор. Видеоадаптер VGA был пассивным устройством, поскольку работал под управлением центрального процессора и сам не принимал участия в формировании содержимого кадрового буфера.

Современный видеоадаптер для мультимедийного PC — это весьма сложное и вполне самостоятельное устройство, которое, наряду с выполнением своей основной функции, способно решать дополнительные задачи: аппаратное ускорение 2D- и 3D-графики, обработку видеоданных, прием телевизионных сигналов и многое другое. Эти дополнительные возможности обеспечиваются путем включения в архитектуру стандартного видеоадаптера VGA соответствующих элементов. Конструктивно эти элементы размещаются на отдельных картах расширения (дочерних платах) или встраиваются в один мощный графический процессор, являющийся основой универсальной графической видеоплаты. В настоящее время наиболее распространенным является второй вариант. По сложившейся традиции такие интегрированные изделия по-прежнему называют видеоадаптерами (например, в прайс-листах фирм, торгующих компьютерным оборудованием), однако при этом следует помнить, что по своим возможностям они значительно превосходят стандартный видеоадаптер VGA.

Таким образом, под видеоадаптером в этой главе будем понимать любое из упомянутых выше устройств. Как бы сильно ни изменялся облик современного видеоадаптера, основное его назначение остается прежним, таким же, как у стандартного видеоадаптера VGA: сканирование и цифро-аналоговое преобразование содержимого кадрового буфера с последующим формированием непрерывного трехкомпонентного RGB-сигнала.

Видеоадаптер является исключительно важным элементом видеосистемы, поскольку определяет следующие ее характеристики:

- максимальные разрешение и частоты разверток (совместно с монитором);
- максимальное количество отображаемых оттенков цветов;
- скорость обработки и передачи видеоданных.

Описание видеоадаптеров начнем с рассмотрения структуры и принципа действия стандартного видеоадаптера VGA. С одной стороны, это поможет лучше понять особенности архитектуры современного видеоадаптера, поскольку он содержит те же самые элементы стандартного адаптера VGA. С другой стороны, отличия архитектуры современных видеоадаптеров от базовой продиктованы стремлением устранить факторы, ограничивающие производительность видеосистемы VGA. В частности, для повышения производительности видеосистемы в современных видеоадаптерах используются:

- графические акселераторы операций по обработке данных двумерной и трехмерной графики;
- быстродействующая видеопамять;
- высокоскоростные шины ввода/вывода.

Кроме того, современные видеоадаптеры выполняют ряд дополнительных мультимедийных функций:

- воспроизведение цифровых видеоданных в различных форматах (например, MPEG);
- прием телевизионных программ и формирование телевизионного изображения на экране монитора;
- вывод информации на экран телевизора (функция TV-out);
- ввод/вывод аналоговых видеосигналов и др.

В данной главе мы рассмотрим структуру, принцип действия и способы повышения производительности видеосистемы и частично аппаратные средства для реализации мультимедийных функций.

2.1. Устройство и особенности работы стандартного видеоадаптера VGA

Первоначально видеоадаптер VGA появился не как самостоятельная плата, а как элемент видеосистемы, встроенный в материнскую плату компьютеров IBM PS/2 моделей 50 и 60 (напомним, что эти компьютеры выполнены на основе шины MCA и несовместимы с компьютерами семейства IBM PC). Строго говоря, именно это устройство следует считать *стандартным видеоадаптером VGA* фирмы IBM.

Однако очень скоро видеоадаптеры VGA появились в виде самостоятельных плат с интерфейсом ISA, предназначенных для использования в составе IBM PC. Помимо IBM, производством таких адаптеров занялось множество других фирм. Сохраняя неизменной общую архитектуру стандартного адаптера VGA, различные производители вносили в нее незначительные изменения и дополнения, позволявшие улучшить

характеристики базовой модели, в частности, увеличивали частоты тактовых генераторов с целью повышения быстродействия видеоадаптера и обеспечения более высокой частоты кадров. В результате появилось огромное количество так называемых *VGA-совместимых адаптеров.* В дальнейшем под видеоадаптером VGA будем понимать именно VGA-совместимый адаптер. Заметим, что улучшение характеристик VGA-совместимых адаптеров постепенно привело к появлению видеоадаптера Super VGA.

В общих чертах архитектура VGA-совместимого адаптера соответствует приведенной на рис. 1.1 структурной схеме стандартного видеоадаптера VGA. Вместе с тем он содержит элементы, которые отсутствуют в видеоадаптерах предыдущих типов (рис. 2.1).

Видеоадаптер VGA содержит следующие элементы:

- видеопамять;
- ROM Video BIOS;
- контроллер ЭЛТ, или CRTC *(Cathode Ray Tube Controller);*
- графический контроллер;
- контроллер атрибутов;
- секвенсор *(sequencer);*
- цифро-аналоговый преобразователь (ЦАП) данных, хранящихся в ОЗУ ЦАП, или RAMDAC *(Random Access Memory Digital-to-Analog Converter);*
- синхронизатор;
- тактовые генераторы;
- интерфейс с шиной ввода/вывода;
- 15-контактный D-образный разъем для подключения монитора.

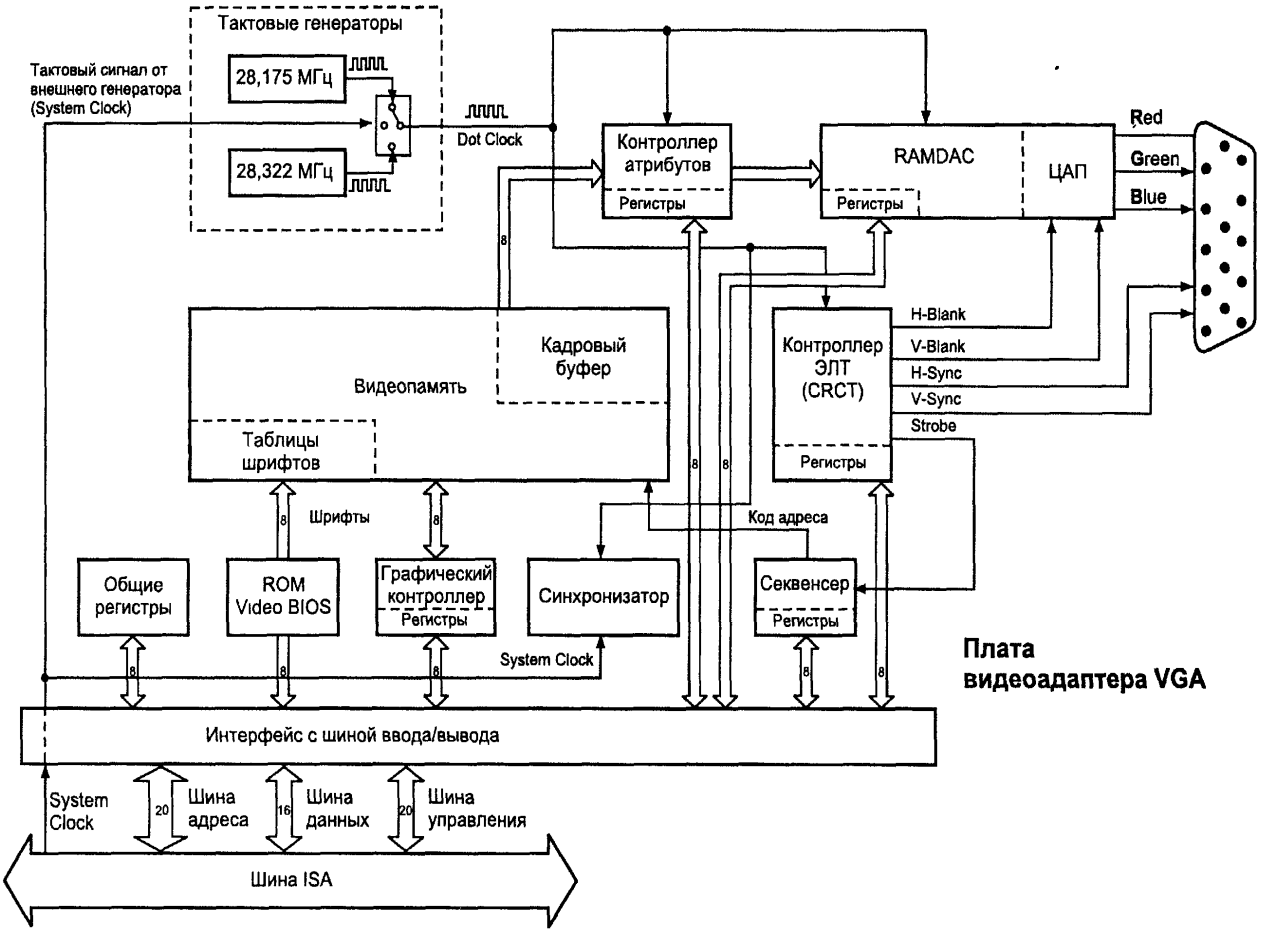


Рис. 2.1. Структурная схема видеоадаптера VGA

Большинство перечисленных элементов видеоадаптера содержат специальные 8-разрядные *регистры,* доступные центральному процессору для чтения/записи данных. Эти регистры, как правило, предназначены для управления работой соответствующих элементов видеоадаптера. Модифицируя содержимое этих регистров, CPU управляет работой видеоадаптера.

Помимо регистров в составе указанных элементов, видеоадаптер VGA имеет несколько отдельных регистров (внешние регистры), участвующих в процессе формирования видеосигнала, в том числе:

- *многоцелевой выходной регистр;* предназначен для задания адресов портов ввода/вывода элементов видеоадаптера, начального адреса кадрового буфера и выбора тактового генератора;

- *регистр состояния* (доступен только для чтения; используется для синхронизации процесса обновления кадрового буфера с сигналами обратного хода кадровой развертки).

Конструктивно основные элементы видеоадаптера VGA, за исключением видеопамяти, ROM Video BIOS, тактовых генераторов и интерфейса с шиной ввода/ вывода, реализованы в одной микросхеме.

2.1.1. Видеопамять

Видеопамять — это специализированное ОЗУ, размещенное на плате видеоадаптера и предназначенное для хранения цифрового образа формируемого изображения. Синонимами данного понятия являются термины *видеобуфер* и *Video RAM.* Отметим, что термин Video RAM (или VRAM) используется также для обозначения конкретного типа высокоскоростной двухпортовой видеопамяти, используемой в дорогих профессиональных видеоадаптерах (см. разд. 2.2.2). Видеопамять видеоадаптеров, в которых для построения изображения используется не только своя собственная, но и системная память PC (например, видеоадаптеры с интерфейсом AGP), часто называют *локальной,* подчеркивая тем самым место ее физического размещения.

Объем видеопамяти определяет максимальное разрешение и количество цветовых оттенков формируемого изображения (см. разд. 1.1). Стандартный видеоадаптер VGA имеет 256 Кбайт видеопамяти, что позволяет ему поддерживать графические режимы 640×480/16, 320×200/256, а при использовании специальной программной поддержки — 800×600/16 и 640×400/256 (последние два режима являются нестандартными и используются редко).

2.1.1.1. Логическая структура адресного пространства PC

Прежде чем излагать особенности доступа к видеопамяти, напомним логическую структуру адресного пространства PC.

Как известно, CPU обращается к различным устройствам компьютера, выставляя код адреса устройства на шине адреса. Общее количество адресов, которое может сформировать CPU, называют *размером адресного пространства.* Очевидно, что размер адресного пространства определяется разрядностью шины адреса. Поскольку первый персональный компьютер фирмы IBM — IBM PC — имел 20-разрядную шину адреса, размер его адресного пространства составлял 2^{20} = 1048576 = 1024^2, или 1024 К = 1 М (1 К = 2^{10}, 1 М = 2^{20}) различных адресов. В шестнадцатеричном коде этот диапазон, называемый *стандартным адресным пространством,* обозначается 00000—FFFFF. Первоначально основными адресуемыми объектами в IBM PC являлись 8-разрядные ячейки оперативной памяти, поэтому часто говорят, что размер адресного пространства IBM PC составляет $2^{20} \times 8$ = 1 Мбайт. Однако измерение размера адресного пространства в мегабайтах не совсем корректно, поскольку 1 Мбайт — это объем адресуемой памяти, а не количество ее адресуемых ячеек. Если память состоит, например, из 16-разрядных ячеек, то для адресации 1 Мбайт такой памяти достаточно не 1 М, а 2^{19}=512K адресов.

Для более эффективной адресации все адресное пространство условно делится на *сегменты* размером 2^{16}=64 К, в результате чего стандартное адресное пространство будет состоять из 16 сегментов. Удобство такого деления обусловлено тем, что внутренние регистры CPU первых моделей PC были 16-разрядными, поэтому наиболее быстро и просто адресация обеспечивалась именно в пределах сегмента. При использовании такой схемы адресации полный адрес ячейки памяти состоит из двух 16-разрядных чисел: *адреса сегмента,* определяющего положение начала сегмента, и *смещения,* равного порядковому номеру ячейки в пределах сегмента. Поскольку адрес сегмента 16-, а не 20-разрядный, то он определяет не фактический адрес первой ячейки сегмента, а только порядковый номер группы из 16 ячеек, в которую попадает начало сегмента. Поэтому сегмент всегда начинается в ячейке, адрес которой кратен 16, а фактический адрес начала сегмента получается путем умножения адреса сегмента на 16, что в шестнадцатеричной системе счисления эквивалентно добавлению нуля справа к коду адреса.

Адрес сегмента и смещение записывают в виде двух четырехразрядных шестнадцатеричных чисел, разделенных двоеточием. Например, запись A000:0FC2h обозначает ячейку, имеющую смещение 0FC2h в сегменте с начальным адресом A0000h. Фактический, 20-разрядный адрес ячейки определяется как сумма адреса сегмента, умноженного на 16 (10h), и смещения: A000h×10h + 0FC2h = A0000h + 0FC2h = A0FC2h.

Большая часть адресного пространства IBM PC в диапазоне 00000— 9FFFFh (первые 10 сегментов) была отведена для обращения к оперативной памяти: процессор мог адресовать 10×2^{16} = 640 К 8-разрядных ячеек памяти, т. е. 640 Кбайт. Область оперативной памяти, расположенная в этом диапазоне адресов, принято называть *обычной* или *стандартной памятью (Conventional Memory).* Остальные шесть сегментов, т. е. 6×2^{16}= 384 К адресов в диапазоне A0000 - FFFFFh, были зарезервированы для обращения к внешним устройствам, в том числе и к видеоадаптеру. Эту область адресного пространства называют *верхней памятью* или *UMA (Upper Memory Area).*

В начале 80-х годов, когда появились первые модели PC, размер стандартного адресного пространства казался более чем достаточным. Операционная система MS-DOS, ставшая основной для IBM PC, также была ориентирована на использование адресного пространства размером 1024 К = 1 М. Однако уже через несколько лет, с появлением процессора i80286, ограниченность стандартного адресного пространства стала очевидной. Имея 24-разрядную шину адреса, процессор i80286 мог адресовать 16 Мбайт памяти, но

заложенные в операционной системе MS-DOS ограничения не позволяли ему выйти за предел 1 Мбайт, т. е. задействовать для исполняемых программ оперативную память с адресами выше FFFFFh. Ограниченность стандартного адресного пространства стала еще более очевидной с появлением 32-разрядных процессоров (80386 и выше). Сложилась довольно нелепая ситуация: объем памяти PC значительно превысил 1 Мбайт, но эффективно использоваться не мог.

Для того чтобы эффективно использовать память с адресами выше FFFFFh, называемую *расширенной,* была разработана *спецификация расширенной памяти,* или *XMS (eXtended Memory Specification).* Доступ к расширенной памяти реализуется в так называемом *защищенном режиме работы* CPU (работа в пределах стандартного адресного пространства реализуется в *реальном режиме работы* CPU). Необходимую программную поддержку при работе с расширенной памятью в реальном режиме обеспечивают специальные драйверы — *менеджеры расширенной памяти.* Наиболее распространенным является драйвер HIMEM.SYS фирмы Microsoft, популярен также универсальный менеджер памяти QEMM фирмы Quartedeck.

Доступ к расширенной памяти из реального режима можно обеспечить используя одну из недокументированных возможностей процессоров Intel 80x86 - линейная адресация данных в реальном режиме DOS.

2.1.1.1.1. Линейная адресация данных в реальном режиме DOS.

В литературе по программированию описано три режима работы микропроцессоров серии 80x86 — реальный режим (режим совместимости с архитектурой 8086), защищенный режим и режим виртуальных процессоров 8086 (являющийся неким подвидом защищенного режима).

Основной недостаток реального режима состоит в том, что адресное пространство имеет размер всего в 1 Мбайт и при этом сегментировано — «нарезано» на кусочки размером по 64 Кбайт. Одного мегабайта очень мало для современных ресурсоемких прикладных программ (текстовых и графических редакторов, геоинформационных систем, систем проектирования и т.д.), а сегментация не позволяет нормально работать с видеопамятью и большими массивами данных.

Что можно сказать о защищенном и виртуальном режимах? Многие книги и учебники по микропроцессорам Intel *заканчиваются* главой «Переход в защищенный режим». Недостаток этого режима — необходимость *заново* создавать программное обеспечение для работы с периферийными устройствами на низком уровне, то есть фактически полностью переписывать *все* основные функции DOS. Можно, конечно, использовать Windows, но эта операционная система предназначена для офисных целей и плохо адаптируется к решению задач оперативного управления техническими системами. Кроме того, Windows забирает для собственных нужд изрядную часть ресурсов компьютера и ограничивает доступ к периферийным устройствам. В некоторых случаях универсальные многозадачные операционные системы типа Windows и Unix неприменимы по причинам, не относящимся напрямую к области вычислительной техники. Первая причина — лицензионные соглашения между изготовителями и потребителями программ. Прочтите внимательно любую лицензию: разработчик программы не несет ответственности *ни за что*! Следовательно, за все сбои и неисправности расплачивается потребитель. Например, за аварию в системе управления транспортом разработчикам этой системы придется отвечать по статьям Уголовного кодекса. Что касается систем военного назначения, то вообще сомнительно, что на таких лицензионных условиях какая-либо программа может быть официально принята в эксплуатацию на территории России.

Вторая причина — огромный объем универсальных операционных систем — десятки миллионов строк на языках высокого уровня! Полностью протестировать такие системы невозможно — у фирмы Microsoft, например, хватает сил только на доскональную проверку небольшого ядра Windows! Тем более на это не способен потребитель, у которого нет всей документации. Даже в случае открытой системы типа Linux, если документация есть и все исходные коды доступны — попробуйте *доказать* военным или банкирам, что в системе нет скрытых ловушек и «черного хода»!

Создать собственную программу для переключения в защищенный режим и работы в нем — непростая задача. При работе с аппаратурой в защищенном режиме программист должен четко понимать, какими возможностями аппаратуры пользоваться опасно. Например, приводимые в учебниках примеры программ для защищенного режима часто проявляют несовместимость с определенными конфигурациями оборудования, поскольку их авторы не имели достаточно широкой лабораторной базы для тестирования. Дело в том, что периферийные устройства всегда имеют какие-нибудь нестандартные особенности, добавляемые их изготовителями в рекламных целях. При работе в реальном режиме DOS такие особенности не применяются и потому никак не проявляются. Однако они могут показать себя с самой неприятной стороны при переключении в защищенный режим, когда программисту приходится перенастраивать периферийные устройства на новую модель организации оперативной памяти, перезаписывая при этом множество различных регистров аппаратуры. Возможно две ситуации: либо в стандартных регистрах некоторые разряды применяются нестандартным образом, но программисту об этом ничего не известно, либо вообще имеются какие-либо дополнительные регистры, не описанные в документации, но влияющие на режим работы системы. Возникает абсурдная ситуация: простой (реальный) режим работы задается процедурами BIOS фирмы-изготовителя системной платы, которая обычно хорошо осведомлена об особенностях применяемого на этой плате чипсета, а программы для перехода в защищенный режим вынуждены писать совершенно посторонние люди, не располагающие документацией в полном объеме. В BIOS включено

некоторое количество процедур для работы в защищенном режиме, но они охватывают лишь часть необходимых операций.

Вообще говоря, изобилие управляющих регистров в современных персональных компьютерах (их общее количество достигает нескольких тысяч) — явление совершенно ненормальное, теоретически приводящее к увеличению количества возможных режимов работы до бесконечности. Поскольку протестировать функционирование системы в миллиардах различных режимов технически невозможно, разработчики программного обеспечения не могут использовать дополнительные средства и ограничены несколькими общепринятыми (стандартными) режимами. Чтобы убедиться в этом, достаточно сравнить полный набор команд любого периферийного устройства с реально используемым (например, в BIOS) подмножеством команд данного набора. Большая часть регистров в настоящее время в принципе не нужна — установкой режима работы периферийного устройства должен заниматься его встроенный специализированный процессор, а не центральный процессор компьютера. Однако переход на новые технологии произойдет, вероятно, только после очередного кризиса в развитии компьютерной индустрии, а пока что приходится приспосабливаться к сложившейся ситуации.

Изложенные выше причины приводят к тому, что программисты вынуждены искать различные обходные пути. Один из возможных приемов — использование линейной адресации памяти. Линейная адресация — это наиболее простой, с точки зрения программиста, способ работы непосредственно с аппаратурой ЭВМ (логические адреса при этом совпадают с физическими). Различия в организации памяти в реальном, защищенном и линейном режимах работы процессора иллюстрирует рис. 2.2.

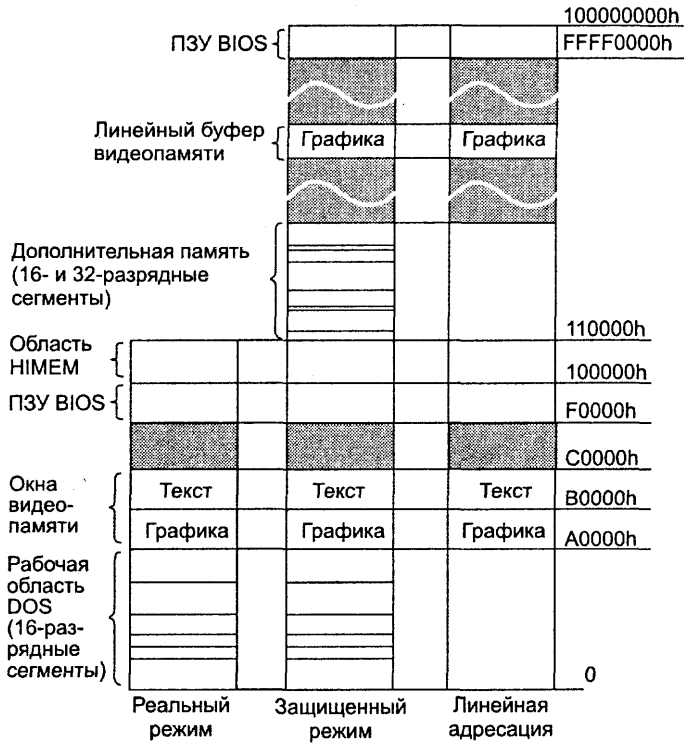


Рис. 2.2. Организация адресного пространства памяти в реальном, защищенном и линейном режимах работы процессора x86

Линейную адресацию можно использовать в специализированных программах, активно эксплуатирующих ресурсы ЭВМ — как в компьютерных играх, так и в системах автоматики, измерительных системах, системах управления, связи и т. п. Применение линейной адресации целесообразно в том случае, если проектируемая система предназначена для выполнения ограниченного, заранее известного набора функций и требует высокого быстродействия и надежности.

Разработчики процессоров начали внедрять линейную адресацию (в качестве одного из возможных режимов работы) при переходе с 16-разрядной архитектуры на 32-разрядную. Фирма Intel ввела такой режим в процессоре 80386, после чего он стал фактически стандартным (поддерживается не только всеми последующими моделями, но и всеми клонами архитектуры x86), однако остался недокументированным (почти не описан в литературе и не рассматривается в фирменном руководстве по программированию). Для пользователей обычных персональных компьютеров линейная адресация в чистом виде интереса не представляет по тем же причинам, что и защищенный режим: DOS и BIOS функционируют только в реальном режиме с 64-килобайтными сегментами, и при переходе в любой другой режим программист оказывается один на один с аппаратурой ЭВМ — без документации. Однако кроме чистых режимов процессоры Intel способны работать и в режимах гибридных. Еще в 1989 году Томас Роден (Thomas Roden)

предложил использовать интересную комбинацию сегментной (для кода и данных) и линейной (только для данных) адресации [11]. Предложенный им метод позволяет, находясь в обычном режиме DOS, работать со всей доступной памятью в пределах четырех гигабайтного адресного пространства процессора Intel 80386. Чтобы включить режим линейной адресации данных, необходимо снять ограничения на размер сегмента в теневом регистре, соответствующем одному из дополнительных сегментных регистров FS или GS (при необходимости описание архитектуры процессора Pentium вы можете найти в документации, размещенной в Интернете на сервере Intel для разработчиков). Через избранный регистр можно обращаться к любой области памяти с помощью прямой адресации или используя в качестве индексного любой 32-разрядный регистр общего назначения. После снятия ограничения запись в выделенный для линейной адресации сегментный регистр выполнять нельзя, иначе нарушится информация в соответствующем ему теневом регистре (предел сегмента сохранится, но начальный адрес будет перезаписан новым значением). Однако стандартные компиляторы и функции DOS с регистрами FS и GS не работают, и соответственно, при вызове процедур эти регистры можно вообще «не трогать» — их не нужно сохранять и восстанавливать. Достаточно один раз снять ограничение на размер адресного пространства, и после выхода из программы (до перезагрузки компьютера) линейную адресацию можно будет использовать из любой другой программы DOS, как поступил в своем примере Томас Роден.

Рассмотрим более подробно процедуру переключения одного из дополнительных сегментных регистров в режим линейной адресации. Каждый сегментный регистр состоит из видимой и невидимой (теневой) частей. Информацию в видимую часть можно записывать напрямую при помощи обычных команд пересылки данных (MOV и др.), а для записи в невидимую часть применяются специальные команды, которые доступны только в защищенном режиме. Теневая часть представляет собой так называемый дескриптор (описатель) сегмента, длина которого равна 64 разрядам.

При переходе от 16-разрядной архитектуры к 32-разрядной (то есть от i286 к i386) разработчики нового процессора попытались сохранить совместимость снизу вверх по структуре системных регистров, в результате чего дескрипторы сегментов приобрели довольно уродливый (с точки зрения технической эстетики) вид — поля предела и базового адреса разделены на несколько частей. Кроме того, поле предела оказалось ограничено 20 разрядами, что вынудило разработчиков применить еще один радиолюбительский трюк — ввести бит гранулярности G, чтобы можно было задавать размер сегмента, превышающий 16 Мбайт.

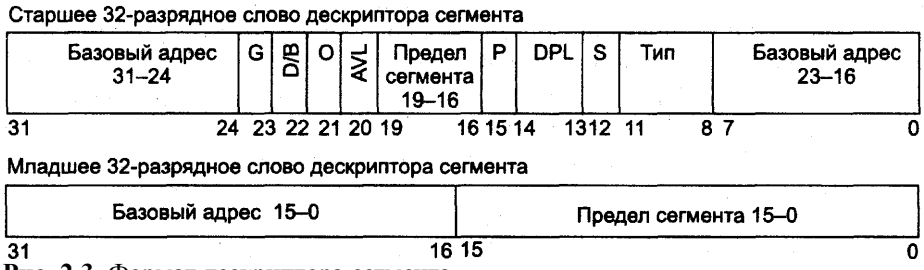


Рис. 2.3. Формат дескриптора сегмента

Формат дескриптора сегмента показан на рис. 2.3. Дескриптор состоит из следующих полей.

- Базовый адрес — 32-разрядное поле, задающее начальный адрес сегмента (в линейном адресном пространстве).
- Предел сегмента — 20-разрядное поле, которое определяет размер сегмента в байтах или 4-килобайтных страницах (в зависимости от значения бита гранулярности G). Поле предела содержит значение, которое должно быть на единицу меньше реального размера сегмента в байтах или страницах.
- Тип — 4-разрядное поле, определяющее тип сегмента и типы операций, которые допустимо с ним выполнять.
- Бит S — признак системного объекта (0 — дескриптор описывает системный объект, 1 — назначение сегмента описывается полем типа).
- DPL — 2-разрядное поле, определяющее уровень привилегий описываемого дескриптором сегмента.
- Бит P — признак присутствия сегмента в оперативной памяти компьютера (0 — сегмент «сброшен» на диск, 1 — сегмент присутствует в оперативной памяти).
- Бит AVL — свободный (available) бит, который может использоваться по усмотрению системного программиста.
- Бит D — признак используемого по умолчанию режима адресации данных (0 — 16-разрядная адресация, 1 — 32-разрядная).
- Бит G — гранулярность сегмента (0 — поле предела задает размер сегмента в байтах, 1 — в 4-килобайтных страницах).

В нашем случае признак используемого по умолчанию режима адресации данных D можно установить в 0 (использовать по умолчанию 16-разрядные операнды), но особой роли его значение не играет — в смешанном режиме сегментно-линейной адресации при работе с линейным сегментом строковые команды,

использующие значение этого разряда, применять нельзя. Бит гранулярности G должен быть установлен в 1, чтобы обеспечить охват всего адресного пространства процессора.

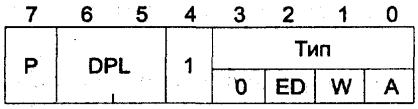


Рис. 2.4. Формат байта прав доступа для сегмента данных

Для сегментов данных формат байта прав доступа (включающего поле типа сегмента) имеет вид, показанный на рис. 2.4. Как видно из рисунка, поле S для сегментов данных должно быть установлено в 1, а старший разряд поля типа должен иметь значение 0. Поля P и DPL уже упоминались выше. Бит присутствия сегмента P следует установить в 1 (сегмент присутствует в памяти), а в поле DPL нужно установить максимальный уровень привилегий (значение 00). Бит расширения вниз ED для сегментов данных имеет значение 0 (в отличие от стековых сегментов, для которых ED=1). Бит разрешения записи W следует установить в единицу, чтобы можно было не только считывать, но и записывать информацию в сегмент. Бит A фиксирует обращение к сегменту и автоматически устанавливается в единицу всякий раз, когда процессор производит операции считывания или записи с сегментом, описываемым данным дескриптором. При инициализации регистра бит A можно сбросить в 0.

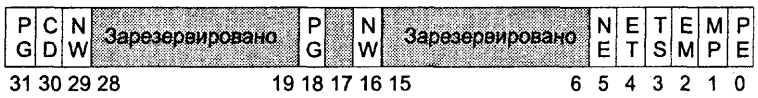


Рис. 2.5. Формат регистра управления CR0

Осуществить загрузку теневых регистров можно только в защищенном режиме. Для переключения режимов работы процессора используется регистр управления CR0, формат которого показан на рис. 2.5. Регистр CR0 содержит флаги, отражающие состояние процессора и управляющие режимами его работы. Назначение флагов следующее.

- PE (Protect Enable) — включение защищенного режима (0 — процессор работает в реальном режиме, 1 — в защищенном режиме).
- MP (Math Present) — признак наличия математического сопроцессора (0 — нет сопроцессора, 1 — сопроцессор присутствует в системе).
- EM (Emulation) — эмуляция математического сопроцессора (0 — выключена, 1 — включена).
- TS (Task Switched) — признак переключения задачи (флаг устанавливается в 1 при каждом переключении задач и проверяется перед выполнением команд математического сопроцессора).
- ET (Extension Type) — поддержка набора инструкций математического сопроцессора (0 — выключена, 1 — включена). В процессорах P6 флаг всегда установлен в 1,
- NE (Numeric Error) — встроенный механизм контроля ошибок математического сопроцессора (0— выключен, 1 — включен).
- WP (Write Protect) — защита от записи информации в страницы уровня пользователя из процедур супервизора (0 — выключена, 1 — включена).
- AM (Alignment Mask) — автоматический контроль выравнивания (0 — запрещен, 1 — разрешен).
- NW (Not Write-through) — запрещение сквозной записи (0 — сквозная запись разрешена, 1 — запрещена).
- CD (Cache Disable) — запрещение кэш-памяти (0 — использование кэш-памяти разрешено, 1 — запрещено).
- PG (Paging) — страничное преобразование (0 — запрещено, 1 — разрешено).

2.1.1.1.2. Подпрограммы установки режима линейной адресации

Набор подпрограмм, необходимых для переключения сегментного регистра GS в режим линейной адресации, показан в листинге 2.1. Как указано выше, перезапись содержимого теневого регистра процессора возможна только в защищенном режиме, а переход в этот режим, как видно из листинга, требует ряда дополнительных операций, выполняемых процедурой Initialization. В частности, нужно перенастроить на специально выделенные в кодовом сегменте области данных регистры DS, SS и SP. В момент перенастройки регистров стека должны быть запрещены прерывания, поскольку некоторые обработчики прерываний пишут информацию в стек прерываемой программы.

Процедура SetLAddrModeForGS, непосредственно осуществляющая перенастройку регистра GS в режим линейной адресации, воспроизводит (с незначительными изменениями) метод Родена. Прежде чем осуществить переключение, нужно вначале подготовить таблицу GDT (настроить на текущие сегменты кода и данных) и загрузить ее. Затем нужно войти в защищенный режим — установить в единицу бит PE регистра CR0, а остальные разряды сохранить без изменений (в том виде, в котором они находились при работе в реальном режиме). В защищенном режиме нужно перезагрузить сегментные регистры, сняв при этом

ограничения с GS, и сразу же вернуться в реальный режим DOS, сбросив в ноль бит PE. Длительное пребывание в защищенном режиме нежелательно, поскольку переключение в него выполнялось по упрощенной схеме: таблица прерываний не создавалась, а сами прерывания были просто отключены.

После выполнения процедуры SetLAddrModeForGS обязательно следует отменить заворачивание адресного пространства, то есть разблокировать адресную линию A20, которая управляется контроллером клавиатуры. Для этого необходимо послать в порт A контроллера соответствующую команду. Посылка команды осуществляется при помощи процедур Enable_A20 и Wait8042BufferEmpty.

Листинг 2.1. Подпрограмма, устанавливающая режим линейной адресации данных (list2_01.inc)

```
; Порт, управляющий запретом немаскируемых прерываний
CMOS_ADDR       equ  0070h
CMOS_DATA       equ  0071h
; Селекторы сегментов
SYS_PROT_CS     equ  0008h
SYS_REAL_SEG    equ  0010h
SYS_MONDO_SEG   equ  0018h
CODESEG
;*********************************************
;* ВКЛЮЧЕНИЕ РЕЖИМА ЛИНЕЙНОЙ АДРЕСАЦИИ ПАМЯТИ *
;*       (процедура параметров не имеет)     *
;*********************************************
PROC Initialization NEAR
        pushad
; Сохранить значения сегментных регистров в
; реальном режиме (кроме GS)
        mov     [CS:Save_SP],SP
        mov     AX,SS
        mov     [CS:Save_SS],AX
        mov     AX,DS
        mov     [CS:Save_DS],AX
; (работаем теперь только с кодовым сегментом)
        mov     AX,CS
        mov     [word ptr CS:Self_Mod_CS],AX
        mov     DS,AX
        cli
        mov     SS,AX
        mov     SP,offset Local_Stk_Top
        sti

; Установить режим линейной адресации
        call    SetLAddrModeForGS

; Восстановить значения сегментных регистров
        cli
        mov     SP,[CS:Save_SP]
        mov     AX,[CS:Save_SS]
        mov     SS,AX
        mov     AX,[CS:Save_DS]
        mov     DS,AX
        sti
; Разрешить работу линии A20
        call    Enable_A20
        popad
        ret
ENDP Initialization

; Область сохранения значений сегментных регистров
Save_SP DW ?
Save_SS DW ?
Save_DS DW ?
; Указатель на GDT
GDTPtr  DQ ?
; Таблица дескрипторов сегментов для
; входа в защищенный режим
GDT DW 00000h,00000h,00000h,00000h ;не используется
    DW 0FFFFh,00000h,09A00h,00000h ;сегмент кода CS
    DW 0FFFFh,00000h,09200h,00000h ;сегмент данных DS
    DW 0FFFFh,00000h,09200h,0008Fh ;сегмент GS
; Локальный стек для защищенного режима
; (организован внутри кодового сегмента)
label GDTEnd word
        DB 255 DUP(0FFh)
Local_Stk_Top DB (0FFh)
```

Листинг 2.1 (продолжение)

```
;*********************************************
;*        ОТМЕНИТЬ ПРЕДЕЛ СЕГМЕНТА GS        *
;* Процедура изменяет содержимое теневого     *
;* регистра GS таким образом, что становится *
;* возможной линейная адресация через него    *
;* 4 Gb памяти в реальном режиме              *
;*********************************************
PROC SetLAddrModeForGS near
; Вычислить линейный адрес кодового сегмента
        mov     AX,CS
        movzx   EAX,AX
        shl     EAX,4   ;умножить номер параграфа на 16
        mov     EBX,EAX ;сохранить линейный адрес в EBX
; Занести младшее слово линейного адреса в дескрипторы
; сегментов кода и данных
        mov     [word ptr CS:GDT+10],AX
        mov     [word ptr CS:GDT+18],AX
        ; Переставить местами старшее и младшее слова
        ror     EAX,16
; Занести биты 16-23 линейного адреса в дескрипторы
; сегментов кода и данных
        mov     [byte ptr CS:GDT+12],AL
        mov     [byte ptr CS:GDT+20],AL
; Установить предел (Limit) и базу (Base) для GDTR
        add     EBX, offset GDT
        mov [word ptr CS:GDTPtr],(offset GDTEnd-GDT-1)
        mov     [dword ptr CS:GDTPtr+2],EBX
; Сохранить регистр флагов
        pushf
; Запретить прерывания, так как таблица прерываний IDT
; не сформирована для защищенного режима
        cli
; Запретить немаскируемые прерывания NMI
        in      AL,CMOS_ADDR
        mov     AH,AL
        or      AL,080h      ;установить старший разряд
        out     CMOS_ADDR,AL ;не затрагивая остальные
        and     AH,080h
        ; Запомнить старое состояние маски NMI
        mov     CH,AH
; Перейти в защищенный режим
        lgdt    [fword ptr CS:GDTPtr]
        mov     BX,CS    ;запомнить сегмент кода
        mov     EAX,CR0
        or      AL,01b   ;установить бит PE
        mov     CR0,EAX  ;защита разрешена
        ; Безусловный дальний переход на метку SetPMode
        ; (очистить очередь команд и перезагрузить CS)
                DB      0EAh
                DW      (offset SetPMode)
                DW      SYS_PROT_CS
        mov     ES,AX
        mov     FS,AX
        ; Снять ограничения с сегмента GS
        mov     AX,SYS_MONDO_SEG
SetPMode:
        ; Подготовить границы сегментов
        mov     AX,SYS_REAL_SEG
        mov     SS,AX
        mov     DS,AX
```

```
        mov     GS,AX
; Вернуться в реальный режим
        mov     EAX,CR0
        and     AL,11111110b ;сбросить бит PE
        mov     CR0,EAX      ;защита отключена
        ; Безусловный дальний переход на метку SetRMode
        ; (очистить очередь команд и перезагрузить CS)
            DB 0EAh
            DW (offset SetRMode)
Self_Mod_CS DW ?

SetRMode:
        ; Регистры стека и данных
        ; настроить на сегмент кода
        mov     SS,BX
        mov     DS,BX
        ; Обнулить дополнительные сегментные
        ; регистры данных (GS не трогать!)
        xor     AX,AX
        mov     ES,AX
        mov     FS,AX
        ; Возврат в реальный режим,
        ; прерывания снова разрешены
        in      AL,CMOS_ADDR
        and     AL,07Fh
        or      AL,CH
        out     CMOS_ADDR,AL
        popf
        ret
ENDP SetLAddrModeForGS
```

```
;*************************************
;* Разрешить работу с памятью выше 1 Мб *
;*************************************
PROC Enable_A20 near
        call    Wait8042BufferEmpty
        mov     AL,0D1h ;команда управления линий A20
        out     64h,AL
        call    Wait8042BufferEmpty
        mov     AL,0DFh ;разрешить работу линии
        out     60h,AL
        call    Wait8042BufferEmpty
        ret
ENDP Enable A20
;*********************************************
;*   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
```

ВНИМАНИЕ

Как уже было указано выше, после выхода из защищенного режима нельзя перезаписывать регистр GS, иначе будет полностью или частично стерта информация в соответствующем теневом регистре. В частности, нельзя выполнять операции сохранения/восстановления содержимого регистра при помощи команд работы со стеком push и pop.

2.1.1.1.3. Использование области верхней памяти видеосистемой PC

Логическая структура адресного пространства современного PC представлена на рис. 2.6.

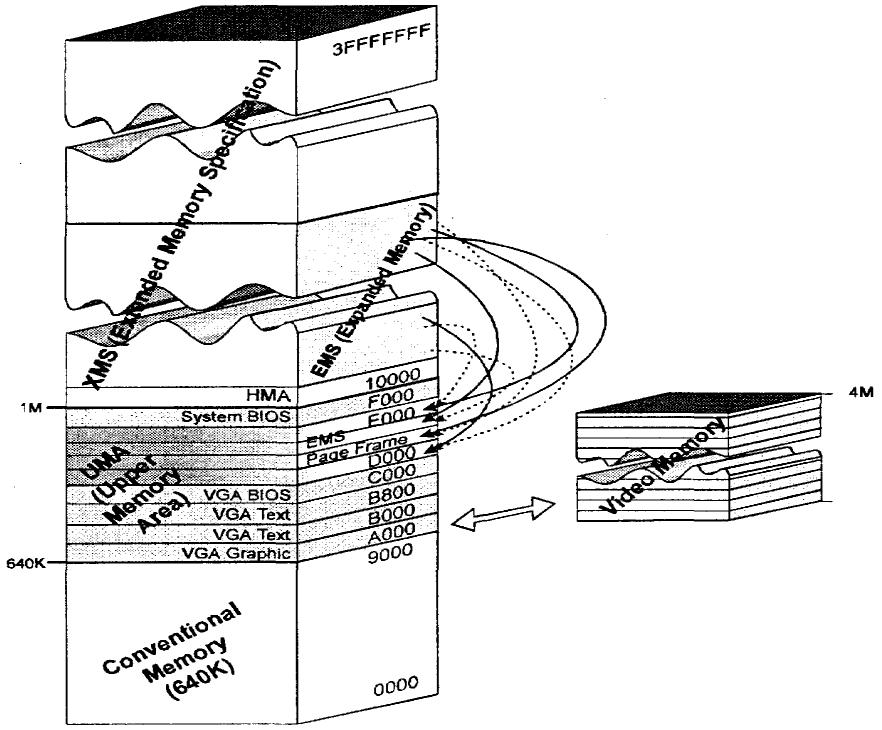


Рис. 2.6. Логическая структура адресного пространства PC

Кратко рассмотрим, как используется область верхней памяти видеосистемой PC (табл. 2.1). Первые два сегмента UMA отведены для обращения к видеопамяти. Выставляя на шине адреса код адреса в диапазоне A0000—BFFFFh, CPU получает доступ к кадровому буферу, таблицам загружаемых шрифтов и другим данным, хранящимся в локальной памяти видеоадаптера. Следующий сегмент отведен для Video BIOS. Заметим, что видеосистемы различных типов по-разному используют это адресное пространство.

Таблица 2.1. Распределение адресного пространства между видеопамятью и Video BIOS

Тип видеосистемы	Видеопамять		Video BIOS	
	Размер, Кбайт	Диапазон адресов	Размер, Кбайт	Диапазон адресов
MDA	4	B0000-B0FFFh	-	-
CGA	16	B8000-BBFFFh	-	-
EGA	128	A0000-BFFFFh	16	C0000-C3FFFh
VGA	128	A0000-BFFFFh	32	C0000-C7FFFh

Видеоадаптеры MDA и CGA не имеют собственного ROM Video BIOS (программы, реализующие видеосервис BIOS, содержатся в ROM BIOS PC) и задействуют различные области адресного пространства, отведенного для нужд видеосистемы. Это позволяет, во-первых, использовать линейную адресацию ячеек видеопамяти и, во-вторых, — использовать в PC одновременно две видеосистемы: MDA и CGA.

Видеоадаптер VGA в основных графических режимах работы полностью использует адресное пространство, отведенное для обращения к видеопамяти. Однако в текстовых, а также в графических режимах с низким разрешением он использует те же области видеопамяти, что и упомянутые выше видеоадаптеры MDA и CGA. Это сделано для совместимости с программами, ориентированными на использование видеосистем MDA и CGA.

Видеоадаптер VGA имеет ROM Video BIOS емкостью 32 Кбайт, содержимое которого в общем адресном пространстве PC размещается по адресам C0000 - C7FFFh. Современные видеоадаптеры SVGA с интерфейсом PCI позволяют назначить видеопамяти и Video BIOS произвольные адреса, что дает возможность установить в PC два видеоадаптера и подключить два монитора.

2.1.1.2. Организация видеопамяти

Область видеопамяти, используемая для хранения цифрового образа изображения, называется *кадровым буфером (frame buffer).* Как правило, размер кадрового буфера меньше, чем объем видеопамяти. Например, в видеорежиме 640×480/16 кадровый буфер занимает 150 Кбайт из 256 доступных. Помимо кадрового буфера, в видеопамяти хранятся и другие данные, такие, как, например, загружаемые национальные шрифты.

Количество адресуемых ячеек видеопамяти (256 К) превышает адресное пространство (128 К), отведенное операционной системой для обращения к видеопамяти. Поэтому в графических режимах, для реализации которых требуется более 128 Кбайт видеопамяти, линейная адресация кадрового буфера, использовавшаяся в видеоадаптерах MDA и CGA, невозможна. Для того чтобы CPU мог обратиться к любой ячейке, видеопамять разбивается на отдельные области, именуемые *банками* или *страницами* (обычно на четыре банка размером 64 Кбайт каждый) CPU формирует код адреса в пределах отведенного диапазона, обращаясь тем самым к одноименным ячейкам в четырех различных банках одновременно. Этот код (только младшие 16 разрядов) трактуется видеоадаптером как смещение ячейки видеопамяти в пределах одного банка. Номер этого банка определяется содержимым регистров графического контроллера и секвенсера (см. разд. 2.14). Каждый раз перед обращением к видеопамяти содержимое этих регистров обновляется. Такой метод адресации видеопамяти используется в 16-цветных графических режимах работы видеоадаптера.

Метод, используемый для адресации видеопамяти, определяется способом хранения информации о цвете пиксела. В графическом режиме она может храниться в формате *Packed Display Format* (PDF) или в виде цветовых плоскостей.

2.1.1.2.1. Формат PDF

Если для кодирования цвета пиксела используется 8 бит (видеорежим 13h), то применяется метод PDF — формат пакетного (компактного) хранения содержимого кадрового буфера. Согласно формату PDF, номер цвета пиксела целиком хранится в одной 8-разрядной ячейке кадрового буфера (рис. 2.7). Поскольку разрешение в видеорежиме 13h составляет 320×200 (на экране формируется 64 000 пикселов), для размещения кадрового буфера понадобится только один банк видеопамяти размером 64 Кбайт. Поэтому в формате PDF используется линейная адресация кадрового буфера, не требующая переключения банков. При выполнении операций чтения/записи модифицируется только один пиксел, следовательно, скорость обновления кадрового буфера невелика.

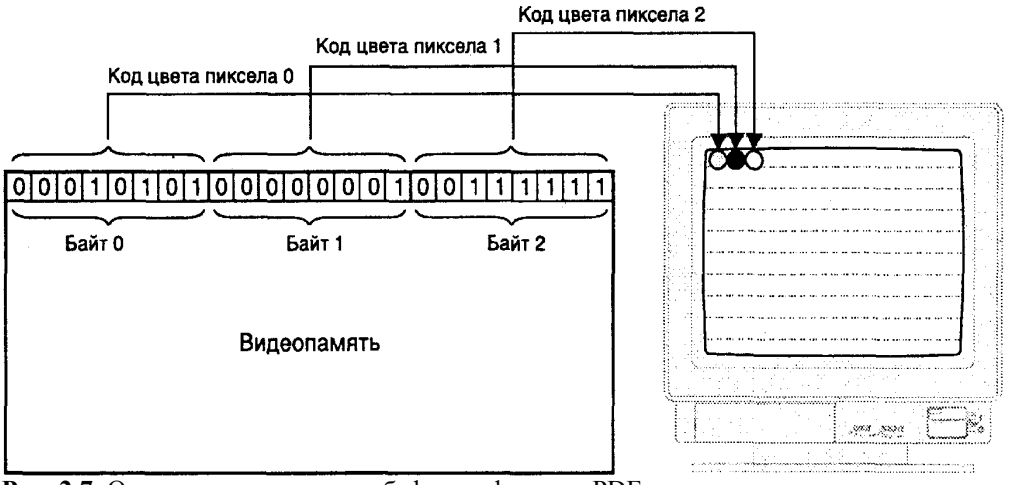


Рис. 2.7. Организация кадрового буфера в формате PDF

2.1.1.2.2. Метод цветовых плоскостей

Метод цветовых плоскостей для организации кадрового буфера первоначально использовался в видеоадаптерах CGA и EGA. *Цветовая плоскость* — это участок видеопамяти, представляющий собой матрицу бит, каждый из которых соответствует отдельному пикселу на экране ЭЛТ. Адресуемым элементом такой плоскости является не байт, а бит. Количество цветовых плоскостей равно количеству бит, отводимых для кодирования цвета пиксела (2 или 4), а их размеры соответствуют разрешению экрана. Например, в режиме 12h видеоадаптера VGA (640×480/16) используются четыре битовых плоскости размером 640×480 бит, при этом одновременно отображается $2^4 = 16$ цветов (рис. 2.8).

Основное преимущество метода битовых плоскостей состоит в том, что биты, принадлежащие разным плоскостям, но относящиеся к одному и тому же пикселу изображения (т. е. имеющие в пределах цветовой плоскости одинаковые координаты), имеют одинаковое смещение и поэтому обновляются центральным процессором *одновременно*. Посылая в видеопамять один байт информации по выбранному адресу, CPU тем самым записывает в кадровый буфер коды цвета 2 или 4 пикселов. Очевидно, что скорость обновления видеобуфера при использовании метода цветовых плоскостей выше, чем при использовании метода PDF. Для управления чтением/записью информации в каждую битовую плоскость используется специальный буферный регистр графического контроллера — *регистр-защелка.*

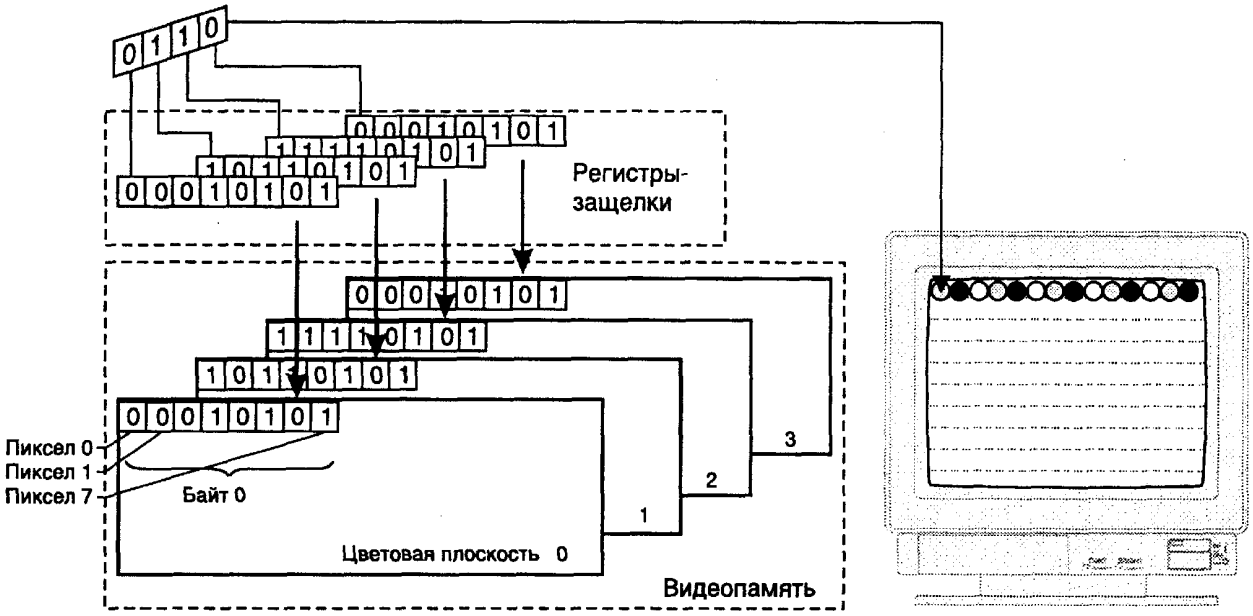


Рис. 2.8. Организация видеопамяти с использованием цветовых плоскостей

2.1.1.3. Разрядность видеопамяти

Важной характеристикой видеопамяти является разрядность ее ячеек, которая определяет разрядность *внутренней шины данных* видеоадаптера. В видеоадаптере VGA использовалась 8-разрядная внутренняя шина, а в видеоадаптерах SVGA — сначала 16-, а затем 32-разрядная шина. В современных видеоадаптерах используется 64- или 128-разрядная внутренняя шина, а наиболее совершенные модели видеоадаптеров имеют 256-разрядную шину данных.

2.1.2. Video BIOS (VGA BIOS)

Как уже отмечалось, все операции по модификации содержимого видеопамяти и регистров видеоадаптера VGA выполняет центральный процессор. Реализация любой графической операции (смена видеорежима, чтение/запись информации в кадровый буфер, управление курсором и т. п.) требует от центрального процессора выполнения весьма длинной последовательности элементарных команд (чтение/запись, перемещение данных в регистрах и др.). Кроме того, во время этих действий в регистры видеоадаптера заносятся константы, значения которых для различных моделей VGA-совместимых адаптеров могут различаться. С целью упрощения процедуры программирования видеоадаптера, а также для обеспечения совместимости аппаратного и программного обеспечения все наборы команд (подпрограммы) CPU, реализующие графические функции, помещаются в специальное ПЗУ, расположенное на плате видеоадаптера VGA. Эти подпрограммы образуют так называемое *расширение базовой системы ввода/вывода (BIOS extension)* для решения задач вывода изображения на экран монитора и называются *Video BIOS.*

Фактически Video BIOS — это набор подпрограмм, написанных в кодах команд центрального процессора и предназначенных для реализации основных функций видеосистемы. Набор этих функций принято называть *видеосервисом.* Кроме того, Video BIOS содержит множество данных (констант), необходимых для работы в различных видеорежимах, а также сведения о производителе, модели и возможностях видеоадаптера.

Video BIOS хранится в специальном ПЗУ (ROM), размещенном на плате видеоадаптера VGA. Его емкость составляет 32 Кбайт. При включении компьютера первыми получают управление именно программы Video BIOS и лишь затем программа тестирования оборудования компьютера POST *(Power-On Self Test),* содержащаяся в ROM BIOS PC. Именно поэтому первые сообщения, появляющиеся на экране монитора при включении PC, содержат сведения о видеоадаптере, версии Video BIOS и фирме-производителе.

ROM Video BIOS видеоадаптера VGA является 8-разрядным. Низкое быстродействие ПЗУ, по сравнению с оперативной памятью PC, замедляет работу видеосистемы и компьютера в целом. Для ускорения работы практически во всех PC предусмотрена возможность копирования (разрешается при помощи программы конфигурирования CMOS Setup) содержимого ROM BIOS и ROM Video BIOS в специально отведенную область оперативной памяти PC, называемую *теневой памятью (Shadow Memory).* Использование механизма теневой памяти увеличивает быстродействие видеосистемы, а также позволяет при необходимости программным путем заменить Video BIOS на более новую версию.

2.1.2.1. Видеосервис BIOS

Видеосервис Video BIOS (в дальнейшем для краткости будем использовать термин «видеосервис BIOS») реализуется с помощью механизма *программных прерываний,* т. е. путем выдачи прикладной программой центральному процессору команды INT на прерывание с указанием номера прерывания. Напомним, что *прерывание (interrupt)* — это изменение нормального хода выполнения центральным процессором текущей программы и переход к выполнению более приоритетной, называемой *программой обработки прерывания.* После завершения программы обработки прерывания CPU возвращается к выполнению приостановленной программы. Архитектурой PC предусмотрены аппаратные и программные прерывания.

2.1.2.1.1. Аппаратные прерывания

Аппаратные прерывания инициализируются различными устройствами PC (клавиатурой, мышью и др.) путем подачи на CPU электрического сигнала по специальным линиям, называемым *линиями запроса на прерывание IRQ (Interrupt Request Line).* В IBM PC/AT предусмотрено 15 линий IRQ (IRQ0 - IRQ15, (по линии IRQ2 осуществляется каскадное соединение контроллеров прерываний). Номер линии, по которой пришел запрос на прерывание, однозначно определяет обслуживающую данное прерывание программу. В частности, для обработки событий, связанных с перемещением курсора мыши, традиционно используется прерывание IRQ4, для клавиатуры - IRQ1, для принтера, подключенного к порту LPT1, - IRQ7 и т. д.

2.1.2.1.2. Программные прерывания

Программные прерывания реализуются путем подачи прикладной программой команды INT центральному процессору с указанием в ней номера прерывания. Перед подачей этой команды прикладная программа заносит номер функции в регистр AX (первый из *регистров общего назначения* процессоров Intel 8088 и 80x86). Обычно в рамках одного прерывания доступно несколько *функций,* при этом отдельные функции могут содержать несколько *подфункций.* Для выбора необходимой подфункции ее номер перед вызовом прерывания также заносится в соответствующий регистр общего назначения CPU.

Номер прерывания (аппаратного или программного) однозначно связан с адресом программы обработки прерывания, которую должен вызвать центральный процессор при инициализации прерывания. Для простой и эффективной локализации процедуры обслуживания прерываний служит *таблица векторов прерываний,*

содержащаяся в первых ячейках оперативной памяти. В каждой строке этой таблицы (четыре байта) содержится адрес программы обработки прерывания, а порядковый номер строки таблицы соответствует номеру прерывания. С точки зрения языков программирования высокого уровня (PASCAL, FORTRAN), процедура инициализации прерывания подобна процедуре вызова подпрограммы из основной (главной) программы.

2.1.2.2. Функции Video BIOS

Видеосервис BIOS реализуется при помощи программного прерывания INT 10h. Прерывание 10h имеет несколько десятков функций управления изображением на экране монитора (табл. 2.2).

Таблица 2.2. Основные функции Video BIOS

Номер функции	Описание функции
00h	Установка видеорежима с номером N (N = 0... 13h)
01h	Установка типа курсора (задание формы курсора)
02h	Установка позиции курсора (в координатах (строка, столбец))
03h	Считывание текущей позиции курсора
04h	Считывание позиции манипулятора типа «световое перо» (в видеоадаптере VGA не используется)
05h	Задание номера активной страницы (в текстовом режиме)
06h	Прокрутка окна вверх
07h	Прокрутка окна вниз
08h	Считывание символа и его атрибутов в текущей позиции курсора
09h	Запись символа и его атрибутов в позицию, в которой находится курсор
0Ah	Запись символа в позицию, в которой находится курсор
0Bh	Выбор палитры и цвета окаймления
0Ch	Запись пиксела
0Dh	Считывание пиксела
0Eh	Запись символа в телетайпном режиме
0Fh	Получение данных о состоянии видеосистемы
10h	Управление регистрами палитры, регистрами RAMDAC и атрибутом яркости/мерцания
11h	Управление знакогенератором
12h	Конфигурирование видеосистемы (выбор альтернативного видеоадаптера, если таковой установлен)
13h	Запись символьной цепочки
1Ah	Получение или задание комбинации видеосистемы
1Bh	Получение информации о состоянии видеосистемы
1Ch	Сохранение/восстановление состояния видеосистемы

Как и сами прерывания, функции нумеруются шестнадцатеричными числами.

В качестве наиболее простого и наглядного примера использования видеосервиса BIOS рассмотрим функцию 00h прерывания 10h — установка выбранного видеорежима. Номер видеорежима выбирается в соответствии с табл. 1.4. Например, основной текстовый режим видеосистемы VGA (80×25 символов, 720×400/16) имеет номер 03h. Для его инициализации при помощи видеосервиса BIOS необходимо выполнить следующие действия:

- занести в младший байт регистра AX (AL, от англ. *Low* — нижний) номер видеорежима — в данном случае 03h;
- занести в старший байт регистра AX (AH, от англ. *High* — верхний) номер функции — в данном случае 00h;
- инициализировать прерывание 10h командой INT 10h.

Для выполнения указанных действий на языке АССЕМБЛЕР потребуются всего лишь три оператора (все числовые параметры указываются в шестнадцатеричной системе счисления):

```
mov al, 03 ;занесение в регистр AL номера видеорежима
mov ah, 00 ;занесение в регистр AH номера функции
int 10     ;инициализация прерывания 10h, реализующего
            функцию 00h
```

В процессе выполнения этой короткой программы будет активизирована одна из подпрограмм Video

BIOS, которая корректно выполнит модификацию содержимого соответствующих регистров видеоадаптера и служебных областей системной памяти PC, в результате чего видеосистема корректно перейдет в выбранный видеорежим.

Отметим, что за каждой функцией видеосервиса BIOS закреплены свои регистры общего назначения. Например, координаты курсора в текстовом режиме всегда заносятся в регистр DX: номер столбца — в младший байт DL, а номер строки—в старший DH.

Видеосервис BIOS обеспечивает простое и надежное управление видеосистемой PC независимо от ее конкретной аппаратной реализации. Однако он имеет один существенный недостаток — сравнительно низкую скорость выполнения функций. С целью повышения быстродействия видеосистемы используется альтернативный метод: непосредственное изменение состояния регистров видеоадаптера прикладной программой. Такой метод потенциально является более быстрым и эффективным, однако требует от программиста детального знания аппаратных особенностей используемого видеоадаптера (адресов портов ввода/ вывода, назначения и структуры данных в управляющих регистрах и т. п.).

Отметим, что управление видеоадаптером путем модификации содержимого его регистров — это работа на свой страх и риск, поскольку малейшая ошибка может привести к сбоям в работе видеосистемы или к «зависанию» компьютера, а в некоторых случаях — и к выходу монитора из строя. Наглядный пример таких «сырых» программ, некорректно обновляющих содержимое регистров видеоадаптера, — драйверы к первым моделям видеоадаптеров SVGA, особенно изделиям типа *Noname.* В лучшем случае такой видеоадаптер работал медленно и некорректно, выводя на экран «мусор», а в худшем — приводил к краху операционной системы. Поэтому главное отличие качественного видеоадаптера — качественное программное обеспечение (драйвер).

Рассмотрим более подробно функции прерывания INT 10h для VGA BIOS

int 10 - функция 00h - УСТАНОВКА ВИДЕОРЕЖИМА.
AH = 00h
AL= режим (графический режим, если есть графические возможности. См. табл. 1.4).

int 10 - функция 01h - УСТАНОВКА ХАРАКТЕРИСТИК КУРСОРА.
AH = 0lh
CH биты 0 - 4 = начальная строка для курсора, биты 5—б = атрибут мигания (00 = нормальный, 01 = невидимый, 10 = медленный, 11= быстрый)
CL биты 0 - 4 = конечная строка для курсора

int 10 - функция 02h - УСТАНОВКА ПОЗИЦИИ КУРСОРА.
AH = 02h
DH, DL = строка, столбец (0, 0 = верхний левый угол),
BH =- номер страницы (0 в графическом режиме)

int 10 - функция 03h - СЧИТАТЬ ПОЗИЦИЮ КУРСОРА.
AH - 03h
BH = номер страницы (0 в графическом режиме)
Возвращает:
DH, DL = строка, столбец
CH = начальная строка курсора
CL = конечная строка курсора

int 10 - функция 04h - СЧИТАТЬ ПОЗИЦИЮ СВЕТОВОГО ПЕРА.
AH = 04h
Возвращает:
AH = 0: световое перо не приведено в действие
AH = 1: параметры светового пера в регистрах:
DH, DL = строка, колонка текущей позиции
CH = линия растра (0 - 199) (EGA), старый графический режим
CX = (EGA) линия растра (0 - nnn), новый графический режим
BX = пиксельная колонка (0 - 319 или 0 - 639)

int 10 - функция 05h - ВЫБОР СТРАНИЦЫ ДИСПЛЕЯ.
AH = 05h
AL = 0 - 7: значение новой страницы для режимов 0 и 1
0 - 3: значение новой страницы для режимов 2 и 3
0 - для режимов 4 - 7.

int 10 - функция 06h - СДВИГ СТРАНИЦЫ ВВЕРХ.

AH = 06h
AL = число строк, на которое сдвигается окно (0 = очищается все окно)
BH = атрибуты очищаемой строки
CH, CL = строка, столбец верхнего левого угла сдвигаемого окна
DH, DL = строка, столбец нижнего правого угла сдвигаемого окна

int 10 - функция 07h - СДВИГ СТРАНИЦЫ ВНИЗ.
AH = 07h
AL = число строк, на которое сдвигается окно (0 = очищается все окно)
BH = атрибуты, которые будут использоваться для очистки строк
CH, CL = строка, столбец верхнего левого угла сдвигаемого окна
DH, DL = строка, столбец нижнего правого угла сдвигаемого окна.

int 10 - функция 08h - ЧТЕНИЕ АТРИБУТА/СИМВОЛА С ПОЗИЦИИ КУРСОРА.
AH = 08h
BH = страница дисплея
Возвращает:
AL = символ
AH = атрибут символа (текстовый режим).

int 10 - функция 09h - ВЫВОД АТРИБУТА/СИМВОЛА, НАЧИНАЯ С ПОЗИЦИИ КУРСОРА.
AH = 09h
AL = символ
BH = страница дисплея
BL = атрибут символа (текстовой режим) или цвет (графические режимы). Если в графическом режиме бит 7 = 1, то образ символа складывается по модулю 2 с образом, расположенным в текущей позиции экрана
CX = число повторений вывода
Замечание. Отображаются все символы, включая CR, LF и BS.

int 10 - функция 0Ah - ВЫВОД ТОЛЬКО СИМВОЛОВ, НАЧИНАЯ С ТЕКУЩЕЙ ПОЗИЦИИ КУРСОРА.
AH = 0Ah
AL = символ
BH = страница дисплея — текстовой режим
BL = цвет символа (графический режим, PCjr только). Если в графическом режиме бит 7 = 1, то образ символа складывается по модулю 2 с образом, расположенным в текущей позиции экрана.
CX = число повторений вывода. Для EGA в графическом режиме повторная запись символа будет правильна в случае, если все символы помещаются на одну строку
Замечание. Отображаются все символы, включая CR, LF и BS.

int 10 - функция 0Bh - УСТАНОВКА ПАЛИТРЫ ДИСПЛЕЯ.
AH = 0Bh
BH = 0
BL = цвет бордюра (0 - 15) (текстовые режимы) цвет бордюра и цвет фона (графический режим) (EGA) (см. описание видеоадаптеров)
BL = цвет бордюра (0 - 15) и высокоинтенсивный цвет фона
BH = 1
BL = палитра (0 - 3)

int 10 - функция 0Ch - ЗАПИСЬ ТОЧКИ НА ЭКРАН.
AH = 0Ch
AL = цвет точки (0/1 в режиме 6, 0 - 3 в режиме 4 и 5), если бит 7=1, то значение нового цвета складывается по модулю 2 со значением цвета текущего пикселя
BH = страница дисплея (игнорируется, если режим поддерживает только одну страницу)
CX = колонка
DX = строка
Замечание. Возможно только в графическом режиме.

int 10 - функция 0Dh - ЧТЕНИЕ ТОЧКИ ЭКРАНА.
AH = 0Dh
BH = страница дисплея (игнорируется, если режим поддерживает только одну страницу)

CX = колонка
DX = строка
Возвращает:
AL = считанный цвет.
Замечание. Возможно только в графическом режиме.

int 10 - функция 0Eh - ЗАПИСЬ СИМВОЛА И ПРОДВИЖЕНИЕ КУРСОРА (ТЕЛЕТАЙПНАЯ ЗАПИСЬ).
AH = 0Eh
AL = символ
BH = страница дисплея (текстовой режим)
BL = цвет изображения (графический режим)

int 10 - функция 0Fh - ПОЛУЧЕНИЕ ТЕКУЩЕГО ВИДЕОРЕЖИМА.
AH = 0Fh
Возвращает:
AH = число символов в строке экрана
AL = текущий видеорежим (см. int l0h/AH = 00h)
BH = текущая активная страница дисплея.
Замечание. Если режим был установлен с единицей в бите 7 («не мигающий»), то возвращаемый режим будет также иметь установленный бит 7.

int 10 - функция 10h, подфункция00h - УСТАНОВКА РЕГИСТРА ПАЛИТРЫ (Jr, PS, EGA, VGA).
AX = l000h
BL = устанавливаемый регистр палитры (0 -15)
BH = сохраняемое значение цвета.
Замечание. При MCGA поддерживается только BX - 0712h.

int 10 - функция 10h, подфункция01h - УСТАНОВКА РЕГИСТРОВ ЦВЕТА БОРДЮРА (Jr, PS, EGA, VGA).
AX=1001h
BH = сохраняемое значение цвета

int 10 - функция 10h, подфункция02h - УСТАНОВКА ВСЕХ РЕГИСТРОВ ПАЛИТРЫ (Jr, PS, EGA, VGA).
AX = 1002h
ES:DX → 17-байтный список:
байты 0 - 15 = значения для регистров палитры 0 - 15,
байт 16 = значения для регистра цвета бордюра.

int 10 - функция 10h, подфункция03h - ПЕРЕКЛЮЧЕНИЕ БИТА «ПОВЫШЕННАЯ ЯРКОСТЬ/ МИГАНИЕ» (Jr, PS, EGA, VGA).
AX = 1003h
BL = 00h включена повышенная яркость
BL = 0lh включено мигание

int 10 - функция 10h, подфункция07h - ПОЛУЧЕНИЕ ЗНАЧЕНИЯ ОДНОГО РЕГИСТРА ПАЛИТРЫ (VGA).
AX = 1007h
BL = номер регистра палитры
Возвращает:
BH = значение регистра палитры

int 10 - функция 10h, подфункция08h - ЧТЕНИЕ РЕГИСТРА ЦВЕТА БОРДЮРА (VGA).
AX =1008h
Возвращает:
BH = значение

int 10 - функция 10h, подфункция09h - ЧТЕНИЕ ВСЕХ РЕГИСТРОВ ПАЛИТРЫ И РЕГИСТРА ЦВЕТА БОРДЮРА (VGA).
AX-1009h
ES:DX → 17-байтный буфер (см. AX = 1002h)

int 10 - функция 10h, подфункция10h - УСТАНОВКА ОДНОГО DAC-РЕГИСТРА (EGA, VGA/MCGA).

AX = 1010h
BX = номер регистра
CH = новое значение для зеленого цвета (0—63)
CL = новое значение для голубого цвета (0—63)
DH = новое значение для красного цвета (0—63)

int 10 - функция 10h, подфункция12h - УСТАНОВКА БЛОКА DAC-РЕГИСТРОВ (VGA/MCGA).

AX = 1012h
BX = начальный регистр цвета
CX = число устанавливаемых регистров
ES:DX → таблица из 3 × CX байт, где каждая группа из 3-х байтов содержит по байту для каждого цвета - красного, зеленого, голубого (0 - 63)

int 10 - функция 10h, подфункция13h - ВЫБОР ПАЛИТРЫ ВИДЕО-DAC (VGA).

AX=1013h
BL = 00h выбор палитры
BH = 00h выбор 4 цветов из 64
BH = 0lh выбор 16 цветов из 16
BL = 0lh выбор страницы
BH = номер страницы (00h - 03h) или (00h - 0Fh).

Замечание. **Недействительно в видеорежиме 13h.**

int 10 - функция 10h, подфункция15h - ЧТЕНИЕ ОДНОГО DAC-РЕГИСТРА (VGA/MCGA).

AX = 1015h
BL = номер регистра палитры

Возвращает:

DH = значение для красного
CH = значение для зеленого
CL = значение для голубого

int 10 - функция 10h, подфункция17h - СЧИТАТЬ БЛОК DAC-РЕГИСТРОВ (VGA/MCGA).

AX = 1017h
BX = начальный регистр цвета
CX = число считываемых регистров
ES:DX -> буфер (3×CX байт)

Возвращает:

CX число троек из красного, зеленого, голубого в буфере

int 10 - функция 10h, подфункция18h - недокументируемое — УСТАНОВКА PEL МАСКИ (VGA/MCGA).

AX = 1018h
BL = новое значение PEL

int 10 - функция 10h, подфункция19h - недокументируемое — ЧТЕНИЕ PEL МАСКИ (VGA/MCGA).

AX = 1019h

Возвращает:

BL = считанное значение.

int 10 - функция 10h, подфункция1Ah — ПОЛУЧЕНИЕ ПАЛИТРЫ ВИДЕО-DAC (VGA).

AX=101Ah

Возвращает:

BL= палитра
0 - четыре из 64,
1 - шестнадцать из 16
BH = текущая страница.

int 10 - функция 10h, подфункция1Bh — ВЫПОЛНЕНИЕ СУММИРОВАНИЯ ШКАЛЫ ЯРКОСТИ (EGA, VGA/MCGA).

AX=101Bh
BX = начальный регистр палитры
CX = количество преобразуемых регистров.

int 10 - функция 11h, режимы 00h, 10h, 01h, 11h, 02h, 12h, 03h, 04h, 14h — ФУНКЦИИ ГЕНЕРАТОРА СИМВОЛОВ ТЕКСТОВОГО РЕЖИМА (PS, EGA, VGA).

AH = 11h
AL=00h, 10h: загрузка определенного пользователем шаблона в блок символов
ES:BP → таблица пользователя
CX = число символов
DX = начальное смещение (номер символа) в таблице
BL = номер блока (0—3) карты битов второго видеобуфера
BH = число байтов в шаблоне символа
AL = 01h, 11h: набор для символов (8×14) копируется из ПЗУ
BL = номер блока
AL = 02h, 12h: набор для символов (8×8) копируется из ПЗУ
BL = номер блока
AL = 03h: установка определенного блока
BL = определитель блока (EGA/MCGA)
биты 0, 1 = номер блока символов, который берется, когда бит 3 байта атрибутов равен 0
биты 2, 3 = номер блока символов, который берется, когда бит 3 байта атрибутов равен 1 (VGA)
биты 0, 1,4 = номер блока символов, который берется, когда бит 3 байта атрибутов равен 0
биты 2, 3, 5 = номер блока символов, который берется, когда бит 3 байта атрибутов равен 1
AL = 04h, 14h: набор для символов (8×16) копируется из ПЗУ (VGA).

Замечание. Программы, вызываемые с AL = 1xh, разработаны под вызов непосредственно после установки режима и подобны программам, вызываемым с AL = 0xh, но имеются следующие отличия:

a. Страница 0 должна быть активной.
b. Число байтов на символ пересчитано.
c. Максимальное число строк символов пересчитано.
d. Длина CRT буфера пересчитана.
e. CRTC регистры перепрограммированы следующим образом:
R09 = bytes/char-1; максимальная линия развертки (только режим 7)
R0A = bytes/char-2; начальная позиция курсора
R0B = 0 ; конечная позиция курсора
R12 = ((rows+1)×(bytes/char)) - 1;
R14 = bytes/char.

int 10 - функция 11h, режимы (20h - 24h) - ФУНКЦИИ ГЕНЕРАТОРА СИМВОЛОВ ГРАФИЧЕСКОГО РЕЖИМА (PS, EGA, VGA).

AH = 11h
AL = 20h: установить графические символы пользователя 8×8 (int IFh)
ES:BP -> таблица пользователя
AL = 21h: установить графические символы пользователя
ES:BP -> таблица пользователя
CX = число байтов на символ
BL = количество строк
0: указывает пользователь
DL = число строк
1: 14 строк
2: 25 строк
3: 43 строки
AL= 22h: установить набор символов 8×14 из ПЗУ
BL = количество строк
AL = 23h: установить набор символов 8×8 из ПЗУ
BL = количество строк
AL = 24h: загрузить графические символы 8×16 (VGA/MCGA)
BL = количество строк.

Замечание. Эти функции могут быть вызваны только после установки режима.

int 10 - функция 11h, подфункция30h — ПОЛУЧЕНИЕ ИНФОРМАЦИИ О ЗНАКОГЕНЕРАТОРЕ (EGA, MCGA, VGA).

AX = 1130h
BH = определитель указателя
0: текущий указатель int 1Fh

1: текущий указатель int 44h (int 43h)
2: указатель на ПЗУ 8×14
3: указатель на ПЗУ 8×8 (двойной)
4: ПЗУ 8×8 DD (верхняя половина)
5: указатель на ПЗУ 9×14 (альтернативный)
6: указатель на знакогенератор 8×16 (только VGA)
7: указатель на знакогенератор 9×16 (только для VGA)

Возвращает:
ES:BP = указатель
CX = число байтов на символ
DL = число строк

int 10 - функция 12h, подфункция 10h - ПОЛУЧЕНИЕ ИНФОРМАЦИИ О EGA (PS, EGA, VGA, MCGA).

AH = 12h
BL = 10h

Возвращает:
BH = 0: в действии цветной режим (3Dx),
1: в действии монохромный режим (3Bx)
BL = 0: установлено 64 Кбайт памяти
1: установлено 128 Кбайт памяти
2: установлено 192 Кбайт памяти
3: установлено 256 Кбайт памяти
CH = биты признака
CL = настройка переключателя

int 10 - функция 12h, подфункция 20h - АЛЬТЕРНАТИВНАЯ ПРОГРАММА ПЕЧАТИ ЭКРАНА (PS, EGA, VGA, MCGA).

AH = 12h
BL = 20h выбор альтернативной программы печати экрана

int 10 - функция 12h, подфункция 30h - ВЫБОР ВЕРТИКАЛЬНОГО РАЗРЕШЕНИЯ (VGA).

AH = 12h
BL = 30h
AL = вертикальное разрешение
00h: 200 строк развертки
01h: 350 строк развертки
02h: 400 строк развертки

Возвращает:
AL = 12h, если данная функция поддерживается

int 10 - функция 12h, подфункция 31h - ЗАГРУЗКА ПАЛИТРЫ (VGA, MCGA).

AH = 12h
BL = 31h
AL = 00h - разрешение загрузки палитры, загружаемой по умолчанию
0lh - запрещение загрузки палитры, загружаемой по умолчанию

Возвращает:
AL = 12h, если данная функция поддерживается

int 10 - функция 12h, подфункция 32h - ВИДЕОАДРЕСАЦИЯ (VGA, MCGA).

AH = 12h
BL = 32h
AL = 00h - разрешить видеоадресацию
01h - запретить видеоадресацию

Возвращает:
AL = 12h, если данная функция поддерживается.

int 10 - функция 12h, подфункция 33h - СУММИРОВАНИЕ ШКАЛЫ ЯРКОСТИ (VGA, MCGA).

AH = 12h
BL=33h
AL = 00h - разрешение суммирования шкалы яркости,
01h - запрещение суммирования шкалы яркости

Возвращает:
AL = 12h, если данная функция поддерживается.

int 10 - функция 12h, подфункция 34h — ЭМУЛЯЦИЯ КУРСОРА (VGA).
AH = 12h
BL = 34h
AL = 00h разрешение алфавитно-цифровой эмуляции курсора,
0lh запрещение алфавитно-цифровой эмуляции курсора
Возвращает:
AL = 12h, если данная функция поддерживается.

int 10 - функция 12h, подфункция 36h - УПРАВЛЕНИЕ ВИДЕОРЕГЕНЕРАЦИЕЙ (PS, VGA).
AH = 12h
BL = 36h
AL = 00h разрешить регенерацию,
0lh запретить регенерацию
Возвращает:
AL = 12h, если данная функция поддерживается.

int 10 - функция 13h - ПЕЧАТЬ СТРОКИ (AT, XT286, PS, EGA, VGA).
AH=13h
AL = режим
бит 0: курсор устанавливается
бит 1: бит устанавливается, если в строке альтернативные символы и атрибуты
BL = атрибут, если в AL бит 1 очищен
BH = номер страницы дисплея
DH, DL = строка, столбец начальной позиции курсора
CX = длина строки
ES:BP → начало строки.
Замечание. Различает символы CR, LF, BS и bell.

int 10 - функция 15h — ЧТЕНИЕ ХАРАКТЕРИСТИК ДИСПЛЕЯ.
AH = 15h
Возвращает:
AX = тип адаптера дисплея:
0000h никакой
5140h LCD
5151h монохромный
5153h CGA
ES:DI -> таблица параметров
Слово 0: номер модели дисплея
1: точек на метр по вертикали
2: точек на метр по горизонтали
3: число точек по вертикали
4: число точек по горизонтали
5: высота точки в микрометрах
6: ширина точки в микрометрах

int 10 - функция 1Ah — КОМБИНАЦИЯ ДИСПЛЕЕВ (PS, VGA/MCGA).
AH - lAh
AL = 00h считать код комбинации дисплеев
Возвращает: BL = код активного дисплея
BH = код альтернативного дисплея
01h установить код комбинации дисплеев
BL = код активного дисплея
BH = код альтернативного дисплея
Возвращает:
AL = lAh, если данная функция поддерживается

Коды комбинации дисплеев:
00h - нет дисплея
01h - монохромный адаптер с монохромным дисплеем
02h - CGA с цветным дисплеем

03h - зарезервировано
04h - EGA с цветным дисплеем
05h - EGA с монохромным дисплеем
06h - PGA с цветным дисплеем
07h - VGA с монохромным аналоговым дисплеем
08h - VGA с цветным аналоговым дисплеем
09h - зарезервировано
0Ah - MCGA с цветным цифровым дисплеем
0Bh - MCGA с монохромным аналоговым дисплеем
0Ch - MCGA с цветным аналоговым дисплеем
FFh - неизвестный тип дисплея

int 10 - функция 1Bh — ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ/ИНФОРМАЦИЯ СОСТОЯНИЯ (PS, VGA/MCGA).

AH = 1Bh
BX - тип реализации (BX = 0000h возвращает функциональные возможности/информация состояния)
ES:DI - адрес 64 байтного буфера

Возвращает:

AL = 1Bh, если данная функция поддерживается
ES:DI → буфер, заполненный информацией состояния

Формат информации состояния:

Смещение	Размер	Описание
00h	DWORD	Адрес таблицы неизменяемых функциональных возможностей
04h	BYTE	Активный видеорежим
05h	WORD	Число столбцов
07h	WORD	Длина regen буфера в байтах
09h	WORD	Начальный адрес regen буфера
0Bh	WORD	Позиция курсора на странице 0
0Dh	WORD	Позиция курсора на странице 1
0Fh	WORD	Позиция курсора на странице 2
11h	WORD	Позиция курсора на странице 3
13h	WORD	Позиция курсора на странице 4
15h	WORD	Позиция курсора на странице 5
17h	WORD	Позиция курсора на странице 6
19h	WORD	Позиция курсора на странице 7
1Bh	WORD	Тип курсора
1Dh	BYTE	Активная страница дисплея
1Eh	WORD	Адрес порта видеоконтроллера
20h	BYTE	Текущая установка регистра (3?8)
21h	BYTE	Текущая установка регистра (3?9)
22h	BYTE	Число строк
23h	WORD	Байтов на символ
25h	BYTE	DCC активного дисплея
26h	BYTE	DCC альтернативного дисплея
27h	WORD	Число цветов, поддерживаемых в текущем режиме
29h	WORD	Число страниц, поддерживаемых в текущем режиме
2Ah	BYTE	Число строк развертки (0, 1,2,3) = (200, 350, 400, 480)
2Bh	BYTE	Исходный блок символов
2Ch	BYTE	Вторичный блок символов

2Dh	BYTE	Многоцелевые флаги: бит 0 — все режимы для всех дисплеев разрешены 1 — включено суммирование оттенков серого 2 — монохромный дисплей подключен 3 — загрузка палитры, принимаемой по умолчанию, запрещена 4 — эмуляция курсора разрешена 5 0 - повышенная интенсивность, 1 - мигание 6 — зарезервировано 7 — зарезервировано
2Eh	3 BYTEs	Зарезервировано
31h	BYTE	Доступная видеопамять: 00h - 64К, 0lh - 128К, 02h - 192К, 03h - 256К
32h	BYTE	Флаги состояния: бит 0 — установка 512 символов активна 1 — есть динамически сохраняемая область 2 — переопределение символьных фонтов активно 3 — переопределение графических фонтов активно 4 — переопределение палитры активно 5 — переопределение DCC активно 6 — зарезервировано 7 — зарезервировано
33h	13 BYTEs	Зарезервировано

int 10 - функция 1Ch — СОХРАНЕНИЕ/ВОСТАНОВЛЕНИЕ ВИДЕОСОСТОЯНИЯ (PS50+, VGA).

AH = 1Ch

AL = 0: возврат размера буфера состояния

Возвращает:

BX = число необходимых 64-байтных блоков

1: сохранить видеосостояние

ES:BX → буфер для сохранения

2: восстановить видеосостояние

ES:BX → буфер, содержащий сохраненное ранее состояние

CX = требуемые состояния

бит 0 аппаратные средства видеоадаптера

1 области данных BIOS

2 регистры цвета и DAC-состояние

3 - 15 зарезервировано

Возвращает: AL = 1Ch, если данная функция поддерживается

2.1.3. Контроллер ЭЛТ

Контроллер ЭЛТ формирует сигналы горизонтальной и вертикальной синхронизации, сигналы инкремента (увеличения) счетчика адреса ячеек видеопамяти, в которых хранится цифровой образ изображения, а также стробирующие сигналы чтения/записи видеопамяти. Эти сигналы формируются таким образом, что движение луча по экрану ЭЛТ осуществляется синхронно с процессом сканирования ячеек видеопамяти, причем цвет пиксела на экране соответствует коду, содержащемуся в соответствующей ячейке кадрового буфера.

Помимо выполнения указанной основной функции, контроллер ЭЛТ решает ряд дополнительных задач:

- выбор формы курсора в текстовом режиме;
- выбор выводимой на экран активной видеостраницы;
- установка параметров аппаратного подчеркивания символов.

Работа контроллера ЭЛТ синхронизируется сигналами одного из двух тактовых генераторов *(Dot Clock* или *Pixel Clock),* установленных на плате видеоадаптера. Их частоты равны 28,322 и 25,175 МГц соответственно. Выбор синхросигнала производится программно.

2.1.3.1. Регистры контроллера ЭЛТ

По сравнению с другими элементами видеоадаптера, контроллер ЭЛТ имеет наибольшее количество (26) программно управляемых регистров. Первым образцом контроллера ЭЛТ для PC была микросхема 6845

фирмы Motorola, использовавшаяся в видеоадаптере MDA. Она содержала 16 8-разрядных регистров, большинство из которых было доступно центральному процессору только для записи, что затрудняло контроль за работой видеосистемы. В последующих моделях видеоадаптеров (в частности, в видеоадаптерах VGA) количество регистров было увеличено, большинство из них стали доступны как для записи, так и для чтения, однако основные регистры, реализованные в 6845, были сохранены в целях совместимости с программным обеспечением.

Доступ центрального процессора к регистрам контроллера ЭЛТ осуществляется через два порта ввода/вывода. При работе видеосистемы в цветном режиме они имеют адреса 3D4h и 3D5h, а в монохромном - 3B4h и 3B5h (для обеспечения совместимости с программами, написанными для видеосистемы MDA).

По адресу 3D4h расположен так называемый *индексный (адресный) регистр,* предназначенный для управления обменом данными со всеми остальными регистрами контроллера ЭЛТ. При необходимости обращения к любому регистру центральный процессор предварительно должен занести его номер, или *индекс* (табл. 2.3), в индексный регистр (по адресу 3D4h). В дальнейшем, при обращении CPU к порту ввода/вывода по адресу 3D5h, порт оказывается соединенным с тем регистром, номер (индекс) которого находится в индексном регистре. При этом говорят, что индексный регистр *отображается* в порт 3D4h, а все остальные регистры контроллера ЭЛТ — в порт 3D5h.

Для задания координат текущего символа или пиксела на экране контроллер ЭЛТ имеет два счетчика: счетчик символов, определяющий горизонтальную координату пиксела и содержащий номер текущей символьной позиции, и счетчик строк, определяющий вертикальную координату и содержащий номер текущей строки растра. В моменты, когда значения этих счетчиков совпадают с числами, записанными в определенных регистрах контроллера ЭЛТ (см. табл. 2.3), начинают формироваться сигналы горизонтальной и вертикальной синхронизации, а также стробирующие сигналы чтения/записи видеопамяти.

Таблица 2.3. Регистры контроллера ЭЛТ

Порт	Номер (индекс)	Наименование	Назначение
3D4h/ 3B4h	—	Индексный регистр *(CRTC Address Register)*	Содержит индекс (номер) регистра контроллера ЭЛТ, к которому будет выполняться обращение
3D5h/ 3B5h	00h	Регистр общего числа символьных позиций по горизонтали *(Horizontal Total Register)*	Содержит общее количество символов в строке сканирования, включая обратный ход по строке (обычно 5 позиций приходится на обратный ход пуча)
3D5h/ 3B5h	01h	Регистр окончания вывода по горизонтали *(Horizontal Display Enable End Register)*	Содержит номер последней видимой символьной позиции в строке, т. е. уменьшенное на 1 количество символов в строке
3D5h/ 3B5h	02h	Регистр начала гашения по горизонтали *(Start Horizontal Blanking Register)*	Содержит номер символьной позиции, начиная с которой действует строчный гасящий импульс
3D5h/ 3B5h	03h	Регистр окончания гашения по горизонтали *(End Horizontal Blanking Register)*	Содержит пять младших разрядов номера символьной позиции, в которой заканчивается строчный гасящий импульс (биты 0-4), а также количество тактов задержки (в символах) между обнулением счетчика символов и началом вывода на экран (биты 5 и 6): [6] [5] Смещение в символах 0 0 0 0 1 1 1 0 2 1 1 3
3D5h/ 3B5h	04h	Регистр начала обратного хода по горизонтали *(Start Horizontal Retrace Register)*	Содержит номер символьной позиции, с которой начинается обратный ход строчной развертки

Таблица 2.3. Регистры контроллера ЭЛТ (продолжение).

Порт	Номер (индекс)	Наименование	Назначение
3D5h/ 3B5h	05h	Регистр окончания обратного хода по горизонтали *(End Horizontal Retrace Register)*	Содержит номер символьной позиции, в которой заканчивается обратный ход строчной развертки (биты 0 - 4), значение задержки между окончанием импульса H-Sync и началом вывода на экран (биты 5 и 6), а также шестой разряд номера символьной позиции, в которой заканчивается строчный гасящий импульс (бит 7): [6] [5] Сдвиг строчного СИ в символах 0 0 0 0 1 1 1 0 2 1 1 3 [7] - разряд [5] регистра конца гашения по строке 3D5.03
3D5h/ 3B5h	06h	Регистр общего числа строк растра по вертикали *(Vertical Total Register)*	Содержит 8 младших разрядов числа, задающего общее количество строк растра, уменьшенное на 1, включаяя обратный ход луча по кадру. Биты 9 и 10 хранятся в регистре переполнения 07h
3D5h/ 3B5h	07h	Регистр переполнения *(CRT Controller Overflow Register)*	Содержит старшие разряды тех чисел, которые целиком не помещаются в 8-разрядные регистры контроллера ЭЛТ: [0] - разряд 8 регистра общего количества линий сканирования в кадре 3D5.06; [1] - разряд 8 регистра конца отображения по кадру 305.12; [2] - разряд 8 регистра начала кадрового синхроимпульса 3D5.10; [3] - разряд 8 регистра начала гашения по кадру 3D5.15; [4] - разряд 8 регистра сравнения линий 3D5.18; [5] - разряд 9 регистра общего количества линий сканирования в кадре 3D5.06; [6] - разряд 9 регистра конца отображения по кадру 3D5.12; [7] - разряд 9 регистра начала кадрового синхроимпульса 3D5.10
3D5h/ 3B5h	08h	Регистр заданной строки *(Preset Row Scan Register)*	Содержит число, соответствующее адресу строки, которая будет выведена на экран первой (используется для выполнения прокрутки экрана по вертикали в текстовых режимах): [0-4] - регистр предварительной установки линии сканирования в строке; [5, 6] - управление побайтовым панорамированием: [6] [5] Групповой сдвиг на 0 0 1 байт 0 1 2 байта 1 0 3 байта 1 1 4 байта

Таблица 2.3. Регистры контроллера ЭЛТ (продолжение).

Порт	Номер (индекс)	Наименование	Назначение
3D5h/ 3B5h	09h	Регистр высоты символьной позиции *(Maximum Scan Line Register)*	Содержит число, равное уменьшенному на 1 количеству строк растра, необходимому для прорисовки символа (т. е. уменьшенную на 1 высоту символьной позиции в пикселах): [0-4] - регистр максимальной линии сканирования в строке; [5] - разряд 9 регистра начала гашения по кадру 3D5.15; [6] - разряд 9 регистра сравнения линий 3D5 . 18; [7] - дублирование строк: 1- счетчик линий сканирования в строке синхронизируется с частотой, вдвое меньшей обычной, благодаря чему каждая линия сканирования отображается дважды. Используется для отображения 200-строковых режимов как режимов с разрешением 400 строк, 0- счетчик линий сканирования в строке синхронизируется с обычной частотой
3D5h/ 3B5h	0Ah	Регистр нижнего края курсора *(Cursor Start Register)*	Задает номер строки растра в пределах символьной матрицы, начиная с которой формируется изображение курсора: [0-4] - определяют линию сканирования в строке, где начинается отображение курсора; [5] - управление курсором: 1 - курсор отключен, 0 - курсор включен.
3D5h/ 3B5h	0Bh	Регистр верхнего края курсора *(Cursor End Register)*	Задает номер строки растра в пределах символьной матрицы, в которой завершается формирование изображения курсора: [0-4] - определяют линию сканирования в строке, где кончается отображение курсора; [5, 6] - управление сдвигом курсора: [6] [5] Сдвиг курсора в символах 0 0 0 0 1 1 1 0 2 1 1 3
3D5h/ 3B5h	0Ch	Регистр старшего байта начального адреса *(Start Address High Register)*	Содержит старший байт адреса видеопамяти, начиная с которого размещается отображаемая страница видеопамяти (используется в режиме прокрутки изображения)
3D5h/ 3B5h	0Dh	Регистр младшего байта начального адреса *(Start Address Low Register)*	Содержит младший байт адреса видеопамяти, начиная с которого размещается отображаемая страница видеопамяти (используется при переключении страниц видеопамяти)
3D5h/ 3B5h	0Eh	Регистр старшего байта адреса курсора *(Cursor Location High Register)*	Вместе с содержимым регистра 3D5.0F определяет позицию курсора в пределах 64 Кбайт буфера экрана (максимальный размер одной плоскости). Используется для задания или считывания позиции курсора.
3D5h/ 3B5h	0Fh	Регистр младшего байта адреса курсора *(Cursor Location Low Register)*	Содержит младший байт адреса ячейки видеопамяти, содержащей код символа (пиксела), соответствующего текущему положению курсора (используется для задания или считывания позиции курсора)

Таблица 2.3. Регистры контроллера ЭЛТ (продолжение).

Порт	Номер (индекс)	Наименование	Назначение
3D5h/ 3B5h	10h	Регистр начала обратного хода по вертикали *(Vertical Retrace Start Register)*	При записи содержит 8 младших разрядов числа, определяющего номер строки растра, с которой начинается обратный ход кадровой развертки. При чтении - регистр светового пера (старший байт); в регистре содержится адрес видеобуфера, который защелкивается первым после возникновения строба светового пера синхроимпульсом символа
3D5h/ 3B5h	11h	Регистр окончания обратного хода по вертикали *(Vertical Retrace End Register)*	При записи содержит 4-разрядное число, определяющее номер строки растра, в которой заканчивается обратный ход кадровой развертки, а также биты 4 и 5, управляющие инициализацией прерывания, по которому производится обновление кадрового буфера во время обратного хода кадровой развертки: [0-3] - определяют момент сброса кадрового синхроимпульса. В регистре программируется величина, равная значению, содержащемуся в регистре 3D5.10, плюс длительность кадрового синхроимпульса (в линиях сканирования) ; [4] - сброс прерывания по обратному ходу кадра. Логический нуль сбрасывает прерывание; [5] - разрешение прерывания по обратному ходу кадра. Логический нуль разрешает выдачу прерывания; [6] - выбор режима регенерации памяти: 1 - выбор 5 циклов регенерации динамической памяти во время обратного хода по строке, 0 - выбор 3 циклов регенерации; [7] - блокировка записи: 1 - блокировка записи в регистры 3D5.00-3D5.07, управляющие синхронизацией видеомонитора , 0 - запись разрешена. При чтении - регистр светового пера (младший байт). Вместе с регистром 3D5.10 светового пера содержит 16-разрядный адрес видеобуфера.
3D5h/ 3B5h	12h	Регистр окончания вывода по вертикали *(Vertical Display Enable End Register)*	Десятиразрядный регистр (см. разряды 1, 6 регистра 3D5.07) определяет количество отображаемых линий сканирования в кадре.
3D5h/ 3B5h	13h	Регистр смещения *(Offset Register)*	Задает объем видеопамяти, необходимый для вывода одной строки растра (определяет логическую ширину видимой области экрана в словах). С помощью этого регистра определяется начальный адрес каждой следующей строки экрана во время отображения (прямого хода по кадру).
3D5h/ 3B5h	14h	Регистр положения символа подчеркивания *(Underline Location Register)*	В пределах символьной матрицы определяет положение строки, содержащей символ подчеркивания [0-4] - определяют линию сканирования в строке, где возможно подчеркивание символа; [5] - счет по четыре: 1 - счетчик адреса регенерации МА синхронизируется каждым четвертым синхроимпульсом символа, 0 - счетчик адреса регенерации МА синхронизируется каждым синхроимпульсом символа; [6] - режим двойного слова: 1- адресация двойных слов (см. 3D5.17 [6]), 0- побайтовая адресация.
3D5h/ 3B5h	15h	Регистр начала гашения по вертикали *(Start Vertical Blanking Register)*	Содержит 8 младших бит номера строки растра, с которой начинается кадровый гасящий импульс (9-й бит номера хранится в регистре переполнения 3D5.7 [3], 10-й — в регистре высоты символьной позиции 3D5.09 [5])

Таблица 2.3. Регистры контроллера ЭЛТ (продолжение).

Порт	Номер (индекс)	Наименование	Назначение
3D5h/ 3B5h	16h	Регистр окончания гашения по вертикали *(End Vertical Blanking Register)*	Содержит 8-разрядный номер строки растра, в которой заканчивается кадровый гасящий импульс
3D5h	17h	Регистр выбора режима *(Mode Control Register)*	Позволяет изменять режимы обращения к видеопамяти, режимы работы счетчика адреса ячеек видеопамяти и др. Регистр управления: [0] - режим совместимости со стандартом CGA. При наличии логического нуля в этом разряде значение разряда 0 счетчика линий в строке (RA0) заменяет значение разряда 13 адреса регенерации MA, что обеспечивает адресацию нечетных линий сканирования через 8 Кбайт в области буфера дисплея. При логической единице такая замена не осуществляется; [1] - режим совместимости со стандартом HGC. При наличии логического нуля в этом разряде значение разряда 1 счетчика линий в строке заменяет значение разряда 14 адреса регенерации MA, что обеспечивает адресацию каждой второй и третьей линий сканирования через 16 Кбайт в области буфера дисплея. При логической единице такая замена не осуществляется; [2] - удвоение разрешающей способности по вертикали. Этот разряд управляет разрешающей способностью CRT по вертикали. Десятиразрядный регистр 3D5.06 обеспечивает максимальное разрешение, равное 1024 линиям сканирования в кадре. Если счетчик линий в кадре синхронизируется с частотой, меньшей вдвое, то максимальное число линий сканирования в кадре удваивается (2048): [2] = 0 - синхронизация счетчика линий в кадре осуществляется по каждой линии сканирования, [2] = 1 - синхронизация счетчика линий в кадре осуществляется по каждой второй линии сканирования. Системный сброс RESET устанавливает данный разряд в нуль ; [3] - счет по два: [3]=0 - счетчик MA синхронизируется от синхроимпульса символа; [3]=1 - счетчик MA синхронизируется от синхроимпульса символа, деленного на 2; [5] - преобразование адреса. В режиме байта (3D5.17 [6] = 1) преобразование адреса не происходит. В режиме слова (3D5.17 [6]=0) адрес слова формируется следующим образом: если [5]=0, то MA0=MA13, если [5]=1, то MA0=MA15; [6] - режим слова или режим байта: [6]=0 - режим слова, [6]=1 - режим байта. В режиме слова выходы счетчика адреса регенерации MA сдвигаются влево на один разряд (MA1=MA0, MA2=MA1 и т.д.), а старший значащий разряд счетчика (MA13 или MA15) в зависимости от значения разряда 6 становится младшим значащим выходом адреса регенерации; [7] - аппаратурный сброс. Логический нуль сбрасывает VSYN и HSYN. Логическая единица разрешает формирование VSYNC и HSYNC. Системный сброс RESET устанавливает данный разряд в нуль
3D5h	18h	Регистр сравнения строк *(Line Compare Register)*	Используется при создании оконного графического интерфейса

Как следует из табл. 2.3, хранящиеся в регистрах контроллера ЭЛТ значения полностью определяют параметры растра на экране монитора: разрешение, частоту кадров, размеры знакоместа и др. Смена видеорежима реализуется путем записи в эти регистры новых значений, соответствующих выбранному режиму (эту задачу решает функция 00h прерывания 10h). В качестве примера рассмотрим, как определяются значения регистров, задающих параметры горизонтальной развертки, для видеорежима 03h (текстовый режим 80×25 символов, размер знакоместа 9×16, разрешение 720×400, 16 цветов, частота строк 31,469 кГц, частота кадров 70 Гц). Значения этих регистров приведены в табл. 2.4.

Таблица 2.4. Значения, содержащиеся в регистрах контроллера ЭЛТ при инициализации видеорежима 03h

Регистр	00h	01h	02h	03h	04h	05h
Содержимое регистра	5Fh(99)	4Fh(79)	50h (80)	82h (98)	55h (85)	81h (97)

Примечание. В скобках указаны десятичные номера символьных позиций.

В данном режиме используется тактовый генератор Dot Clock с частотой $f_{такт}$= 28,322 МГц, т. е. за одну секунду выводится 28 322 000 пикселов. Поскольку частота строк в этом режиме составляет $f_{стр}$ = 31,469 кГц, в течение одного периода строчной развертки видеоадаптер может сформировать. $f_{такт}/f_{стр}$ = 28322000/31469 = 900 пикселов, или 900/9 = 100 знакомест (первое знакоместо имеет номер 0, последнее — 99). Число 100, уменьшенное на 5 (95 или 5F в шестнадцатеричном коде, или 01011111 — в двоичном), заносится в регистр 00h. Из этих 95 позиций на экране отображаются только 80: с нулевой по 79-ю, поэтому в регистр 01h записывается число 79 (4Fh).

Поскольку импульс гашения по горизонтали начинается с 80-й позиции, в регистр 02h заносится число 80 (50h).

Номер символьной позиции, в которой заканчивается строчный гасящий импульс, задается двоичным шестиразрядным числом. Когда число в шести младших разрядах счетчика символов станет равным этому числу, спустя 2 символьных такта импульс гашения по горизонтали закончится. Поэтому в качестве момента окончания строчного гасящего импульса указывают 98-ю символьную позицию (1100010), а в регистр 03h заносят только пять младших бит этого числа, т. е. двоичное число 00010. Старший, 7-й бит этого регистра всегда установлен в единицу, а 5-й и 6-й — в нуль, поэтому с учетом сказанного выше в данном регистре будет записано двоичное число 10000010, или 82h. Шестой бит числа 1100010, равный 1, записывается в старший, 7-й разряд регистра 05h.

Импульс обратного хода (сигнал H-Sync) начинается с 85-й символьной позиции, поэтому в регистр 04h заносится число 55h.

Момент окончания сигнала H-Sync задается пятиразрядным двоичным числом. Когда число в пяти младших разрядах счетчика символов станет равным этому числу, импульс обратного хода по горизонтали закончится. Поскольку сигнал H-Sync заканчивается в 97-й символьной позиции (1100001), в разряды 0—4 регистра 05h записывается двоичное число 00001, т. е. 5 младших бит числа 1100001. Старший, 7-й разряд данного регистра содержит 1, поскольку в него записывается значение 6-го бита числа, задающего момент окончания гашения по горизонтали, а в 5-м и 6-м разрядах регистра 05h всегда записан нуль. Следовательно, в регистр 05h будет записан двоичный код 10000001, или число 81h.

Временная диаграмма, иллюстрирующая взаимное расположение растра, сигнала гашения по горизонтали и сигнала обратного хода, приведена на рис. 2.9.

Из рис. 2.5 видно, что длительность импульса обратного хода всегда меньше, чем длительность импульса гашения. Благодаря этому по краям растра образуется небольшая область, называемая *окаймлением (overscan)* или *бордюром (border).* Поскольку во время действия импульса гашения вывод данных из кадрового буфера не производится, окаймление по умолчанию будет черного цвета.

2.1.3.2. Программная настройка растра

Изменяя ширину окаймления путем модификации содержимого регистров контроллера ЭЛТ, можно регулировать положение и размеры растра на экране. Такая программная регулировка растра полезна в том случае, когда используется монитор устаревшей модели, не имеющий системы цифрового управления. При изменении разрешения или частоты кадров у такого монитора положение растра на экране и его размеры могут значительно изменяться, поэтому каждый раз после смены видеорежима потребуется ручная регулировка растра, что весьма утомительно. Программное управление растром, выполняемое путем изменения содержимого регистров контроллера ЭЛТ, дает возможность отрегулировать растр для каждого видеорежима и затем сохранить значения регистров в специальном файле настроек, который используется драйвером видеоадаптера. Благодаря этому обеспечивается стабильность размеров и положения растра на экране при смене видеорежима.

На практике подобная настройка растра выполняется при помощи специальных утилит, которые поставляются вместе с драйвером видеоадаптера: в среде DOS используются отдельные программы, в среде Windows 95/98 при установке драйвера видеоадаптера в диалоговом окне **Свойства: Экран** появляется

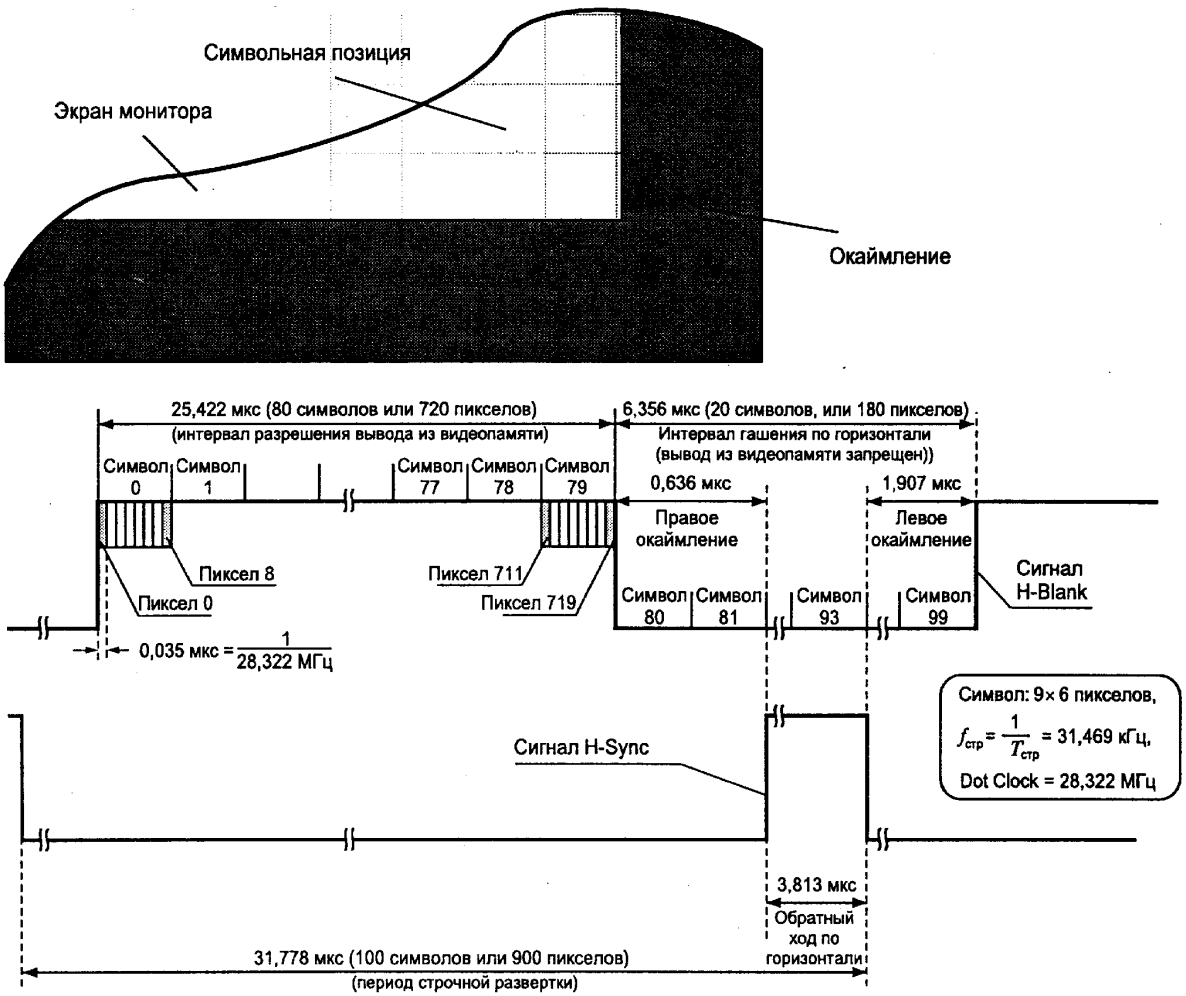


Рис. 2.9. Взаимное расположение растра, сигнала гашения по горизонтали и сигнала обратного хода

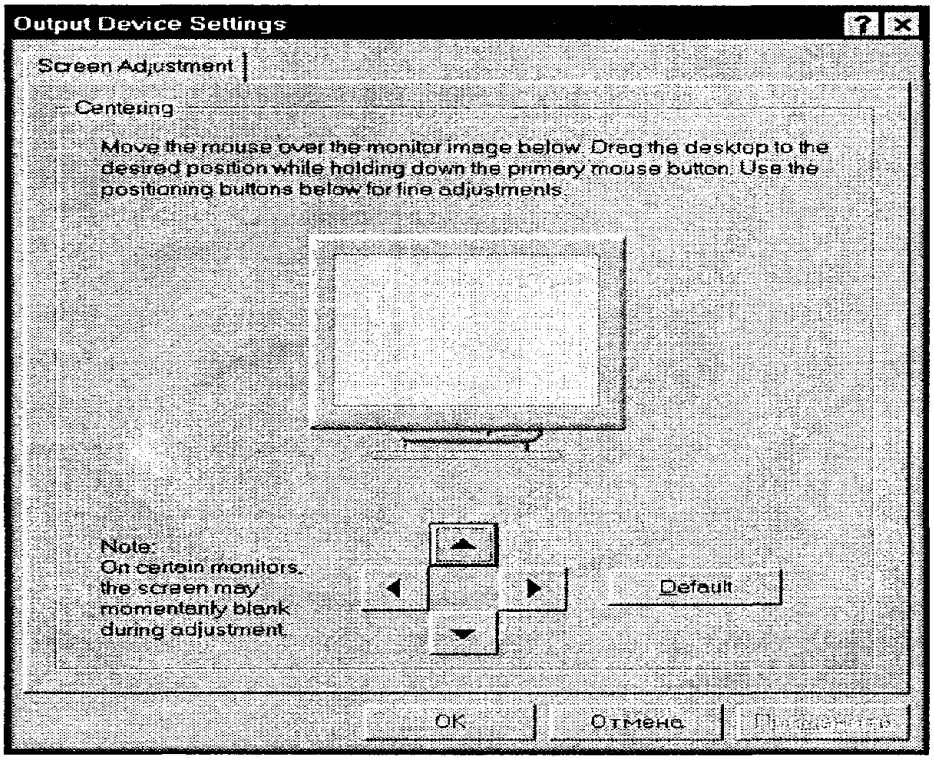


Рис. 2.10. Настройка растра средствами драйвера видеоадаптера Riva TNT

дополнительная закладка (рис. 2.10).

Если драйвер видеоадаптера лишен такой функции, программную настройку растра в среде Windows

95/98 можно выполнить при помощи программы PowerStrip (рис. 2.11). Заметим, что эта программа позволяет выполнить тонкую (с точностью до пиксела) настройку растра как по горизонтали, так и по вертикали.

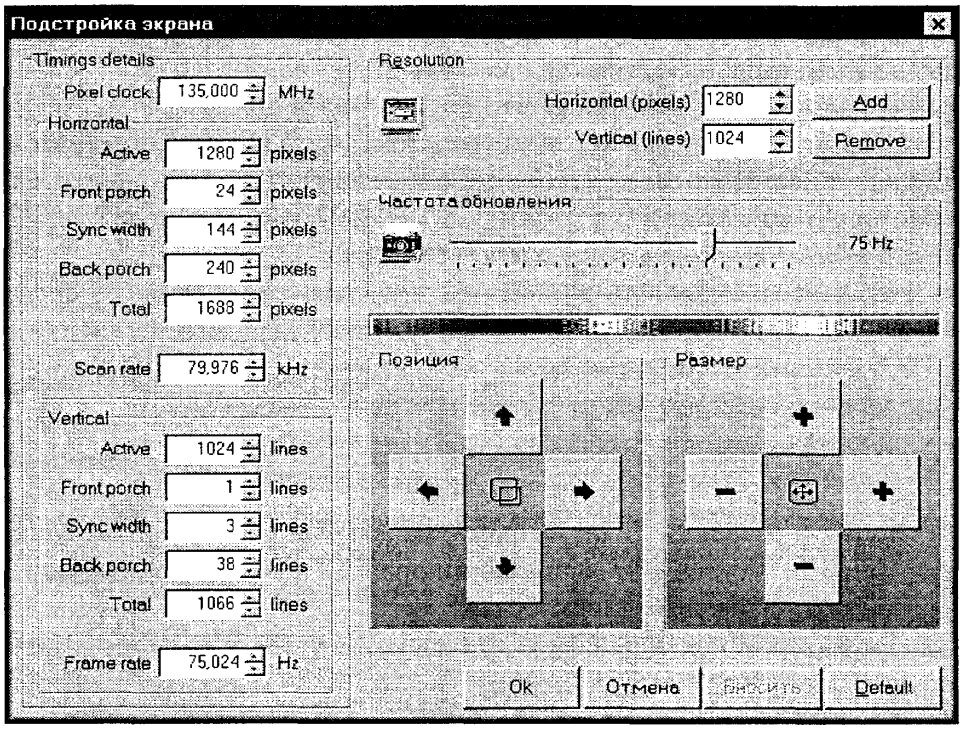


Рис. 2.11. Настройка растра средствами программы PowerStrip

2.1.4. Графический контроллер

Графический контроллер предназначен для управления обменом данными между центральным процессором и видеопамятью, а также для выполнения элементарных преобразований этих данных. Необходимость в наличии такого «интеллектуального посредника» между CPU и кадровым буфером обусловлена следующими причинами:

- сложностью адресации ячеек видеопамяти при разбиении последней на цветовые плоскости (напомним, что ячейки видеопамяти с одинаковыми координатами, но принадлежащие разным цветовым плоскостям имеют один и тот же адрес, поэтому необходимо устройство, производящее выбор этих плоскостей);
- стремлением обеспечить возможность оперативной модификации содержимого отдельных ячеек видеобуфера, т. е. стремлением ускорить обновление кадрового буфера.

Графический контроллер помогает центральному процессору выполнять следующие элементарные операции:

- запись пиксела (группы пикселов) по заданному адресу;
- считывание значения пиксела (группы пикселов) по заданному адресу;
- модификация цвета пиксела (группы пикселов) путем выполнения таких элементарных логических операций, как «И», «ИЛИ», «Исключающее ИЛИ», циклический сдвиг бит в пределах байта и др.;
- считывание из кадрового буфера кода пиксела (считывание со сравнением цвета; фактически реализуется поиск пиксела с заданным цветом) и др.

Выполнение этих операций осуществляется следующими элементами графического контроллера:

- десятью регистрами (табл. 2.5), один из которых является индексным, а остальные девять отображены в порт 3CFh;
- четырьмя 8-разрядными регистрами-защелками *(latches),* играющими роль буферных регистров при выполнении операций чтения/записи данных, содержащихся в битовых плоскостях;
- арифметико-логическим устройством (АЛУ).

2.1.4.1. Регистры графического контроллера

Доступ к регистрам графического контроллера выполняется точно так же, как и к регистрам контроллера ЭЛТ: вначале в индексный регистр по адресу 3CEh заносится номер выбранного регистра, а затем через порт 3CFh выполняется обращение к нему.

Таблица 2.5. Регистры графического контроллера

Порт	Номер (индекс)	Наименование	Назначение
3CEh		Индексный регистр *(Address Register)*	Содержит индекс (номер) регистра графического контроллера, к которому будет выполняться обращение
3CFh	00h	Регистр установки/сброса *(Set/Reset Register)*	Предназначен для управления режимами записи: Разряды 0-3 регистра записываются соответственно в плоскостях 0-3, если выбран режим записи 0 и разрешен режим установки (сброса) для соответствующей плоскости регистром разрешения установки (сброса) 3CF.01
3CFh	01h	Регистр разрешения установки/ сброса *(Enable Set/Reset Register)*	Предназначен для управления режимами записи: Этот регистр работает совместно с 3CF.00. Единичное состояние разрядов 0-3 разрешает действие регистра установки (сброса) на соответствующую плоскость. При нулевом значении любого разряда 3CF.01 соответствующая плоскость загружаются данными шины данных процессора в режиме записи 0
3CFh	02h	Регистр сравнения цвета *(Color Compare Register)*	Содержит код цвета, с которым сравнивается значение считываемого пиксела: Если регистр режима установлен в режим считывания 1, данные, считанные из плоскостей памяти 0-3, сравниваются с разрядами 0-3 регистра сравнения цвета. Результатом сравнения является логическая единица в соответствующем разряде шины данных процессора, например: [0] 1 1 1 1 1 1 1 1 1 - плоскость 0, [1] 1 0 1 1 0 0 0 0 1 - плоскость 1, [2] 0 1 1 0 1 1 1 1 0 - плоскость 2, [3] 0 0 0 1 0 0 0 0 0 - плоскость 3, результат 0 0 1 0 0 0 0 1 - шина данных
3CFh	03h	Регистр циклического сдвига данных *(Data Rotate Register)*	Используется при записи и модификации содержимого регистров-защелок: Циклический сдвиг данных и выбор функции: [0-2] - счетчик сдвига влево данных процессора. Используется в режиме записи 0; [3, 4] - выбор функции для режимов записи 0, 2. Разрешают логические операции между данными, считанными из буфера, и данными, записываемыми в память: 00 - данные не изменяются; 01 - ИЛИ; 10 - И; 11 - ИСКЛЮЧАЮЩЕЕ ИЛИ
3CFh	04h	Регистр выбора считываемого банка памяти, т. е. цветовой плоскости *(Read Map Select Register)*	Содержит номер цветовой плоскости (регистра-защелки), данные из которой будут считаны в регистр общего назначения CPU: [0-2] - закодированный номер плоскости, считываемой в процессор. Используется только в режиме считывания 0

Таблица 2.5. Регистры графического контроллера (продолжение)

Порт	Номер (индекс)	Наименование	Назначение
3CFh	05h	Регистр режима *(Mode Register)*	Определяет тип (разновидность) выполняемой операции чтения/записи данных, содержащихся в кадровом буфере: [0, 1]: 00 - режим записи 0. Каждая из четырех плоскостей загружается данными процессора, сдвинутыми согласно счетчику в 3CF.03, если для этих плоскостей нет разрешения установки (сброса). В случае разрешения установки (сброса) плоскости загружаются данными из регистра 3CF.00; 10 - режим записи 1. Каждая из четырех плоскостей загружается данными, считанными в предыдущем цикле; 01 - режим записи 2. Плоскости 0-3 заполняются значениями шины данных процессора DO-D3 соответственно; 11 - режим записи 3. В каждую из плоскостей записывается 8-разрядное число, содержащееся в регистре установки (сброса) для этой плоскости (регистр разрешения установки (сброса) влияния не оказывает). Сдвинутые данные процессора логически суммируются с данными/ находящимися в регистре маски разряда; [3] - режим считывания: 1 - режим считывания 1. Процессор считывает результат логического сравнения между данными четырех плоскостей памяти и содержимым регистра сравнения цвета; 0 - режим считывания 0. Процессор считывает данные из плоскости памяти. Плоскость памяти выбирается регистром 3CF.04; [4] - режим адресации: 1 - режим чет (нечет), 0 - последовательная адресация; [5] - формат занесения данных в регистр сдвига: 1 - режим совместимости с CGA, 0 - режим EGA/VGA; [6] - 256-цветный режим: 0 - загрузка сдвиговых регистров в соответствии со значением 3CF.05 [5]; 1 - загрузка сдвиговых регистров, обеспечивающая 256-цветный режим
3CFh	06h	Вспомогательный регистр *(Miscellaneous Register)*	Используется для задания размера и положения адресного окна, используемого для доступа к видеопамяти (см. табл. 4.1): [0] - режим работы адаптера: 1 - режим графики, 0 - знаковый режим; [1] - сцепление плоскостей: 1 - память делится на две плоскости и нечетная плоскость идет после четной, т.е. для первой плоскости - плоскости 0, 1, для второй плоскости - плоскости 2, 3, 0 - обычная адресация; [2, 3] - адресация буфера в адресном пространстве процессора: 00 - A000 для 128 Кбайт; 10 - A000 для 64 Кбайт; 01 - B000 для 32 Кбайт; 11 - B800 для 32 Кбайт

Таблица 2.5. Регистры графического контроллера (продолжение)

Порт	Номер (индекс)	Наименование	Назначение
3CFh	07h	Регистр безразличного цвета *(Color Don't Care Register)*	Содержит номера цветовых плоскостей, в которых не требуется проводить анализ цвета пиксела: [0-3] - управление сравнением цветов. Работает только в режиме считывания 1. Если разряды 0 - 3 равны нулю, то результатом сравнения в этом случае будут логические единицы на всех разрядах шины данных процессора независимо от считываемой информации, т.е. нет сравнения цветов. Разряды 0 - 3 управляют плоскостями 0 - 3 соответственно
3CFh	08h	Регистр битовой маски *(Bit Mask Register)*	Предназначен для маскирования отдельных пикселов в пределах группы из 8 пикселов: Любой разряд в этом регистре, установленный в нуль, запрещает запись соответствующего разряда в каждую из четырех плоскостей памяти. Данные, записываемые в память, в этом случае будут соответствовать данным, считанным в предыдущем цикле и записанным в контроллере. Любой разряд регистра, установленный в единицу, разрешает различные действия в соответствующем разряде в каждой из четырех плоскостей памяти. Маска разряда подается одновременно на все четыре плоскости

2.1.4.2. Двумерная адресация

Ранее, при описании логической структуры видеопамяти, отмечалось, что разбиение кадрового буфера на цветовые плоскости используется для ускорения его заполнения в 16-цветных графических режимах (в частности, в режиме 12h видеоадаптера VGA). При этом адресация пиксела фактически становится двумерной. Положение пиксела в кадровом буфере характеризуется уже двумя координатами:

- адресом пиксела в пределах цветовой плоскости (смещением ячейки относительно начала плоскости и номером бита в ячейке);
- номером цветовой плоскости (когда речь идет о методе цветовых плоскостей, термины «цветовая плоскость», «битовая плоскость», «банк» и «видеостраница» являются синонимами).

Однако центральный процессор выполняет операции чтения/записи кадрового буфера побайтно, «не подозревая» о его двумерной организации. Поэтому основное назначение графического контроллера заключается в преобразовании двумерных координат пиксела, содержащихся в регистрах-защелках, в одномерные, передаваемые в регистры центрального процессора, и наоборот.

В зависимости от типа адресуемых данных, содержащихся в кадровом буфере, CPU может выполнять операции двух типов:

- *байтные,* предназначенные для изменения содержимого одного или нескольких регистров-защелок (модификация одноименных бит всех восьми пикселов в пределах выбранной ячейки кадрового буфера);
- *пиксельные,* предназначенные для модификации всех бит, задающих цвет выбранных пикселов (доступ осуществляется ко всем цветовым плоскостям одновременно).

Отметим, что CPU не может обращаться к содержимому кадрового буфера, минуя регистры-защелки.

С точки зрения направления перемещения информации, CPU может выполнять операции трех типов:

- считывание байта из кадрового буфера в регистры-защелки *(считывание процессора);*
- запись байта из регистров-защелок в кадровый буфер *(запись процессора);*
- чтение байта из видеопамяти в регистр-защелку, его модификация и запись обратно в кадровый буфер (операция *чтение—модификация—запись).*

Для выполнения последней операции, помимо регистров-защелок, используются дополнительные регистры графического контроллера, а также арифметико-логическое устройство, выполняющее модификацию считанного байта путем выполнения над ним какой-либо логической операции. Такая последовательность действий наиболее эффективна, поскольку не сопровождается пересылкой данных между CPU и видеоадаптером.

2.1.4.3. Двумерное маскирование

Кроме двумерной адресации, графический контроллер обеспечивает *двумерное маскирование,* т. е. маскирование бит в пределах байта и цветовой плоскости в пределах группы из четырех плоскостей. Маскирование пикселов в пределах группы из 8 пикселов осуществляется путем ввода соответствующего значения в регистр битовой маски и используется при записи, а маскирование цветовых плоскостей — путем ввода соответствующего значения в четыре младших разрядах регистра безразличного цвета.

Двумерное маскирование реализуется графическим контроллером в процессе выполнения операций чтения/записи кадрового буфера. Видеоадаптер VGA обеспечивает два режима считывания (*Чтение 0* и *Чтение 1)* и четыре режима записи *(Запись 0 — Запись 3).*

В качестве примера рассмотрим режим Чтение 0 (рис. 2.12), позволяющий процессору определять координаты пикселов заданного цвета.

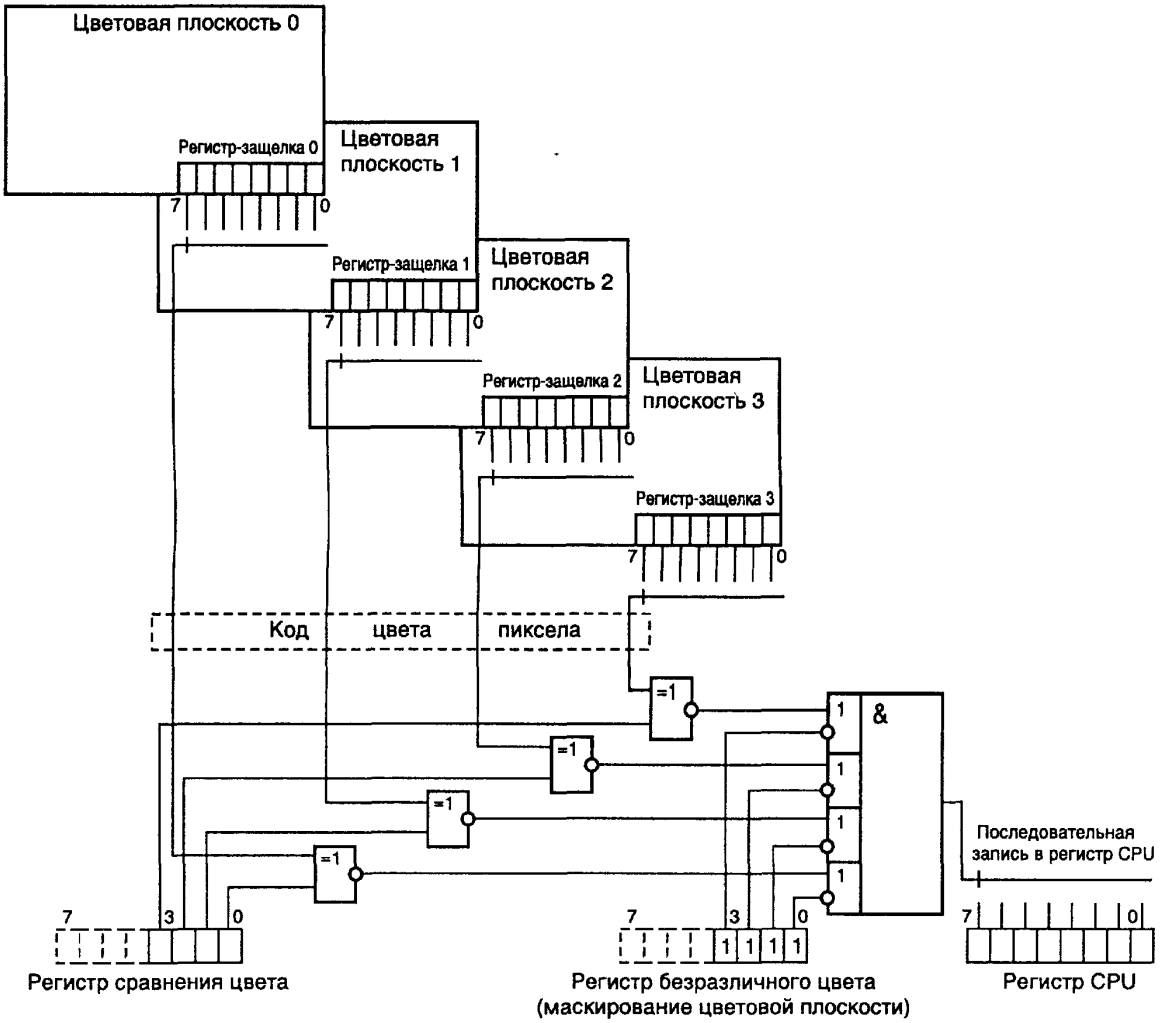


Рис. 2.12. Операция чтения значений 8 пикселов со сравнением (Чтение 0)

В результате выполнения операции чтения в регистры-защелки будут помещены коды цветов восьми соседних пикселов, а в 4 младших разряда регистра сравнения цвета — код заданного цвета. Содержимое четырех одноименных разрядов регистров-защелок (т. е. код цвета одного пиксела) 8 раз сравнивается с содержимым 4 младших разрядов регистра сравнения цвета. В случае совпадения кода пиксела с кодом заданного цвета в i-и разряд регистра процессора (i = 0, .., 7), соответствующий i-му пикселу группы, будет записана 1, а в случае несовпадения — 0. На основании этой информации CPU определяет координаты пиксела с заданным цветом в кадровом буфере. При необходимости анализировать содержимое всех цветовых плоскостей в четыре младших разряда регистра безразличного цвета должно быть записано двоичное число 1111. Если же надо исключить из сравнения какую-либо плоскость, в соответствующий бит этого регистра должен быть записан 0 (маскирование плоскости).

Графический контроллер видеоадаптера VGA можно считать простейшим прообразом графических акселераторов, используемых в современных видеоадаптерах SVGA (см. разд. 2.2.1).

2.1.5. Секвенсер

Секвенсер, или указатель последовательности (контроллер синхронизации), предназначен для

генерации сигналов, необходимых для сканирования видеопамяти. Другими словами, секвенсер обеспечивает последовательную адресацию и считывание содержимого ячеек видеопамяти (отсюда и происходит его название) и передачу их содержимого в контроллер атрибутов и далее — в RAMDAC. Работа секвенсора синхронизируется стробирующими сигналами, формируемыми контроллером ЭЛТ (см. разд. 2.1.3).

Кроме того, секвенсер позволяет выполнять переключения между 8 различными таблицами национальных шрифтов, при этом одновременно могут использоваться два шрифта.

2.1.5.1. Регистры секвенсора.

Секвенсер имеет один индексный (порт 3C4h) и пять управляющих регистров, отображенных в порт 3C5h (см. табл.2.6.). Механизм адресации этих регистров такой же, как в контроллере ЭЛТ и графическом контроллере.

Таблица 2.6. Регистры секвенсора

Порт	Номер (индекс)	Наименование	Назначение
3C4h	-	Индексный регистр *(Address Register)*	Содержит индекс (номер) регистра графического контроллера, к которому будет выполняться обращение
3C5h	00h	Регистр сброса (Reset):	Определяет режим сброса: [0] - асинхронный сброс: 0 - асинхронный сброс и останов, 1 - запуск устройства, если 3C5.00 [1]=1; [1] - синхронный сброс: 0 - синхронный сброс и останов, 1 - запуск устройства, если 3C5.00 [0]=1
3C5h	01h	Регистр режимов синхронизации (Clock Mode):	Определяет режимы синхронизации: [0] - период синхросигнала символа: 0 - равен 8 периодам синхросигнала точки, 1 - 9 периодам синхросигнала точки ; [2] - загрузка/сдвиг: 0, а также 3C5.00 [4] = 1 - загрузка регистров сдвига осуществляется по каждому синхросигналу символа, 1- по каждому второму синхросигналу символа; [3] - частота синхросигналов для регистров сдвига: 0 - равна частоте задающего генератора, 1 - частоте задающего генератора, деленной на 2; [4] - сдвиг на 4: 0 - загрузка регистров сдвига осуществляется по каждому синхросигналу символа, 1 - по каждому четвертому синхросигналу символа; [5] - гашение экрана: 1 - экран видеомонитора гасится с сохранением внутренней синхронизации, 0 - обычный режим работы экрана.
3C5h	02h	Регистр маски плоскости при записи в ОЗУ(Color Plane Write Enable).	Нулевое значение разрядов запрещает запись в соответствующую плоскость: [0] - маска плоскости 0; [1] - маска плоскости 1; [2] - маска плоскости 2; [3] - маска плоскости 3

Таблица 2.6. Регистры секвенсора (продолжение).

<table>
<tr><td>3C5h</td><td>03h</td><td>Регистр выбора номера знакогенератора в знаковом режиме (Character Generator Select):</td><td>[0, 1, 4] - карта B. Выбирается один из восьми знакогенераторов при разряде 3 байта атрибута, равном единице, в соответствии с таблицей:
<table>
<tr><th>[4]</th><th>[1]</th><th>[0]</th><th>Выбранный знакоге-нератор</th><th>Адрес в плоскости 2</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>Первые 8 К</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>Третьи 8 К</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>2</td><td>Пятые 8 К</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3</td><td>Седьмые 8 К</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>4</td><td>Вторые 8 К</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>5</td><td>Четвертые 8 К</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>6</td><td>Шестые 8 К</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>7</td><td>Восьмые 8 К</td></tr>
</table>
[2, 3, 5] - карта A. Выбирается один из восьми знакогенераторов при разряде 3 байта атрибута, равном нулю, в соответствии с таблицей:
<table>
<tr><th>[5]</th><th>[3]</th><th>[2]</th><th>Выбранный знакоге-нератор</th><th>Адрес в плоскости 2</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>Первые 8 К</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>Третьи 8 К</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>2</td><td>Пятые 8 К</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3</td><td>Седьмые 8 К</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>4</td><td>Вторые 8 К</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>5</td><td>Четвертые 8 К</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>6</td><td>Шестые 8 К</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>7</td><td>Восьмые 8 К</td></tr>
</table>
В знаковых режимах разряд 3 байта атрибута обычно служит для управления яркостью символа. Чтобы сделать разряд 3 байта атрибута переключателем набора знакогенераторов, необходимо выполнение следующих условий:
1) 3C5.04 [1]=1;
2) значение разрядов выбора карты A не равно значению разрядов выбора карты B.
В противном случае используются первые 16 К байт плоскости 2. Переключение наборов знакогенераторов с помощью разряда 3 байта атрибута обеспечивает пользователю доступ одновременно к двум знакогенераторам, т.е. к 512 символам вместо обычных 256</td></tr>
<tr><td>3C5h</td><td>04h</td><td>Регистр режима работы памяти (Memory Mode).</td><td>[1] - расширенная память:
1 - вобьем буферной памяти больше 64 К байт,
0 - равен 64 К байт.
Этот разряд должен быть установлен в 1 для использования 256 Кбайт буферной памяти и для обеспечения функции переключения знакогенераторов, как описано выше;
[2] - чет/нечет (1 - последовательная адресация, 0 - режим чет/нечет).
В режиме последовательной адресации информация в каждую плоскость записывается последовательно, байт за байтом. В режиме чет/нечет в четные плоскости записываются четные (младшие) байты, в нечетные - нечетные (старшие) байты;
[3] - двойной чет/нечет</td></tr>
</table>

2.1.6. Контроллер атрибутов

Контроллер атрибутов видеоадаптера VGA предназначен для управления цветом выводимого на экран монитора изображения. В текстовом режиме работы видеоадаптера он, на основании содержимого байта атрибутов выводимого символа, задает цвет пикселов в пределах символьной матрицы. Кроме того, в этом режиме контроллер атрибутов позволяет создавать такие эффекты, как мигание, инверсия цвета или повышенная яркость символа. В графическом 16-цветном режиме контроллер атрибутов преобразует условный 4-разрядный номер цвета пиксела, хранящийся в видеопамяти, в 8-разрядный номер регистра RAMDAC, содержащего 18-разрядный код отображаемого цвета. Другими словами, в графическом режиме контроллер атрибутов управляет *16-цветной палитрой* — специальной таблицей перекодировки, содержащей 16 8-битных чисел. Эта таблица задает 16 одновременно отображаемых оттенков цветов из теоретически доступной 256-цветной палитры. Отметим, что графический режим с 16-цветной палитрой унаследован видеоадаптером VGA от его предшественника — видеоадаптера EGA, поэтому механизм кодирования цвета пиксела в этом режиме у обоих видеоадаптеров имеет много общего (см. разд. 2.1.11).

Помимо управления 16-цветной палитрой, контроллер атрибутов позволяет управлять цветом окаймления экрана (по умолчанию цвет окаймления всегда черный).

В состав контроллера атрибутов входят регистр адреса и 21 регистр данных, отображенные в один и тот же порт 3C0h. При первом обращении к порту 3C0h записываемое число трактуется как индекс регистра, а при следующем — как его содержимое. В связи с этим обмен информацией между центральным процессором компьютера и контроллером атрибутов выполнялся слишком медленно, и в последующие модификации адаптера VGA были внесены изменения. Это позволило организовать доступ к содержимому регистров контроллера атрибутов путем записи в порт 3C0h адреса (индекса) регистра и последующего обращения к порту 3Clh, то есть стало возможным использовать механизм одновременной записи в индексный регистр и регистр данных (эта возможность является недокументированной). Возможны варианты, когда в самом слове, записываемом по адресу 3C0h, указывается признак записываемой информации: адрес или данные, причем при каждой записи данных происходит ароматическое изменение адреса атрибутного регистра [4].

С выхода контроллера атрибутов данные поступают на RAMDAC видеоадаптера в виде восьмиразрядного слова, которое интерпретируется как адрес одного из 256 18-разрядных регистров палитры RAMDAC.

2.1.6.1. Регистры контроллера атрибутов

В таблице 2.7. приведено достаточно подробное описание форматов регистров контроллера палитры видеоадаптера VGA, с указанием их основного назначения.

Таблица 2.7. Назначение регистров контроллера атрибутов

Порт	Номер (индекс)	Наименование	Назначение
3C0h	-	Адресный регистр *(Attribute Address Register)*	Содержит индекс (номер) регистра контроллера атрибутов, к которому будет производиться обращение: [0-4] - адрес регистра назначения; [5] - разрешение отображения: 1 - отображение информации на экране разрешено, 0 - запрещено; [7]: 0 - адрес, 1 - данные (при обращении к 3C0h).
3C0h (3C1h)	00h-0Fh	Регистры палитры *(Palette Registers)*	Определяют цвет отображаемой точки: [0] - голубой (B); [1] - зеленый (G); [2] - красный (R); [3] - голубой интенсивный (BS); [4] - зеленый интенсивный (GS) ; [5] - красный интенсивный (RS) ; [6] - дополнительный (голубой тоновый) (BT) ; [7] - дополнительный (зеленый тоновый (GT)

Таблица 2.7. Назначение регистров контроллера атрибутов (продолжение)

Порт	Номер (индекс)	Наименование	Назначение
3C0h (3C1h)	10h	Регистр управления режимом *(Mode Control Register)*	Управление режимами: [0]: 1 - графический режим, 0 - знаковый режим; [1]: 1 - эмуляция монохромного режима, 0 - эмуляция цветного режима; [2]: 1 - разрешение символов псевдографики при эмуляции монохромного режима, т.е. девятая точка символа совпадает с восьмой точкой символа, 0 - вместо девятой точки символа отображается фон; [3]: 1 - разрешение мерцания, 0 - запрет мерцания и разрешение разряда яркости для кода фона; [5]: 1 - в режиме разделения экрана на два окна признак сравнения линий блокирует регистр поэлементного панорамирования до тех пор, пока не сформируется синхроимпульс кадровой развертки. Это обеспечивает панорамирование только одного окна, 0 - сравнение линий не влияет на панорамирование изображения; [6]: 1 - 256-цветный режим, 0 - все остальные режимы; [7]: 1 - цифровые видеосигналы P7 - P4 являются разрядами 3 - 0 регистра выбора цвета 3C0.14, 0 - цифровые видеосигналы P5 и P4 соответствуют разрядам 5, 4 регистров палитры, а P7 и P6 - разряды 3 и 2 регистра выбора цвета 3C0.14.
3C0h (3C1h)	11h	Регистр цвета окаймления *(Overscan Color Register)*	Содержит 8-разрядный номер одного из регистров RAMDAC, в котором записан код цвета окаймления (границы экрана)
3C0h (3C1h)	12h	Регистр маскирования цветовой плоскости *(Color Plane Enable Register)*	Четыре младших бита используются при маскировании цветовых плоскостей (по умолчанию содержат двоичное число 1111): [0-3]: 1 - разрешение соответствующей плоскости видеопамяти для отображения, 0 - отображение соответствующей плоскости запрещено; [4, 5]: мультиплексор состояния видеовыходов. Состояние разрядов порта 3DA в зависимости от состояния разрядов регистра 3C0.12 приведено в таблице: Разряд регистра 3C0.12 / Разряд порта состояния (3DA) [5] / [4] / [5] / [4] 0 / 0 / R / B 0 / 1 / BS / G 1 / 0 / RS / GS 1 / 1 / GT / BT
3C0h (3C1h)	13h	Регистр пиксельного панорамирования по горизонтали *(Horizontal Pixel Panning Register)*	Применяется совместно с регистрами контроллера ЭЛТ для выполнения горизонтальной прокрутки экрана в графическом и текстовом режимах: [0-3] - число элементов для сдвига видеоданных влево. При константе панорамирования больше 7 сдвиг видеоданных влево не происходит (приравнивается к константе 0)

Таблица 2.7. Назначение регистров контроллера атрибутов (продолжение)

Порт	Номер (индекс)	Наименование	Назначение
3C0h (3C1h)	14h	Регистр выбора цвета *(Color Select Register)*	Используется для быстрой смены текущей палитры: [0, 1] - могут использоваться вместо разрядов 4 и 5 регистров палитры для формирования 8-разрядного цифрового значения кода цвета; [2, 3] - во всех режимах, кроме 256-цветной графики, эти разряды представляют собой старшие (6, 7) разряды 8-разрядного цифрового кода цвета. В режиме 256-цветной графики 8-разрядный атрибут, хранящийся в видеопамяти, становится 8-разрядным цифровым кодом цвета, выдаваемым видео ЦАП

2.1.7.RAMDAC

Наличие устройства, называемого RAMDAC, в составе видеоадаптера VGA является главным отличительным признаком адаптеров этого типа от всех предыдущих (за исключением MCGA). Основная задача RAMDAC — преобразование кода цвета пиксела в аналоговый сигнал. RAMDAC включает в себя следующие элементы (рис. 2.13):

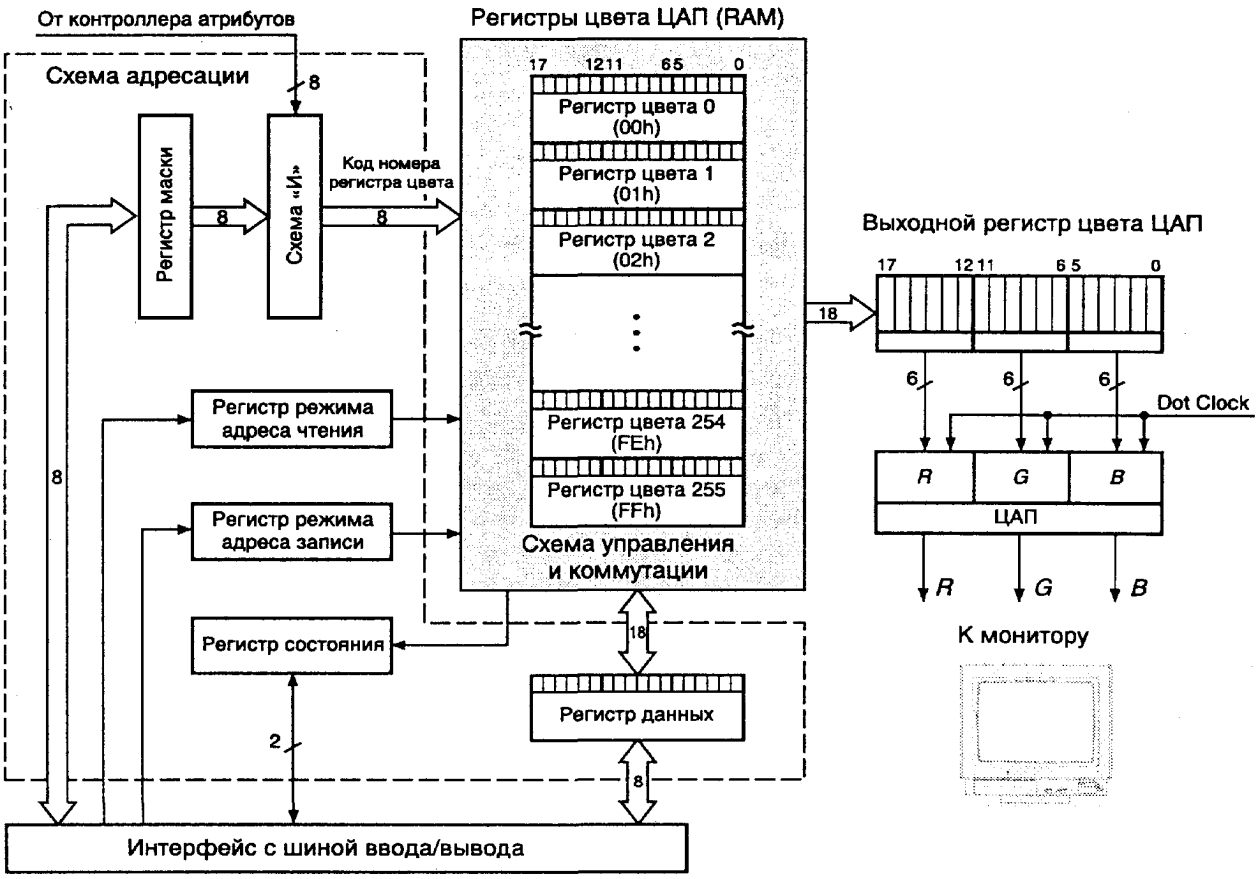


Рис. 2.13. RAMDAC

- трехканальный 6-разрядный ЦАП;
- 256 18-разрядных *регистров цвета* (именно эти регистры и образуют оперативную память, аббревиатура которой — RAM — входит в название данного устройства);
- *выходной 18-разрядный регистр цвета,* выходы которого соединены с соответствующими входами ЦАП;
- *схема адресации.*

Главным элементом RAMDAC является трехканальный ЦАП, количество каналов которого равно количеству основных цветов. Работа ЦАП синхронизируется сигналом Dot Clock тактового генератора видеоадаптера (именно этой частотой фактически определяется верхняя граничная частота спектра

видеосигнала). На входы ЦАП подаются 6-разрядные коды трех основных цветов (R, G и B). На выходе каждого канала ЦАП можно получить $2^6 = 64$ значения выходного напряжения. Нулевому коду на входе канала ЦАП соответствует нулевое напряжение на его выходе, коду 111111 — напряжение около 0,7 В. Разрядность ЦАП определяет количество оттенков цветов, которые потенциально может сформировать видеоадаптер VGA, т. е. размер его палитры. Несложно подсчитать, что палитра составляет $(2^6)^3 = 2^{18} = 262$ 144 оттенков цветов. Оттенок цвета текущего пиксела определяется кодом, записанным в выходной регистр цвета.

На практике количество оттенков цветов, которое видеосистема VGA в состоянии отобразить **одновременно,** значительно меньше размера ее палитры и равно 256. Дело в том, что в выходной регистр цвета нельзя загрузить произвольное число: туда заносятся данные только из регистров цвета, которых в видеоадаптере VGA насчитывается 256. Таким образом, *текущая палитра,* т. е. количество одновременно отображаемых оттенков цветов, ограничена аппаратно. Заметим, что размер палитры определяет максимальное количество разрядов ($8=\log_2 256$), отводимых для кодирования цвета пиксела, и, соответственно, необходимый объем видеопамяти адаптера VGA (256 Кбайт). Видеопамять меньшего объема (например, 128 Кбайт) не позволит реализовать видеорежим 12h (640×480×4 = 1228800 бит = 150 Кбайт > 128 Кбайт), а большего — бесполезна, поскольку при палитре 256 оттенков цветов нет смысла кодировать цвет пиксела более чем 8 битами. Единственная причина, по которой может потребоваться увеличение объема видеопамяти при использовании рассмотренного RAMDAC, — это переход к более высокому, чем предусмотрено стандартом VGA, разрешению. В частности, первые видеоадаптеры SVGA имели видеопамять емкостью 512 Кбайт (режим 800×600/256) или 1 Мбайт (режим 1024×768/256).

2.1.7.1. Регистры цифро-аналогового преобразователя

Регистры цвета ЦАП допускают программное изменение своего содержимого, т. е. CPU может как считывать, так и записывать в них данные. Благодаря этому прикладная программа может изменять текущую палитру, частично компенсируя ее ограниченность. Для реализации этой возможности используется устройство адресации ЦАП, включающее в себя следующие регистры:

- *регистр адреса чтения* (доступен только для записи); при записи в данный регистр 8-разрядного числа становится доступным для чтения регистр цвета, номер которого равен записанному числу;
- *регистр адреса записи* (доступен только для записи); при записи в данный регистр 8-разрядного числа становится доступным для записи регистр цвета, номер которого равен записанному числу;
- *регистр состояния* (доступен только для чтения); значение, содержащееся в двух младших разрядах, определяет направление обмена данными между центральным процессором и ЦАП (чтение или запись);
- *регистр данных* (доступен для чтения и записи); за три цикла обращения к данному регистру CPU может считать (записать) 18-разрядный код, содержащийся в регистре цвета, номер которого предварительно занесен в регистр адреса чтения (записи);
- *Регистр выбора цвета* (входит в состав контроллера атрибутов); позволяет выбирать группы 18-разрядных регистров RAMDAC совместно с содержимым регистров палитры.
- *регистр маски;* позволяет при необходимости ограничить количество используемых регистров цвета ЦАП (т. е. уменьшить текущую палитру), однако на практике эта возможность используется крайне редко.

Более подробная информация об этих регистрах приведена в таблице 2.8.

Таблица 2.8. Регистры цифро-аналогового преобразователя (DAC)

Адрес ввода-вывода	Чтение/ запись	Индекс	Назначение
3C8h	Запись	-	Регистр адреса элемента изображения при записи. Указывает 18-разрядный регистр в таблице преобразования цветов. Каждая операция вывода по этому адресу указывает на дальнейшую последовательность записи, состоящую из трех последовательных операций записи в регистр данных с адресом 3C9. Шесть младших разрядов каждого байта связываются, образуя значение, помещаемое в 18-разрядный регистр данных. Порядок байтов следующий: красный (R), зеленый (G), синий (B). После этого значение регистра адреса элемента изображения автоматически увеличивается на единицу, обеспечивая доступ к следующим ячейкам

Таблица 2.8. Регистры цифро-аналогового преобразователя (DAC) (продолжение).

Адрес ввода-вывода	Чтение/ запись	Индекс	Назначение
3C7h	Запись	-	Регистр адреса элемента изображения при считывании. Каждая операция вывода по этому адресу указывает на дальнейшую последовательность считывания, состоящую из трех последовательных операций считывания из регистра данных с адресом 3C9. Шесть младших разрядов каждого байта, взятых из регистра данных, содержат соответствующий код цвета. Порядок чтения байтов следующий: красный (R), зеленый (G), синий (B). После этого значение в регистре адреса элемента изображения автоматически увеличивается на единицу, обеспечивая доступ к следующим ячейкам
3C7h	Чтение	-	Порт состояния видео ЦАП. [0, 1] - режимы доступа к памяти ЦАП: 00 - чтение из памяти ЦАП; 11 - запись в память ЦАП; 01 и 10 - не определены. [2, 7] - зарезервированы.
3C9h	Чтение/ запись	00h	Регистр данных элемента изображения. Код красного цвета 0
3C9h	Чтение/ запись	00h	Регистр данных элемента изображения. Код зеленого цвета 0
3C9h	Чтение/ запись	00h	Регистр данных элемента изображения. Код голубого цвета 0
3C9h	Чтение/ запись	01h	Регистр данных элемента изображения. Код красного цвета 1
……..	…….	……..	……..
3C9h	Чтение/ запись	FFh	Регистр данных элемента изображения. Код красного цвета FFh
3C9h	Чтение/ запись	FFh	Регистр данных элемента изображения. Код зеленого цвета FFh
3C9h	Чтение/ запись	FFh	Регистр данных элемента изображения. Код голубого цвета FFh
3Clh	Чтение/ запись	14h	Регистр выбора цвета (см. регистры контроллера атрибутов: таблица 2.7., индекс 14h).

С выхода контроллера атрибутов 8-разрядное двоичное число поступает на вход RAMDAC, который воспринимает его как номер одного из 256 регистров цвета. Предварительно это число логически умножается (выполняется операция поразрядного «И») на содержимое регистра маски. Если записать 0 в отдельные разряды регистра маски, можно заблокировать доступ к соответствующим регистрам цвета, т. е. исключить из текущей палитры некоторые оттенки цвета. Например, при установке в 0 младшего разряда регистра маски станут недоступными все регистры цвета с нечетными номерами. По умолчанию в регистр маски записывается двоичное число 11111111, поэтому 8-разрядное число от контроллера атрибутов остается неизменным. Полученный в итоге номер регистра цвета поступает на схему управления и коммутации, которая перемещает содержимое заданного 18-разрядного регистра цвета в выходной регистр цвета, т. е. непосредственно на входы цифро-аналогового преобразователя, после чего на трех выходах ЦАП формируется аналоговый RGB-сигнал.

Такая схема косвенного задания кода цвета пиксела не позволяет увеличить количество одновременно отображаемых цветов даже при увеличении объема видеопамяти. Для расширения текущей палитры в видеоадаптерах SVGA используется *прямое кодирование цвета пиксела,* в соответствии с которым код цвета пиксела в формате RGB хранится в видеопамяти, причем цвет пиксела кодируется уже 15, 16 или 24 битами, и загружается непосредственно в выходной регистр цвета. Это стало возможным благодаря появлению RAMDAC фирмы Sierra Semiconductors, позволяющему загружать код цвета пиксела в выходной регистр цвета непосредственно из видеопамяти, а не из регистров цвета. В результате текущая палитра увеличилась до 2^{16}= 65536 оттенков цветов (режим High Color), а с появлением 24-разрядных RAMDAC — до 2^{24}= 16,7 млн (режим True Color). Кроме того, использование прямого кодирования цвета позволило повысить

быстродействие ЦАП и видеоадаптера в целом, поскольку отпала необходимость в цикле обращения к регистрам цвета ЦАП.

2.1.8. Синхронизатор

Как уже отмечалось, видеопамять видеоадаптера используется двумя устройствами: центральным процессором при обновлении кадрового буфера и самим видеоадаптером (секвенсором) при считывании цифрового образа изображения в процессе сканирования видеопамяти, т. е. во время прямого хода развертки. В подавляющем большинстве моделей видеоадаптеров в качестве видеопамяти используется динамическое ОЗУ, позволяющее в каждый момент времени выполнять только одну операцию: либо считывание, либо запись. Поскольку считывание видеопамяти осуществляется во время прямого хода развертки, на обновление кадрового буфера остается только интервал времени, равный длительности сигнала гашения. В частности, в рассмотренном выше режиме 03h видеоадаптера VGA на обновление кадрового буфера отведено около 6,36 мкс в конце каждой строки и около 1,56 мс в конце каждого кадра. Обычно большинство программ для обновления кадрового буфера увеличивают длительность интервала гашения по вертикали. Если центральный процессор за это время не успел полностью обновить кадровый буфер и завершает эту операцию во время прямого хода развертки, на экране появится «мусор» в виде случайно расположенных черточек и символов, напоминающих снег. Впервые этот недостаток проявился в видеоадаптерах CGA, которыми оснащались компьютеры IBM PC/XT, поэтому его еще называют «снегом CGA» *(CGA snow).*

Для устранения «снега CGA» используются специальные сигналы, информирующие CPU о начале и окончании обратного хода кадровой развертки. В видеосистемах MDA и CGA видеоадаптер в момент начала импульса гашения по вертикали формировал запрос на аппаратное прерывание по линии 2 (IRQ2), который позже был переадресован на линию 9. Поэтому данный сигнал стали обозначать IRQ2/9. Иногда в литературе можно встретить неудачный буквальный перевод названия такого запроса на прерывание — «вертикальное прерывание». Таким образом, центральный процессор обращался к кадровому буферу только во время действия сигнала IRQ2/9. Такой метод синхронизации моментов обновления кадрового буфера замедлял работу видеосистемы. К тому же для его реализации требовалась отдельная линия запроса на прерывание.

В видеоадаптерах EGA и VGA прерывание IRQ2 уже не используется, а синхронизация видеоадаптера и CPU осуществляется специальным устройством — *синхронизатором.* Он управляет доступом CPU к кадровому буферу и разрешает его обновление только во время действия сигналов гашения. В результате устраняется основная причина конфликтов при работе с видеопамятью — асинхронность обращений к ней CPU и внутренних схем видеоадаптера.

Однако для совместимости со старыми программами в новых видеоадаптерах была сохранена возможность генерации запроса на прерывание IRQ2/9.

Для синхронизации обращения CPU к кадровому буферу используется также регистр состояния 1(смю раздел 2.1.11). Бит 3 этого регистра устанавливается в состояние логической 1, если активен сигнал V-Sync, а бит 0 — если активен сигнал разрешения отображения *(Display Enable).* Прикладная программа, считывая биты регистра состояния, обращается к кадровому буферу только тогда, когда вывод информации на экран не производится.

2.1.9. Тактовые генераторы

Работа всех устройств видеоадаптера синхронизируется сигналом Dot Clock или производными от него тактовыми сигналами. Частота сигнала Dot Clock равна верхней граничной частоте выходного видеосигнала. Кроме того, при заданном разрешении этой же частотой определяются значения частот строчной и кадровой синхронизации.

Синхронизация устройств стандартного видеоадаптера VGA в зависимости от выбранного видеорежима осуществляется тактовым сигналом, формируемым одним из двух встроенных генераторов сигналов фиксированных частот 25,175 и 28,322 МГц. Кроме того, предусмотрена возможность использования внешнего тактового генератора (в первых моделях видеоадаптерах VGA, размещавшихся на материнской плате). Выбор тактового генератора осуществляется путем установки в соответствующее состояние разрядов 2 и 3 вспомогательного выходного регистра (порт 3C2h). Значение разрядов данного регистра и соответствующие им частоты синхронизации приведены в табл. 2.9.

Таблица 2.9. Значения частоты тактового сигнала Dot Clock в зависимости от состояния бит 2 и 3 вспомогательного выходного регистра

Бит 2	Бит 3	Частота сигнала Dot Clock, МГц
0	0	25,175
0	1	28,322
1	0	Используется внешний тактовый сигнал

Примечание: эти данные не совпадают с приведенными в таблице 2.10.

Заметим, что частота сигнала Dot Clock должна соответствовать значению полосы пропускания видеотракта подключенного к видеоадаптеру монитора. Например, стандартный монитор видеосистемы

VGA имеет полосу пропускания 30 МГц.

В современных видеоадаптерах, помимо сигнала Dot Clock, используется дополнительный генератор, формирующий сигнал *Memory Clock,* предназначенный для синхронизации работы видеопамяти и набора микросхем (Chipset) видеоадаптера. Частота этого сигнала может изменяться программно в широких пределах. Необходимость использования дополнительного сигнала обусловлена тем, что Chipset видеоадаптера рассчитан на работу с видеопамятью различного быстродействия. На увеличении частоты сигнала Memory Clock основана процедура «разгона» видеоадаптера, заключающаяся в увеличении пропускной способности видеопамяти и, соответственно, повышении быстродействия видеосистемы в целом. В частности, «разгон» большинства распространенных моделей видеоадаптеров SVGA можно выполнить при помощи программы MCLK (аббревиатура от Memory Clock) или PowerStrip (см. [2], разд. 2.1.1).

2.1.10. Интерфейс с шиной ввода/вывода

Видеоадаптер VGA предназначен для работы с шиной ввода/вывода ISA и снабжен специальным интерфейсом, выполняющим следующие функции:

- согласование разрядности внутренней 8-разрядной шины видеоадаптера VGA и 16-разрядной шины ISA компьютера;
- согласование частот тактовых сигналов видеоадаптера (Dot Clock) и шины ISA.

Современные видеоадаптеры подключаются к высокоскоростным шинам (PCI и AGP). Особенности этих шин будут рассмотрены в разд. 2.2.3.

2.1.11 Общие регистры

Большая часть регистров видеоадаптера доступна только в режиме индексной адресации. Особенность общих (внешних) регистров заключается в том, что все они адресуются напрямую, то есть каждый имеет свой собственный адрес порта.

В группу общих регистров входят:

- регистр управления различными режимами работы видеоконтроллера (Miscellaneous Output Register), который доступен только для записи через порт 3C2h;
- регистр управления линиями FC0 и FC1 EGA (Feature Control Register), доступный только для записи через порт 3BAh в монохромном режиме и через порт 3DAh — в цветном;
- нулевой регистр состояния видеоконтроллера (Input Status Register Zero), доступный только для считывания через порт 3C2h;
- первый регистр состояния видеоконтроллера (Input Status Register One), доступный для считывания через порт 3BAh в монохромном режиме и через порт 3DAh — в цветном;
- регистр разрешения работы видеоподсистемы (Video Subsystem Enable Register), доступный для записи и считывания через порт 3C3h.

С точки зрения программиста интерес в этой группе представляет только первый регистр состояния видеоконтроллера (поэтому, наверное, в литературе приводятся весьма противоречивые сведения о назначениях их соответствующих разрядов и о правилах доступа к ним [1, 3-5]). По данным [5] разряды данного регистра имеют следующее значение:

- бит 0 — признак обратного хода луча по строке (0 — обратный ход луча, 1 — прямой ход);
- биты 1 и 2 — не используются;
- бит 3 — признак обратного хода луча по кадру (0 — прямой ход, 1 — обратный ход луча);
- биты 4-7 — не используются.

Бит 3 применяется в компьютерных играх для обеспечения синхронизации работы программного обеспечения и видеоконтроллера (для предотвращения эффекта разрезания изображения), а также при выполнении перезагрузки регистров палитры цифро-аналогового преобразователя (которая должна выполняться во время обратного хода луча). Остальные биты нигде не используются.

В таблице 2.10. приведен вариант описания структуры большинства из общих регистров [4]. Следует отметить, что данные, приведенные в этой таблице, частично противоречат соответствующим данным, представленным в [1, 3]. В частности это касается информации о выбираемых частотах внутренних и внешних генераторов разрядами 2 и 3 регистра управления различными режимами работы видеоконтроллера (Miscellaneous Output Register) с адресом 3C2h.

Таблица 2.10. Общие регистры VGA

Адрес ввода-вывода	Чтение/ запись	Индекс	Назначение
3C2h	Чтение		Нулевой регистр входного состояния (Input Status Register Zero): [0-3] - не используются. [4] - состояние четырехразрядного переключателя режимов. Номер считываемого разряда переключателя задан разрядами 2 и 3 порта 3C2 по записи. [5] и [6] - не используются; [7] - прерывание от контроллера CRT: 0 - обратный ход по кадру; 1 - отображение информации
3BAh/ 3DAh	Чтение		Первый регистр входного состояния (Input Status Register One): [0] - разрешение отображения информации: 0 - разрешено отображение, 1 - запрещено; [1] - триггер светового пера; [2] - переключатель светового пера: 0 - нажат, 1 - отжат; [3] - обратный ход по кадру: 0 - прямой, 1 - обратный; [4-5] - диагностирование видеовыходов монитора (в зависимости от значения разрядов 4 и 5 регистра 3C0.13);
3C2h 3CCh	Запись Чтение		Регистр управления режимом (регистр смешанного вывода - Miscellaneous Output Register). [0] - адреса портов ввода-вывода CRT: 0 - 3BXh, 1 - 3DXh (монохромный или цветной режимы). [1] - разрешение обращения процессора к видеопамяти: 0 - память заблокирована для записи (считывания) процессором; 1 - обращение к видеопамяти разрешено. [3, 2] - выбор частоты задающего генератора: 00 - частота 14,318 МГц; 01 - частота 16,257 МГц; 10 - частота 25,172 МГц; 11 - частота 32,514 МГц. Разряды 3, 2 в случае выбора считываемого разряда переключателя режимов работают вместе с разрядом 4 дополнительного регистра 3C2.04. В этом случае состояние [4, 3, 2] определяет соответственно: 000 - 1-й разряд переключателя 001 - 2-й разряд переключателя 010 - 3-й разряд переключателя 011 - 4-й разряд переключателя 100 - 5-й разряд переключателя 101 - 6-й разряд переключателя 110 - 7-й разряд переключателя 111 - 8-й разряд переключателя. [4] - бит отключения видеосигнала. [5] - страница для режима чет (нечет): 0 - адресация по плоскостям, 1 - четный адрес - плоскость 0, нечетный - плоскость 1. [6] - полярность строчного синхроимпульса: 0 - полярность положительная; 1 - полярность отрицательная. [7] - полярность кадрового синхроимпульса: 0 - полярность положительная; 1 - полярность отрицательная.
3C3h	Чтение/ запись	-	Регистр разрешения работы видеосистемы VGA (Video Subsystem Enable Register).

2.1.12. Кодирование цвета пиксела в различных видеорежимах

В отличие от первых видеоадаптеров фирмы IBM (MDA, CGA), в видеоадаптере VGA используется весьма гибкая и сложная схема кодирования цвета пиксела. В зависимости от используемой схемы

различают следующие видеорежимы:

- 16-цветный (текстовый или графический);
- 256-цветный графический 13h;
- монохромный;
- монохромный (двухцветный), эмулирующий режим работы видеоадаптера MDA (текстовый режим) или CGA (графический режим).

Рассмотрим эти видеорежимы более подробно.

2.1.12.1. 16-цветные режимы

К 16-цветным режимам относятся все текстовые режимы работы видеоадаптера VGA (за исключением режима 07h, в котором эмулируется режим работы видеоадаптера MDA), а также графические режимы 0Dh (320×200/16), 0Eh (640×200/16), 10h (640×350/16) и 12h (640×480/16). Все перечисленные графические режимы, за исключением 12h, поддерживаются видеоадаптером EGA, при этом схемы кодирования цвета пиксела в видеоадаптерах EGA и VGA в основном совпадают. Кроме того, в этих адаптерах используются практически те же самые регистры (за исключением RAMDAC). Благодаря этому обеспечивается полная совместимость программного обеспечения сверху вниз: программы, написанные для видеосистемы EGA, успешно работают с видеосистемой VGA.

Схема формирования цвета пиксела в 16-цветных режимах работы видеоадаптера VGA представлена на рис. 2.14. В основном она справедлива как для текстового, так и для графического режима работы видеоадаптера. Различия заключаются только в способе формирования 4-разрядного кода цвета пиксела и будут рассмотрены ниже.

В графических 16-цветных режимах видеопамять разбивается на 4 цветовых плоскости. Контроллер атрибутов считывает 4-битный код цвета пиксела с учетом его маскирования (см. разд. 2.1.6, табл. 2.7). Использование маскирования цветовых плоскостей позволяет легко уменьшить число отображаемых цветов — получить 4- или 2-цветное изображение. В 16-цветных режимах младшие разряды регистра безразличного цвета содержат двоичное число 1111, поэтому маскирование цветовых плоскостей не производится (используются все четыре цветовые плоскости). Полученный четырехразрядный код воспринимается контроллером атрибутов как номер регистра палитры (напомним, что контроллер атрибутов содержит 16 таких регистров). В каждом из регистров палитры записан 6-разрядный код, соответствующий определенному цвету (оттенку цвета), при этом два старших разряда регистров палитры не используются.

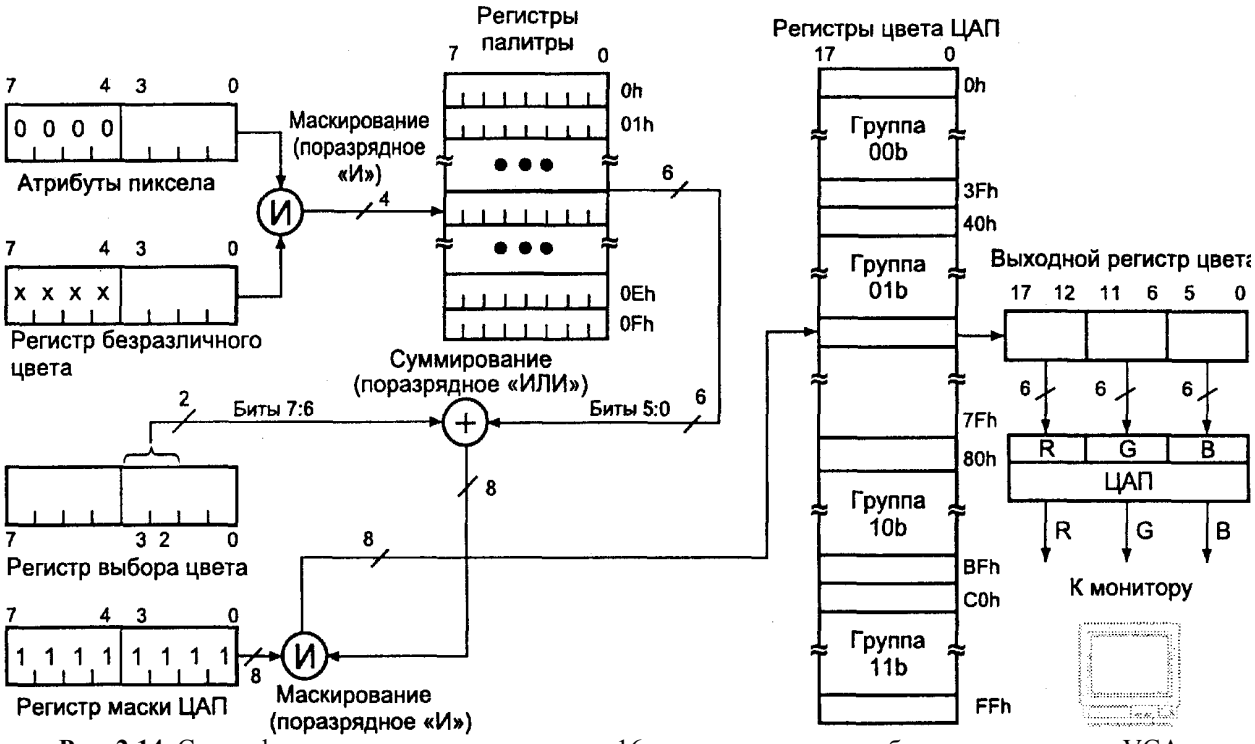


Рис. 2.14. Схема формирования цвета пиксела в 16-цветных режимах работы видеоадаптера VGA

Подобная схема кодирования досталась видеоадаптеру VGA в наследство от его предшественника — видеоадаптера EGA. Напомним, что в составе видеосистемы EGA использовался цифровой монитор, на ЭЛТ которого видеосигнал подавался по 6 линиям: трем основным *(R, G, B)* и трем дополнительным *(r, g, b)*. Такую схему формирования цвета принято называть *RrGgBb*. В видеоадаптере EGA контроллер атрибутов

являлся выходным устройством, т. е. на ЭЛТ монитора 6-разрядный код оттенка цвета в формате *RrGgBb* подавался непосредственно с выходов регистров палитры.

По умолчанию в регистры палитры контроллера атрибутов загружаются коды следующих 16 цветов:

- основных (синего, красного, зеленого): 001, 010 и 100 соответственно;
- неосновных из первой половины таблицы; получаются путем суммирования кодов основных цветов, причем в результате суммирования получаются такие цвета, какие получились бы при аддитивном сложении цветов: например, при сложении кодов красного (100) и синего (001) цветов, т. е. 101 = 100 + 001, получится пурпурный (101);
- цветов второй половины таблицы (с номерами 8—15), являющихся более яркими оттенками цветов 0—7; получаются при одновременной активизации сигналов трех дополнительных цветов.

Содержимое регистров палитры по умолчанию представлено в таблице. 2.11.

Таблица 2.11. Содержимое регистров палитры контроллера атрибутов по умолчанию

Номер регистра (условный номер цвета)	Содержимое регистра						Цвет
	r	g	b	R	G	B	
00h (0)	0	0	0	0	0	0	Черный
01h (1)	0	0	0	0	0	1	Синий
02h (2)	0	0	0	0	1	0	Зеленый
03h (3)	0	0	0	0	1	1	Голубой
04h (4)	0	0	0	1	0	0	Красный
05h (5)	0	0	0	1	0	1	Пурпурный
06h (6)	0	0	0	1	1	0	Коричневый
07h (7)	0	0	0	1	1	1	Белый
08h (8)	1	1	1	0	0	0	Темно-серый
09h (9)	1	1	1	0	0	1	Светло-синий
0Ah (10)	1	1	1	0	1	0	Светло-зеленый
0Bh (11)	1	1	1	0	1	1	Светло-голубой
0Ch (12)	1	1	1	1	0	0	Светло-красный
0Dh (13)	1	1	1	1	0	1	Светло-пурпурный
0Eh (14)	1	1	1	1	1	0	Желтый
0Fh (15)	1	1	1	1	1	1	Ярко-белый

Примечание. В скобках указаны десятичные номера регистров палитры.

Заметим, что одновременная активизация сигналов *r, g* и *b* для формирования цветов с номерами 8 - 15 эквивалентна применению цветовой модели IRGB, реализованной в видеосистеме CGA (см. табл. 1.1). Выбор такой схемы формирования палитры позволяет обеспечить совместимость старых программ, написанных для видеосистемы CGA, с более современными видеосистемами (EGA, VGA). Вместе с тем содержимое регистров палитры всегда можно изменить программным путем, поскольку эти регистры доступны CPU как для чтения, так и для записи. В частности, для модификации регистров палитры контроллера атрибутов может использоваться видеосервис BIOS (подфункция 00h функции 10h программного прерывания INT 10h).

В видеоадаптере VGA, в отличие от EGA, 6-разрядный код, хранящийся в регистрах палитры, является не выходным кодом цвета, а номером одного из 256 регистров цвета ЦАП. В зависимости от состояния 8-го разряда (бит 7) регистра управления режимом контроллера атрибутов (индекс 10h, табл. 2.7) и кода, содержащегося в регистрах палитры, возможны два способа определения номера регистра цвета ЦАП:

1. Бит 7 равен 0. В этом случае контроллер атрибутов добавляет к 6-разрядному коду регистра палитры два старших разряда, в качестве которых используются биты 2 и 3 регистра выбора цвета (индекс 14h, см. табл. 2.7). Такой способ определения номера регистра цвета ЦАП эквивалентен разбиению этих регистров на 4 группы по 64 регистра каждая. Номер регистра ЦАП в пределах группы задается содержимым регистров палитры, а номер группы — битами 2 и 3 регистра выбора цвета. Изменяя состояние последних, можно быстро менять фактическую палитру, не прибегая к модификации содержимого регистров палитры контроллера атрибутов.

2. Бит 7 равен 1. Из регистров палитры считываются не 6, а только 4 младших разряда (биты 0 - 3), к которым затем добавляются в качестве старших биты 0 - 3 регистра выбора цвета. В этом случае 256 регистров цвета ЦАП разбиваются на 16 групп по 16 регистров каждая. Номер группы определяется содержимым четырех младших разрядов регистра выбора цвета, а номер конкретного регистра цвета ЦАП в пределах группы — содержимым четырех младших разрядов регистра палитры.

Заметим, что в обоих случаях максимальное количество отображаемых цветов равно 16 и 8-разрядный номер регистра цвета ЦАП, формируемый контроллером атрибутов, подвергается в RAMDAC маскированию.

Как уже отмечалось, рассмотренная схема формирования цвета пиксела в текстовом режиме отличается способом определения 4-разрядного кода одного из 16 регистров палитры, который получает контроллер атрибутов. В графическом режиме этим числом является результат считывания содержимого четырех битовых плоскостей. В текстовом режиме для определения цвета пиксела в пределах символьной матрицы используется байт атрибутов (см. разд. 1.1.2). Его младшие 4 бита описывают цвет символа, а старшие — цвет фона, причем цвет задается в формате IRGB (рис. 2.15). Кроме того, бит 7 управляет мерцанием символа.

Используя изображение символа, формируемое знакогенератором, контроллер атрибутов определяет, к какому элементу символьной матрицы относится выводимый пиксел: к переднему плану (т. е. собственно символу) или к заднему (т. е. фону). В соответствии с этим код цвета пиксела задается либо младшей, либо старшей тетрадой байта атрибутов. Так же как и в графическом режиме, выбранный 4-разрядный код маскируется при помощи регистра безразличного цвета.

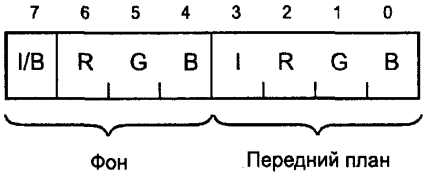


Рис. 2.15. Структура байта атрибутов

16-цветные режимы видеоадаптера VGA обеспечивают очень гибкое, многоуровневое управление палитрой, что позволяет частично скомпенсировать ограниченность текущей палитры. Вместе с тем работа с цветовыми плоскостями (в графических режимах) значительно усложняет программирование доступа CPU к видеопамяти.

2.1.12.2. 256-цветный режим (13h)

В этом графическом видеорежиме (320×200/256) не используется разбиение видеопамяти на цветовые плоскости, поэтому код цвета пиксела хранится в восьми соседних разрядах, т. е. в одной 8-разрядной ячейке видеопамяти. Регистры палитры контроллера атрибутов также не используются. Считываемое из видеопамяти 8-разрядное число сразу передается контроллером атрибутов в RAMDAC и воспринимается последним как номер одного из регистров цвета ЦАП (с учетом маскирования). Роль регистров палитры в этом режиме выполняют регистры цвета ЦАП. Поскольку они являются 18-разрядными (см. рис. 2.9), палитра видеоадаптера VGA в режиме 13h составляет 2^{18}= 262144 оттенков цветов. Текущая палитра, содержащая 256 оттенков цветов (по числу регистров цвета ЦАП), может изменяться программно, при помощи функций видеосервиса BIOS (подфункция 10h или 12h функции 10h программного прерывания INT 10h).

2.1.12.3. Монохромные режимы работы

Одной из уникальных возможностей видеоадаптера VGA является способность корректно преобразовывать цветное изображение в черно-белое (с градациями серого цвета). Для выполнения этого преобразования используется метод, называемый *взвешенным усреднением цветов.* Он заключается в том, что по 6-разрядным значениям трех основных цветов *(R, G* и *B),* образующим цветовой оттенок пиксела, вычисляется *одно* 6-разрядное значение яркости *(Y)* пиксела черно-белого изображения. Это значение заносится во все три 6-разрядных элемента выходного регистра цвета ЦАП. В этом случае на трех входах каналов ЦАП оказывается одно и то же число К. Очевидно, что сформированные на выходах ЦАП аналоговые сигналы *R, G* и *B* также будут одинаковы. В результате три зерна люминофора в пределах триады будут иметь одинаковую яркость свечения и ЭЛТ цветного монитора будет формировать оттенки только серого цвета.

Как известно, человеческий глаз по-разному воспринимает объекты, окрашенные в различные цвета. Эта зависимость описывается *кривой относительной спектральной чувствительности глаза* (рис. 2.16).

В соответствии с данной кривой, при вычислении средневзвешенного значения яркости используются следующие весовые коэффициенты:

$$Y = 0{,}3R + 0{,}59G + 0{,}11B.$$

Для реализации взвешенного усреднения цветов используется подфункция 1Bh функции 10h прерывания INT 10h (т. е. видеосервис BIOS). Она позволяет задать группу регистров цвета ЦАП, считать их содержимое, вычислить для каждого регистра 6-разрядное средневзвешенное значение *Y* и загрузить его обратно в регистр (в виде числа YYY). Исходное содержимое регистров цвета ЦАП при этом теряется.

Таким образом, использование взвешенного усреднения цветов, фактически приводящее к коррекции текущей палитры, позволяет превратить любой цветной режим работы видеоадаптера VGA в монохромный.

При подключении к видеоадаптеру VGA монохромного монитора взвешенное усреднение цветов выполняется автоматически (видеоадаптер VGA автоматически распознает монохромный монитор — см. [2] разд. 2.4.1, табл. 2.9). В этом случае видеосигнал монохромного изображения передается по проводу 2 (канал

зеленого цвета G).

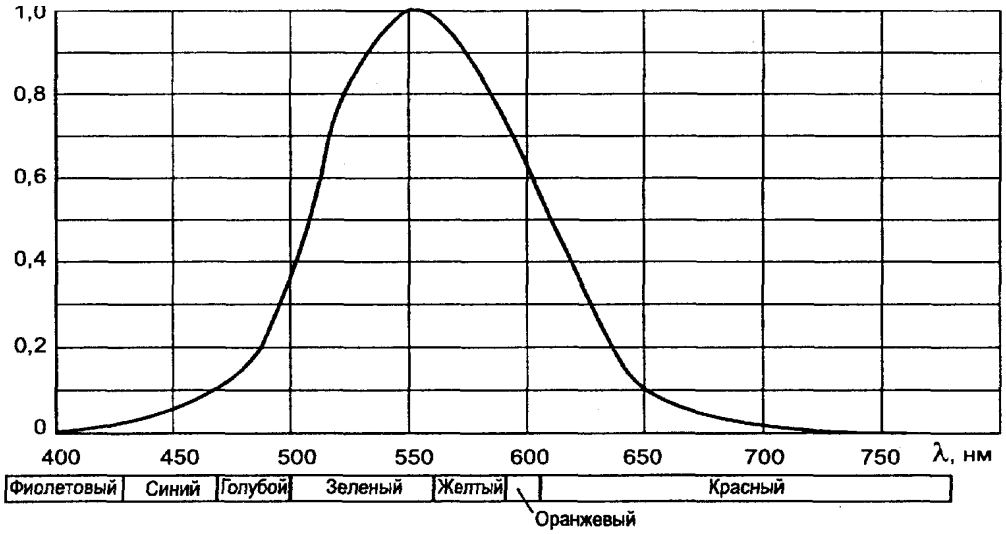


Рис. 2.16. Кривая относительной спектральной чувствительности глаза

Взвешенное усреднение цветов можно также запретить (разрешить) путем вызова подфункции 33h функции 12 прерывания INT 10h (см. раздел 2.1.2.2.).

Поскольку в видеоадаптере VGA имеется 3-канальный 6-разрядный ЦАП, на выходе каждого канала можно получить 2^6= 64 градации напряжения. Следовательно, при использовании взвешенного усреднения цветов палитра видеоадаптера VGA составит 64 оттенка серого. В режиме 12h может быть одновременно отображено только 16 оттенков из этой палитры, а в режиме 13h — все 64. Однако в последнем случае видеопамять адаптера расходуется нерационально: для кодирования оттенка пиксела достаточно б бит вместо 8 используемых.

Большинство видеоадаптеров SVGA оснащены 24-разрядными ЦАП (по 8 разрядов на каждый канал). Такие видеоадаптеры способны сформировать 2^8= 256 оттенков серого.

2.1.12.4. Монохромные (двухцветные) режимы работы

Двухцветные режимы работы видеоадаптера VGA используются весьма редко — в основном для обеспечения совместимости с программами, написанными для видеоадаптеров MDA и CGA.

В двухцветном текстовом режиме 07h (80×25 символов, 720×400/2) видеоадаптер VGA эмулирует режим работы видеоадаптера MDA. Это достигается путем изменения способа интерпретации содержимого байта атрибутов (в частности, вместо использования 16-цветной палитры применяются эффекты мерцания, инверсии, повышенной яркости и подчеркивания символов).

В двухцветном графическом режиме 06h (640×200/2) видеоадаптер VGA эмулирует работу видеоадаптера CGA. Сокращение цветовой палитры до двух цветов достигается за счет маскирования цветовых плоскостей (запрещаются три плоскости из четырех). Кроме того, путем изменения регистров палитры контроллера атрибутов и содержимого регистров цвета ЦАП можно получить не только черно-белое, но и любое двухцветное изображение (например, черно-красное). Аналогичным образом видеоадаптер VGA работает в режимах 0Fh (640×350/2) и 11h (640×480/2).

2.1.13. Недостатки видеоадаптера VGA

Появление видеосистемы VGA явилось качественным скачком в развитии видеосистемы PC. Принцип действия, положенный в ее основу (управление монитором аналоговым RGB-сигналом), обеспечил значительное сближение PC с телевизионной и видеоаппаратурой, ускорив наступление эры мультимедиа. Однако возможности стандартного видеоадаптера VGA оказались ограниченными, и очень скоро он перестал удовлетворять требованиям, предъявляемым мультимедийными приложениями к видеосистеме PC. Наиболее серьезными недостатками стандартного видеоадаптера VGA были следующие:

- ограниченная палитра и низкое разрешение экрана;
- низкое быстродействие видеосистемы;
- недопустимо высокая загрузка центрального процессора задачами формирования изображения, отвлекающая его от решения других задач.

С точки зрения традиционных офисных приложений, с этими недостатками можно было смириться (в самом деле, зачем для работы в Word for Windows 16,7 млн оттенков цветов?). К тому же задача увеличения числа цветов и разрешения экрана была довольно быстро решена с появлением первых адаптеров SVGA, у которых объем видеопамяти составил 512 Кбайт и 1 Мбайт. Однако быстродействие этих адаптеров,

особенно при высоких разрешении и глубине цвета, оставляло желать лучшего. Медленная прорисовка изображения в режимах с высоким разрешением и большим количеством цветов и заметная дискретность кадров при формировании динамических изображений в играх лишали их смысла, поскольку не обеспечивали интерактивного взаимодействия в реальном масштабе времени. Требовалось радикально повысить быстродействие работы видеоадаптера.

Каковы причины низкого быстродействия стандартного адаптера VGA, а также адаптеров SVGA первых моделей? Таких причин несколько:

- пассивность видеоадаптера VGA при формировании и модификации кадрового буфера, приводящая к необходимости привлечения центрального процессора для реализации практически всех графических функций;
- ограниченное быстродействие центрального процессора, который, параллельно выполняя другие задачи, не успевал своевременно обновлять кадровый буфер видеоадаптера;
- недостаточное быстродействие видеопамяти (большое время доступа, отсутствие возможности одновременного считывания и записи данных, сложная логическая организация видеопамяти, затрудняющая доступ к ней центральному процессору);
- низкая пропускная способность внутренней шины данных видеоадаптера;
- недостаточная пропускная способность шины ввода/вывода PC, по которой осуществляется пересылка данных из ОЗУ в видеопамять;
- недостаточное быстродействие RAMDAC;
- отсутствие аппаратной поддержки дополнительных функций, используемых в мультимедийных приложениях (воспроизведение видеоданных, прием телевизионных сигналов и т. п.).

Фактически эти недостатки стандартного видеоадаптера VGA и определили основные направления его совершенствования.

Контрольные вопросы.

1. Чем отличаются современные видеоадаптеры от стандартного видеоадаптера VGA?
2. Какие характеристики видеосистемы определяет видеоадаптер?
3. Что используется в современных видеоадаптерах для повышения производительности видеосистемы?
4. Какие дополнительные мультимедийные функции выполняет современный видеоадаптер?
5. Где впервые появился видеоадаптер VGA и в каком исполнении?
6. Что понимается под VGA-совместимым и стандартным VGA видеоадаптером?
7. Какие элементы содержит видеоадаптер VGA?
8. Какие разновидности регистров содержатся в элементах видеоадаптера VGA?
9. Для чего используется видеопамять и для чего введено понятие "локальная видеопамять"?
10. Каков объем видеопамяти VGA и какие графические режимы он может поддерживать?
11. Как, при работе процессора в режиме реальной адресации, получается 20-разрядный физический адрес из содержимого двух 16-разрядных регистров; как называются эти регистры?
12. Что понимается под стандартной и верхней памятью DOS?
13. Как можно осуществить доступ к расширенной памяти?
14. Что можно сказать об использовании защищенного режима работы?
15. В чем особенность работы с аппаратурой в защищенном режиме?
16. Что понимается под линейным режимом адресации?
17. Когда можно пользоваться линейным режимом адресации?
18. Когда была введена поддержка линейного режима адресации?
19. Что позволяет метод Томаса Родена?
20. Опишите процедуру переключения одного из дополнительных сегментных регистров в режим линейной адресации.
21. Охарактеризуйте поля дескриптора сегмента.
22. Охарактеризуйте формат байта прав доступа для сегмента данных.
23. Охарактеризуйте формат регистра управления CR0.
24. Охарактеризуйте назначение подпрограмм, представленных в листинге 2.1.
25. Какие сегменты UMA используются для обращения к видеопамяти и как их используют различные видеосистемы?
26. В каких компьютерах можно использовать два видеоадаптера?
27. Как организуется доступ к видеобуферу, когда его размер больше 128 Кбайт?
28. Охарактеризуйте формат PDF.
29. Охарактеризуйте метод плоскостей.
30. Чем определяется разрядность видеопамяти?
31. Что понимается под Video BIOS, где она располагается и какой объем памяти занимает?
32. Что такое "теневая память"?

33. Что понимается под видеосервисом?
34. Как обрабатываются аппаратные и программные прерывания?
35. Охарактеризуйте основные функции видеосервиса.
36. Какие задачи решает контроллер ЭЛТ?
37. Сколько программно-доступных регистров содержит контроллер ЭЛТ и через какие адреса осуществляется к ним доступ?
38. Как можно сменить видеорежим, и как можно подстроить растр экрана после смены режима?
39. Для чего предназначен графический контроллер и чем обусловлено его применение?
40. Какие элементарные операции помогает выполнять центральному процессору графический контроллер?
41. Какие элементы входят в состав графического контроллера?
42. В чем заключается двухмерность адресации пиксела в 16-цветных графических режимах VGA?
43. В чем основное назначение графического контроллера?
44. Какие типы операций может выполнять CPU с содержимым кадрового буфера через регистры-защелки?
45. Какие операции может выполнять CPU с точки зрения направления передачи информации между кадровым буфером и регистрами-защелками?
46. Что такое двумерное маскирование и как оно осуществляется?
47. Сколько режимов записи и чтения обеспечивает видеоадаптер VGA?
48. Что позволяет получить режим чтения 0 и как он выполняется?
49. Что такое "секвенсор", для чего он используется в VGA и сколько программно-доступных регистров он содержит?
50. Какие функции выполняет контроллер атрибутов в текстовом режиме?
51. Какие функции выполняет атрибутный контроллер в графическом 16-цветном режиме?
52. Сколько программно-доступных регистров содержит контроллер атрибутов, и как осуществляется к ним доступ?
53. Каково назначение RAMDAC, и какие элементы входят в его состав?
54. Сколько значений напряжений можно получить на выходе каждого канала ЦАП и в каком диапазоне?
55. Какова палитра VGA, сколько оттенков может отображаться одновременно и почему?
56. Какие регистры используются для изменения содержимого регистров цвета?
57. Как можно получить одновременное отображение 65536 и 16,7 млн. атрибутов?
58. Чем отличались первые видеоадаптеры SVGA от VGA?
59. Какие интервалы времени обычно отводятся под обновление кадрового буфера?
60. Что понимается под термином "снег CGA"?
61. Что понимается под сигналом IRQ2/9?
62. В каких видеоадаптерах стал применяться синхронизатор и для чего?
63. О чем информируют биты 3 и 0 регистра состояния видеоадаптера?
64. Что определяет частота сигнала Dot Clock, и какое значение она может принимать в VGA?
65. Как происходит переключение частот сигнала Dot Clock?
66. Для чего используется сигнал Memory Clock в современных видеоадаптерах SVGA?
67. На какую шину ввода/вывода персонального компьютера рассчитан интерфейс видеоадаптера VGA и какие основные функции он выполняет?
68. В каких видеорежимах может работать видеоадаптер VGA?
69. Перечислите 16-цветные режимы работы VGA; какие из этих видеорежимов не поддерживает видеоадаптер EGA?
70. Как формируется содержимое регистров палитры 16-цветного режима по умолчанию?
71. Как используется 6-разрядный код регистров палитры в видеосистемах EGA и VGA?
72. Как определяется номер регистра цвета ЦАП при нулевом значении бита 7 регистра управления режимом контроллера палитры?
73. Тоже самое при единичном значении.
74. Как формируется цветность в текстовом режиме?
75. Опишите формирование цвета в 256-цветном режиме (13h)/
76. Что такое "взвешенное усреднение цвета"?
77. Как формируется монохромное изображение в VGA?
78. Сколько градаций серого формируется в режимах 12h, 13h и в SVGA?
79. Как реализуется монохромный двухцветный режим в видеосистеме VGA, работающей в режиме 07h (текстовый) и 06h (графический)?
80. Перечислите основные недостатки видеоадаптера VGA.
81. Каковы причины низкого быстродействия стандартного видеоадаптера VGA?

В данном разделе были использованы данные из [1, 3 - 5, 15]. Описание подфункций прерывания Int 10h для видеосистем HGC, HGC+ и MCGA можно найти в [15].

2.2. Основные направления совершенствования видеоадаптера VGA. Видеоадаптеры SVGA

Стремясь устранить недостатки стандартного видеоадаптера VGA и получить тем самым преимущество в конкурентной борьбе на рынке, многочисленные фирмы-производители компьютерного оборудования принялись самостоятельно совершенствовать исходную архитектуру видеоадаптера, разработанную фирмой IBM. В результате появилось множество моделей, основанных на различных технических решениях, но относящихся к одному новому типу видеоадаптеров — Super VGA, или SVGA.

Наиболее простой и очевидный способ совершенствования видеоадаптера — увеличение объема видеопамяти на его плате. Пионером в реализации такого технического решения условно считается японская фирма NEC, выпустившая видеоадаптер, оборудованный видеопамятью объемом 512 Кбайт и обеспечивающий видеорежимы 640×480/256 и 800×600/256. Одновременно с увеличением объема видеопамяти пришлось использовать новые методы ее адресации, поскольку количество пикселов на экране превысило размер адресного пространства (128К), отводимого для обращения к видеопамяти (напомним, что в 256-цветном режиме каждому пикселу на экране соответствует одна 8-разрядная ячейка видеопамяти). Практически все производители стали использовать механизм переключения банков памяти подобно тому, как это реализовано в методе цветовых плоскостей, однако теперь в каждом банке может храниться как битовая плоскость (в 16-цветных режимах), так и часть кадрового буфера в формате PDF (в 256-цветных режимах). Увеличение объема памяти позволило повысить разрешение, но не привело к улучшению цветности изображения (размер текущей палитры по-прежнему оставался ограниченным 256 оттенками цветов).

Важным шагом на пути улучшения цветности изображения стал выпуск фирмой Sierra Semiconductors нового RAMDAC, который имел два принципиальных отличия от RAMDAC видеоадаптера VGA:

1. Новый RAMDAC позволял загружать в выходной регистр ЦАП данные непосредственно из видеопамяти, минуя регистры цвета ЦАП. Прямое кодирование цвета пиксела позволило использовать более 8 бит на пиксел, в результате чего было устранено присущее видеоадаптеру VGA ограничение на количество (256) одновременно воспроизводимых оттенков цветов. В итоге появились новые видеорежимы *High Color,* в которых кодирование оттенка цвета пиксела осуществлялось 15 или 16 разрядами, при этом обеспечивалось отображение 32 768 и 65 536 оттенков цветов соответственно.

2. В новом RAMDAC использовался 24-разрядный ЦАП вместо 18-разрядного (в видеоадаптере VGA). Благодаря этому стал доступен режим максимально правдоподобного воспроизведения цвета, получивший название *True Color.* Количество воспроизводимых в данном режиме оттенков цветов составило 2^{24}= 16777216.

На базе такого RAMDAC стали выпускаться видеоадаптеры, оборудованные видеопамятью емкостью 1 Мбайт и более (дело в том, что 512 Кбайт видеопамяти недостаточно для реализации режима High Color даже при минимальном разрешении 640×480).

Вместе с объемом видеопамяти возросла и разрядность ее ячеек (16 бит и более), а также разрядность и пропускная способность внутренней шины данных видеоадаптера. Вместо разбиения видеопамяти на банки стала применяться новая схема линейной адресации, но уже в пределах всей видеопамяти, а не одного банка, как в видеоадаптере VGA.

При использовании этой схемы видеоадаптер выполняет переадресацию всей видеопамяти на непрерывную область адресов, расположенную в расширенной (extended) памяти. В видеоадаптерах SVGA с интерфейсом ISA этот диапазон располагался под границей 16 Мбайт, а с интерфейсом PCI — в любой области расширенной памяти. Благодаря использованию линейной адресации, центральному процессору уже нет необходимости использовать сложный и медленный механизм переключения банков видеопамяти, он просто выставляет на шине адреса код адреса, а видеоадаптер самостоятельно выполняет переадресацию и обеспечивает доступ к соответствующей ячейке видеопамяти. Это позволило значительно упростить адресацию кадрового буфера и ускорить процесс обращения к нему центрального процессора. Естественно, что для реализации такой схемы требуется CPU не ниже Intel 80386SX.

Однако принципиально проблема низкого быстродействия видеоадаптера решена не была. Этому мешали два обстоятельства:

- все операции по обновлению кадрового буфера по-прежнему выполнял центральный процессор, что резко снижало общее быстродействие PC;
- с переходом к более высоким разрешению и глубине цвета многократно возрос объем пересылаемых в видеопамять данных, и пропускная способность стандартной шины ISA перестала удовлетворять потребностям видеосистемы.

Для разгрузки CPU часть рутинной работы по построению изображения возложили на видеоадаптер, снабдив его специализированным вычислительным устройством. Поскольку данное устройство могло работать параллельно с центральным процессором (например, обращаться к системной оперативной памяти, корректировать содержимое кадрового буфера), его назвали *графическим сопроцессором.* В частности, таким сопроцессором оснащен видеоадаптер 8514/A фирмы IBM, обеспечивающий достаточно высокую производительность при разрешении 1024×768. Еще более мощным графическим сопроцессором оснащен

видеоадаптер XGA этой же фирмы. Подобные устройства обеспечивают *аппаратное ускорение* графических функций видеоадаптера, т. е. выполнение этих функций без участия CPU. Основные функции видеоадаптера, требующие аппаратного ускорения, рассмотрены в разд. 2.2.1.

Одновременно с применением аппаратного ускорения графических функций в видеосистеме SVGA стали использоваться новые высокоскоростные шины ввода/вывода (см. разд. 2.2.3), пропускная способность которых многократно превышает пропускную способность шины ISA. Благодаря реализации этих решений удалось повысить быстродействие видеосистемы до уровня, необходимого для работы большинства офисных и ряда мультимедийных приложений.

С применением высокоскоростных шин ввода/вывода и мощных графических сопроцессоров «узким местом» видеоадаптера, ограничивающим его быстродействие, вновь стала видеопамять. Для решения этой проблемы были разработаны новые типы динамической памяти (см. разд. 2.2.2).

Важнейшей проблемой, возникшей после появления видеоадаптеров SVGA, стала выработка единого механизма использования их аппаратных ресурсов. Отсутствие такого механизма на начальной стадии развития адаптеров SVGA породило проблему совместимости программного и аппаратного обеспечения. Основной вклад в решение данной проблемы внесла Ассоциация стандартов по видеоэлектронике (VESA), рекомендации которой позволили разработать *стандарт SVGA (см.* разд. 2.2.4).

И, наконец, разнообразные мультимедийные приложения потребовали от видеоадаптера не только аппаратной поддержки универсальных графических функций, но и решения новых, специфических задач. К ним можно отнести обработку видеоданных на PC, построение трехмерных изображений, прием телевизионных программ и многое другое (см. разд. 2.4).

Таковы основные особенности современного видеоадаптера SVGA. Рассмотрим более подробно те из них, которые в наибольшей степени влияют на производительность видеосистемы мультимедийного PC.

2.2.1. Аппаратное ускорение графических функций

Как уже отмечалось, одним из главных недостатков стандартного адаптера VGA, предопределяющих низкое быстродействие видеосистемы, является его пассивная роль в процессе создания и модификации содержимого кадрового буфера. Это приводит к тому, что для выполнения любых элементарных (низкоуровневых) действий по модификации изображения необходимо задействовать центральный процессор PC в ущерб выполнению других задач. Ярким примером может служить медленная работа PC с процессором 386 или 486 и стандартным адаптером VGA в среде Windows. Практически все ресурсы системы задействуются для создания окон диалога, курсора мыши и т. п. На решение прикладных задач, ради которых и приобретается компьютер, «сил» у него уже не хватает. Не случайно пользователи подобных PC предпочитают по возможности работать в среде DOS — пусть не так красиво и удобно, зато значительно быстрее.

Между тем операции по созданию графического интерфейса в большинстве своем достаточно просты и рутинны. Их необходимо выполнять очень часто. Например, для создания *стандартного графического интерфейса пользователя в среде Windows* (GUI — *Graphic User Interface)* необходимо многократно выполнять в основном простейшие операции по построению *графических примитивов:* линий, прямоугольников, окружностей, а также по их закраске, переносу фрагментов растра (окна) из одной области экрана в другую и т. п. Если эти операции выполнять как последовательность низкоуровневых команд, адресуя каждый пиксел растра (например, с использованием видеосервиса BIOS, а именно, функций 0Ch и 0Dh прерывания INT 10h, см. табл. 2.2), то центральному процессору придется каждый раз выполнять весьма внушительную по размерам подпрограмму, отвлекаясь от решения других задач. В итоге — медленная прорисовка экрана и низкое быстродействие системы в целом.

Ситуация может радикально измениться, если в состав видеоадаптера включить специализированное устройство, предназначенное для выполнения только графических функций. Такому устройству можно выдать одну команду высокого уровня вместо множества низкоуровневых, например, команду построить окружность. При этом достаточно указать геометрические параметры объекта: радиус, цвет, координаты центра. Все промежуточные операции по модификации каждого пиксела, находящегося на окружности, это устройство выполнит самостоятельно, не прибегая к помощи центрального процессора. В этом случае говорят о наличии у видеоадаптера аппаратного ускорения той или иной графической функции, а устройство, обеспечивающее это ускорение, называют *графическим ускорителем (акселератором).*

Таким образом, повышение быстродействия видеосистемы и PC в целом при использовании графического ускорителя достигается за счет:

- аппаратной реализации заданного набора графических функций, выполнение которых осуществляется всего лишь за несколько тактов работы акселератора;
- использования акселератором системы команд высокого уровня, что разгружает шину ввода/вывода — поток данных значительно уменьшается;
- освобождения центрального процессора от необходимости выполнения множества элементарных операций с содержимым кадрового буфера.

Таким образом, использование графического ускорителя является отступлением от классической

архитектуры, где все процессы протекали под управлением центрального процессора, явилось шагом на пути создания *распределенной вычислительной структуры.* Взамен пассивного устройства — графического контроллера — видеоадаптер получил специализированный вычислитель — графический акселератор, который самостоятельно манипулирует содержимым видеопамяти точно так же, как центральный процессор — содержимым оперативной памяти PC.

2.2.1.1. Основные графические функции

Говоря об акселерации графических функций видеоадаптера, необходимо обратить внимание на два обстоятельства:

1. Акселерация необходима только в графическом режиме работы видеоадаптера. Суть действий графического акселератора состоит в изменении цифрового образа растрового изображения в видеопамяти адаптера по командам центрального процессора (напомним, что некоторые элементарные действия по модификации пикселов может выполнять и графический контроллер видеоадаптера VGA, см. разд. 2.1.4).

2. Одно из главных отличий графического акселератора от графического контроллера видеоадаптера VGA заключается в том, что он оперирует не пикселами, а объектами более высокого иерархического уровня, которые называются *графическими примитивами* и состоят из множества пикселов. Примерами графических примитивов являются отрезок прямой, треугольник, прямоугольник, прямоугольник с закругленными краями, многоугольник, дуга, эллипс и т. п. Используя графические примитивы, можно конструировать достаточно сложные изображения значительно быстрее и проще, чем при использовании операций по модификации отдельных пикселов. Поэтому команды для работы с графическими примитивами, которые центральный процессор посылает акселератору, часто называют *высокоуровневыми.*

Для лучшего понимания назначения и структуры графического акселератора кратко рассмотрим основные графические функции, при выполнении которых необходимо использовать аппаратное ускорение.

1. Прорисовка графических примитивов; активно используется при создании элементов графического интерфейса прикладных программ и операционной системы Windows. В *командах прорисовки (Drawing Commands)* параметры этих примитивов задаются в компактном, *векторном виде,* т. е. в виде координат, или векторов узловых точек, по которым легко построить всю фигуру (например, координаты вершин треугольника). Векторное представление графических примитивов значительно удобнее, чем пересылка в видеопамять их полного растрового изображения: во-первых, оно во много раз компактнее растрового и, во-вторых, не зависит от используемого разрешения. При использовании команд прорисовки центральный процессор освобождается от выполнения геометрических расчетов, необходимых для построения упомянутых фигур (например, от решения трансцендентного уравнения при построении эллипса или дуги). К командам прорисовки относятся также команды *заливки (Fill)* заданного контура определенным цветом и *заполнения его заданным узором (Pattern).* Эти команды, чаще всего выполняемые при реализации GUI операционной системы Windows, ускоряются наиболее эффективно, поскольку избавляют центральный процессор от необходимости выполнения огромного количества операций по чтению, анализу и модификации пикселов внутри контура, а также устраняют необходимость пересылки по шине ввода/вывода значений этих пикселов.

2. Перенос блоков изображения из одной части экрана в другую; выполняется при перетаскивании окон, ярлыков и других элементов графического интерфейса операционной системы и прикладных программ, при выполнении прокрутки *(Scrolling)* экрана, при редактировании изображений в графических редакторах, например Adobe Photoshop, Corel Draw! и т. п. Данная функция сводится к перемещению блока бит из одной области видеопамяти в другую, поэтому часто ее обозначают *Bit Blt (Bit Block transferring).* Как и любая другая операция, требующая обработки большого числа пикселов, операция Bit BIt ускоряется особенно эффективно. Размеры исходного и результирующего блоков не обязательно должны совпадать — перенос блока может выполняться с одновременным масштабированием его размеров (операция типа *StretchBIt).*

3. Аппаратная поддержка окон; особенно полезна в многозадачных приложениях с многооконным интерфейсом (и конечно же в Windows!). Суть данной графической функции состоит в следующем. При наличии на экране нескольких раскрытых (особенно перекрывающихся) окон и использовании обычного видеоадаптера VGA каждая прикладная программа тратит довольно много времени на то, чтобы постоянно отслеживать координаты ячеек кадрового буфера, соответствующих области «своего» окна. Естественно, что необходимость выполнения такого контроля значительно замедляет работу программы. Если же используется акселератор с аппаратной поддержкой функции *Hardware Windowing,* то для окна каждой программы в видеопамяти выделяется отдельная область. Таким образом, вместо обычного кадрового буфера, размеры которого определяются разрешением экрана, акселератор использует несколько «кадровых буферов», размеры которых пропорциональны размерам окон соответствующих программ. Очевидно, что при таком подходе программе нет нужды отслеживать координаты «своего» окна, поскольку «чужих» окон в отведенной области видеопамяти просто нет. Помимо использования отдельных областей видеопамяти, акселератор запоминает размер и координаты каждого окна на экране, а также порядок, в котором они накладываются друг на друга. В процессе вывода изображения на экран акселератор сканирует ячейки кадрового буфера не последовательно (как это имеет место в обычном видеоадаптере), а в соответствии с порядком расположения окон, переключаясь при этом с одной области видеопамяти на другую. Заметим, что

выигрыш от использования данной графической функции тем значительнее, чем больше объем видеопамяти, установленной на плате видеоадаптера.

4. Аппаратный курсор. Суть этой графической функции весьма проста. Центральный процессор считывает через COM-порт, к которому подключена мышь, координаты курсора и посылает эту пару чисел в акселератор. Тот, в свою очередь, формирует изображение указателя мыши в заданном месте экрана. При выполнении манипуляций с мышью акселератор самостоятельно переносит изображение курсора в нужное место экрана и ликвидирует «следы» его пребывания на первоначальном месте, внося в кадровый буфер необходимые изменения. Таким образом, центральный процессор освобождается от необходимости корректировать содержимое кадрового буфера видеоадаптера при перемещении указателя мыши; ему достаточно послать акселератору новые координаты курсора.

Аппаратная поддержка курсора основана на использовании акселератором небольших перемещаемых прямоугольных фрагментов изображения, называемых *спрайтами (Sprite* — эльф, гном). Столь странное название объясняется характером их перемещения по экрану. Если операции BitBIt приводят к *перемещению* (т. е. замене) участков растра, то спрайт как бы накладывается на исходное изображение сверху, лишь временно изменяя его. При перемещении спрайта в другое место экрана исходное изображение восстанавливается.

5. Масштабирование растровых изображений; очень распространенная и весьма трудоемкая для центрального процессора процедура, требующая интенсивного обмена данными между CPU и видеопамятью. Такая задача возникает при изменении размеров окна прикладной программы, работе с растровыми изображениями в графических редакторах, полноэкранном просмотре изображений (размер которых отличается от используемого в данный момент разрешения), просмотре видеофильмов на PC и в ряде других случаев. Различают два вида масштабирования: *дублирование (репликация)* и *интерполяция (фильтрация, сглаживание).* Дублирование сводится к простому увеличению размера пиксела в необходимое количество раз. Оно сопровождается появлением характерных дефектов — *пикселизации* и *алиасинга* (зазубренности наклонных линий). Для минимизации этих искажений используется интерполяция. Ее суть заключается в том, что значение каждого пиксела нового, масштабированного изображения определяется как средневзвешенное значений нескольких соседних пикселов. Чем большее количество пикселов обрабатывается при интерполяции, тем лучше результат. Масштабирование по горизонтали и вертикали выполняется раздельно. Высококачественные акселераторы выполняют интерполяцию в обоих направлениях, дешевые — интерполяцию по горизонтали и дублирование по вертикали. Особенно важна качественная и быстрая интерполяция при воспроизведении видеоинформации на PC.

6. Панорамирование; если объем видеопамяти превышает размер кадрового буфера, необходимый при данном разрешении и выбранной палитре, то прикладная программа может синтезировать изображение, размер которого (в пикселах) будет превышать размер видимой области экрана. Важно отметить, что фактическое разрешение экрана (т. е. число пикселов на единицу длины) при этом не меняется. Такая ситуация возникает, например, при использовании плоскопанельных ЖК-мониторов, рассчитанных на работу с фиксированным разрешением. В этом случае для просмотра невидимой в данный момент области экрана используется функция *панорамирования (Panning),* похожая на упомянутую выше прокрутку экрана. Однако панорамирование не требует выполнения операций типа BitBIt, а сводится лишь к изменению начального адреса той области видеопамяти, в которой записана видимая в данный момент часть изображения.

7. Преобразование цветового пространства; используется главным образом в мультимедийных приложениях, связанных с обработкой видеоинформации.

Как известно, видеоадаптер SVGA в режимах High Color (16 бит на пиксел) и True Color (24 бит на пиксел) хранит изображение в кадровом буфере и передает его в RAMDAC в формате RGB. Однако для хранения изображения в формате RGB требуется большой объем памяти. Поэтому в большинстве приложений, связанных с обработкой и хранением динамических изображений, использование данного формата нецелесообразно. В частности, в телевидении, цифровом видео и других областях широко используются более экономичные формы представления, основанные на замене трех сигналов основных цветов сигналами яркости и цветности. В настоящее время наиболее широко распространен формат YUV, первоначально появившийся в системе цветного телевидения PAL (часто вместо термина «формат YUV» употребляют термин «цветовое пространство YUV»). Этот формат подразумевает замену сигналов *R, G, B* сигналом яркости *Y* и двумя цветоразностными сигналами *U и V.* Поскольку сигналы *U и V* имеют гораздо более узкий спектр по сравнению с сигналом яркости Y и сигналами *R, G, B,* преобразование изображения в цифровую форму в пространстве YUV обеспечивает значительное сжатие информации. Поэтому формат YUV является основным для большинства носителей цифровой видеоинформации, используемых в мультимедийных приложениях. Для воспроизведения цифрового изображения в формате YUV на экране монитора PC необходимо преобразовать это изображение в формат RGB. В частности, такая задача возникает при воспроизведении видеоданных, сжатых по методу MPEG. Если возложить задачу преобразования цветового пространства на центральный процессор, то на решение других задач (например, декомпрессию кадров) у него не останется времени, в результате чего скорость воспроизведения (частота кадров) будет недостаточной для создания иллюзии непрерывности движения. Высококачественный графический ускоритель самостоятельно в реальном масштабе времени выполняет это преобразование.

8. Декомпрессия сжатых видеоданных; цифровые видеофильмы хранятся в упакованном, сжатом формате (в противном случае они не поместятся ни на один носитель). В качестве метода сжатия обычно используют MPEG. При воспроизведении необходимо распаковать каждый кадр изображения перед его записью в кадровый буфер видеоадаптера. Эта распаковка, или *декомпрессия,* может выполняться либо программно (центральным процессором), либо аппаратно (графическим ускорителем). Первый вариант возможен только на PC с процессором Pentium-100 и выше. При наличии аппаратной декомпрессии цифровое видео можно воспроизводить даже на компьютере с процессором 386.

9. Построение трехмерных изображений; перечень функций по построению трехмерных изображений весьма обширен и является предметом отдельного разговора. Аппаратные средства ускорения операций с трехмерной графикой (3D) подробно будут рассмотрены в гл. 3.

Итак, мы кратко рассмотрели наиболее важные и распространенные графические функции. В отдельных графических акселераторах в зависимости от их назначения набор реализуемых функций может отличаться от приведенного. В частности, у некоторых современных видеоадаптеров предусмотрена возможность программно-аппаратной поддержки формата DVD, вывода информации на экран обычного телевизора (функция TV-out) и т. п.

Первым массовым графическим акселератором стал так называемый акселератор Windows, в котором на аппаратном уровне были реализованы функции 1—6. Необходимость воспроизведения видеоинформации на PC стала следующим шагом в развитии графических акселераторов, что потребовало аппаратной поддержки функций 5, 7 и, по возможности, 8. И, наконец, наступление эры 3D послужило толчком к разработке средств для реализации на аппаратном уровне функции 9.

Типовой современный графический ускоритель для обработки двумерных графических данных (2D) обычно реализует аппаратное ускорение функций 1—7. С одной стороны, это обеспечивает хорошую производительность в традиционных офисных приложениях типа Microsoft Word, Microsoft Excel и т. п. С другой стороны, аппаратная поддержка масштабирования и преобразования цветового пространства позволяет эффективно воспроизводить видеоданные программным путем на PC средней производительности.

Что же представляет собой видеоадаптер с аппаратным ускорением графических функций с точки зрения его структуры? В зависимости от конкретного способа реализации аппаратной поддержки графических функций и степени «интеллектуальности» видеоадаптера эти устройства первоначально делились на две группы:

- видеоадаптеры с графическим ускорителем (акселератором);
- видеоадаптеры с графическим сопроцессором.

Более универсальным устройством, безусловно, является графический сопроцессор, работающий параллельно с CPU (отсюда и происходит его название, по аналогии с математическим сопроцессором). В современных видеоадаптерах объем и сложность графических функций, выполняемых графическим сопроцессором, возросли до такой степени (особенно в области 3D), что стали сопоставимы с объемом задач, выполняемых центральным процессором PC. По этой причине подчинительная приставка «со» утратила свой первоначальный смысл, и Chipset, составляющий основу современного видеоадаптера с аппаратной поддержкой практически всех упомянутых графических функций, стали называть *графическим процессором,* подчеркивая тем самым его самостоятельность в решении задач формирования изображения на экране монитора.

Однако в рамках данной главы, посвященной принципам построения видеоадаптера и не затрагивающей область 3D, мы будем пользоваться первыми двумя терминами (графический акселератор и графический сопроцессор). Рассмотрим главные различия между этими двумя устройствами с точки зрения их места и роли в составе видеоадаптера и архитектуры PC в целом.

2.2.1.2. Видеоадаптер с графическим акселератором

Графический акселератор является по сути основательно усовершенствованным вариантом графического контроллера видеоадаптера VGA. Это значит, что, освобождая центральный процессор от выполнения многих графических функций, он не изменяет общего характера взаимодействия между CPU и видеоадаптером. Другими словами, акселератор обладает определенной самостоятельностью только в пределах видеоадаптера (при работе с видеопамятью), но не в рамках общей архитектуры PC. В частности, акселератор не может самостоятельно через шину ввода/вывода обращаться к системной оперативной памяти так, как это делает центральный процессор. Не случайно первые акселераторы появились еще в составе видеоадаптеров с интерфейсом ISA (как известно, шина ISA не предусматривает активной роли подключаемых к ней устройств).

Графический акселератор представляет собой устройство комбинационного типа, выполняющее заданные логические или арифметические операции по жесткому алгоритму. Поскольку алгоритм работы определен заранее и не меняется, для выполнения акселератором графических функций достаточно подать соответствующую команду. Типовая структура видеоадаптера с графическим акселератором, выполняющим функции 1—6, представлена на рис. 2.17.

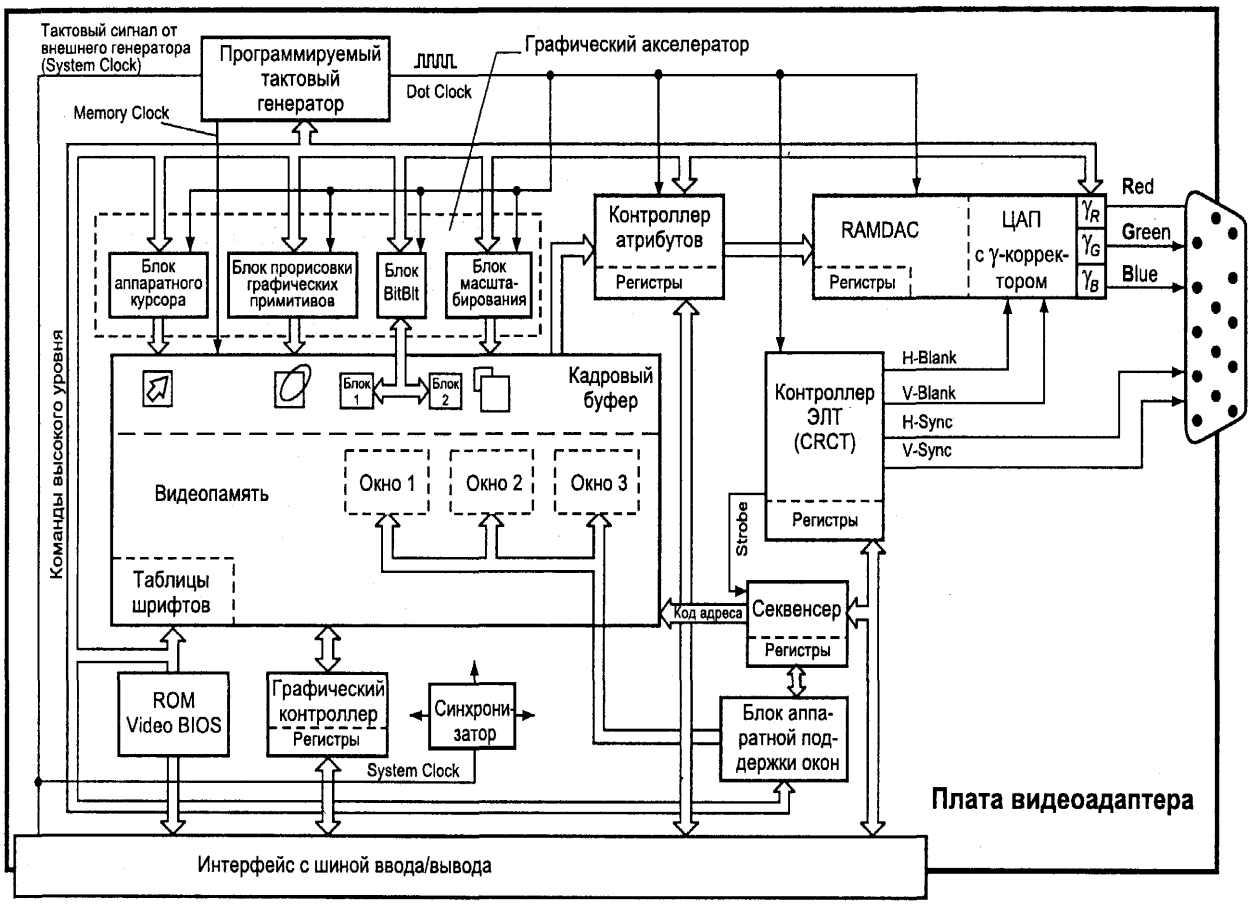


Рис. 2.17. Типовая структура видеоадаптера с графическим акселератором

Основное отличие видеоадаптера с графическим акселератором от стандартного видеоадаптера VGA состоит в том, что вместо простейшего графического контроллера в нем используется специализированное вычислительное устройство со своей системой команд. Центральный процессор под управлением прикладной программы выдает дешифратору команды, которые затем обрабатываются соответствующим блоком акселератора. Заметим, что графический контроллер видеоадаптера VGA не имел в явном виде собственной системы команд, поэтому большая часть операций с содержимым кадрового буфера сводилась к выполнению стандартных команд перемещения данных между регистрами процессора и видеоадаптера (команды типа mov, in, out и т. п.).

Собственно акселератор состоит из нескольких узкоспециализированных блоков (узлов), обеспечивающих аппаратную реализацию соответствующей графической функции. Результатом работы акселератора является модифицированное должным образом содержимое кадрового буфера. Дальнейшая обработка хранящихся в нем данных (считывание, передача на контроллер атрибутов и RAMDAC) происходит подобно тому, как это делается в стандартном видеоадаптере VGA. При поддержке функции Hardware Windowing секвенсер может изменять обычный порядок сканирования видеопамяти.

2.2.1.3. Видеоадаптер с графическим сопроцессором

В принципе, графический сопроцессор решает те же задачи, что и акселератор, но по-иному. Различия проявляются как в структуре этих устройств (рис.2.18), так и в способе их взаимодействия с центральным процессором.

Во-первых, графический сопроцессор — это интеллектуальное устройство, основу которого составляет арифметико-логическое устройство (АЛУ). АЛУ фактически представляет собой специализированный микропроцессор, работающий по программам, содержащимся в собственном ОЗУ микроинструкций. Если в акселераторе графические функции (масштабирование, аппаратный курсор и др.) выполняются различными специализированными устройствами по жесткому алгоритму, то программируемый графический сопроцессор позволяет решать большинство задач при помощи одного элемента — АЛУ. Таким образом, основное отличие графического сопроцессора от графического акселератора заключается в том, что сопроцессор можно запрограммировать на выполнение различных задач, тогда как ускоритель работает по жесткому, неизменному алгоритму.

Во-вторых, графический сопроцессор, в отличие от графического акселератора, является активным устройством, сфера деятельности которого не ограничивается только видеопамятью. В процессе выполнения

своих функций он может наравне с центральным процессором обращаться к системной оперативной памяти и брать на себя управление шиной ввода/вывода. В частности, такую возможность предоставляет архитектура шины PCI, поддерживающая режим управления шиной *Bus Mustering.* В этом режиме видеоадаптер с графическим сопроцессором становится активным, или ведущим устройством *(Master),* задающим на шине код адреса и код операции (напомним, что при использовании шины ISA такие действия мог выполнять только центральный процессор). Наиболее ярко активная роль интеллектуального видеоадаптера проявляется при использовании новой шины AGP в режиме DiME (см. разд. 3.2.4), в котором необходимые преобразования изображения выполняются не в кадровом буфере, а в системной оперативной памяти.

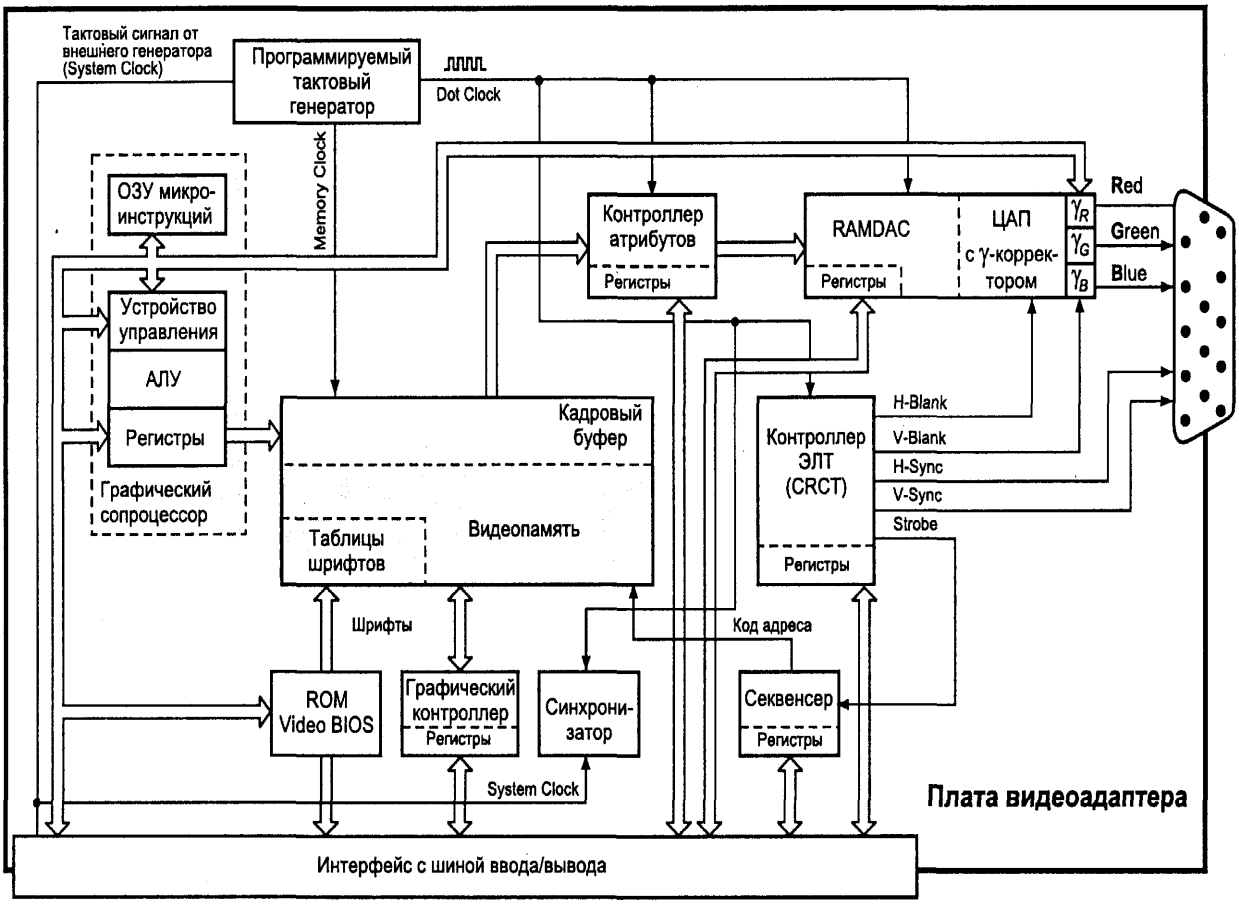


Рис. 2.18. Типовая структура видеоадаптера с графическим сопроцессором

2.2.2. Память для видеоадаптеров

Видеопамять адаптера VGA используется двумя устройствами: центральным процессором, который записывает в нее информацию, и видеоконтроллером, который считывает содержимое кадрового буфера и передает его в RAMDAC. Следовательно, с точки зрения быстродействия видеоадаптера, важна скорость, с которой выполняются операции чтения/записи данных из видеопамяти, а также возможность одновременной работы с видеопамятью центрального процессора PC и видеоконтроллера.

В качестве видеопамяти в стандартном VGA- и первых моделях SVGA-адаптеров использовалась память типа DRAM *(Dynamic Random Access Memory* — динамическая память с произвольным доступом).

Ячейка динамической памяти построена на основе конденсатора (рис. 2.19), хранящего электрический заряд.

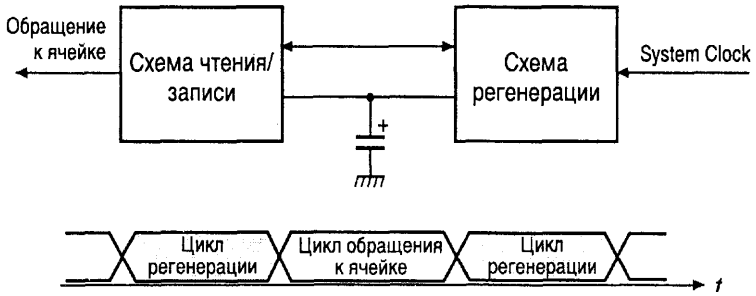


Рис. 2.19. Ячейка динамического ОЗУ

Если конденсатор заряжен, то считается, что в ячейку записана логическая единица, если разряжен — логический ноль. Поскольку емкость конденсатора очень мала (доли пикофарады) и имеется ток утечки, информация в такой ячейке может храниться только в течение короткого промежутка времени, а при считывании данных конденсатор разряжается чуть ли не мгновенно.

Для длительного хранения информации в такой ячейке необходимо периодически, а при считывании — каждый раз проводить *регенерацию* ее содержимого. Регенерация осуществляется путем выполнения (примерно через каждые 10 мс) операций считывания/записи всех хранящихся в памяти данных.

Быстродействие микросхемы динамической памяти характеризуется временем, необходимым для выполнения операции считывания или записи данных. Последовательность этих операций называют *рабочим циклом* или *циклом обращения* памяти.

Время, необходимое для чтения (записи) данных, хранящихся по случайному адресу, называют временем доступа *(Access time).* Для микросхем видеопамяти типа DRAM оно составляло 80—100 нс.

В реальных условиях обращение к памяти чаще происходит не по случайному адресу, поэтому время рабочего цикла всегда меньше времени доступа.

Необходимость в регенерации отсутствует у *статической* памяти (SRAM), ячейка которой представляет собой триггер (рис. 2.20). Быстродействие такой памяти оказывается в несколько раз выше (среднее время доступа 15-20 нс), чем DRAM. Однако и стоимость ее изготовления также значительно выше. Поэтому статическую память обычно используют в качестве скоростного буфера небольшой емкости — *кэш-памяти,* — в котором хранятся наиболее часто используемые данные.

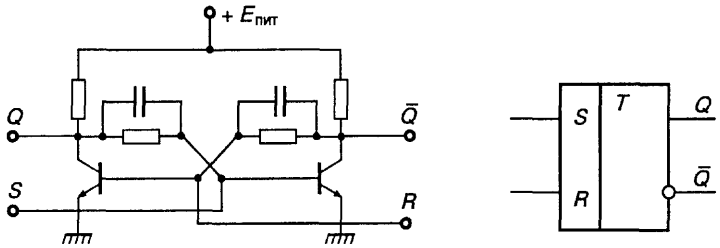


Рис. 2.20. Ячейка статического ОЗУ

Для повышения скорости обмена данными с динамической памятью используется так называемый *пакетный режим.* В этом режиме процессор считывает данные из памяти не побайтно, а в виде пакетов *(burst),* или слов, состоящих, в зависимости от разрядности шины данных, из 32 или 64 бит. Далее используется вероятностный подход, заключающийся в следующем. Допустим, необходимо считать слово по указанному адресу Вероятность того, что возникнет необходимость считать еще несколько слов, расположенных в соседних ячейках памяти, довольно высока. Поэтому, чтобы многократно не повторять операцию чтения данных, процессор дополнительно считывает еще три следующих слова. Время считывания каждого слова измеряется в тактах процессора и записывается в виде *временной диаграммы.* Например, временная диаграмма 7-3-3-3 означает, что на считывание первого слова необходимо 7 тактов процессора, а на считывание каждого последующего — только 3.

Элементы памяти DRAM имеют матричную организацию, причем каждый элемент матрицы (конденсатор) хранит один бит информации. Адресация ячеек памяти осуществляется путем указания кода адреса столбца и строки, на пересечении которых находится требуемая ячейка, а также с помощью специальных сигналов RAS *(Row Address Strobe* — строб адреса строки) и CAS *(Column Address Strobe* — строб адреса столбца), сопровождающих эти коды.

В видеоадаптерах используются однопортовые и двухпортовые элементы динамической памяти.

2.2.2.1. Однопортовая видеопамять

Однопортовая память — это разновидность динамической памяти, имеющей только один порт, через который осуществляется чтение/запись данных. Таким образом, одновременное выполнение операций чтения и записи невозможно. Это сильно ограничивает быстродействие видеоадаптера, так как процессор может начать обновление кадрового буфера лишь после того, как видеоконтроллер завершит его чтение.

2.2.2.1.1. FPM DRAM

FPM DRAM *(Fast Page Mode DRAM)* — память DRAM с быстрым страничным обменом. Память этого типа использовалась в видеоадаптерах с графическим ускорителем и получила широкое распространение. Память FPM DRAM разбита на участки, именуемые страницами. В пределах страницы полный адрес ячейки памяти состоит из двух частей: адреса строки и адреса столбца. Размер страницы может составлять от 512 байт до нескольких килобайт. При считывании данных, расположенных последовательно в пределах одной строки, отпадает необходимость в повторном формировании сигнала RAS, благодаря чему удается значительно ускорить чтение данных. На корпусе микросхем FPM DRAM указывается среднее время доступа к произвольно расположенным данным. Однако реальное время рабочего цикла на 30% меньше.

Например, для FPM DRAM с временем доступа 60 нс длительность рабочего цикла составляет 40 нс.

2.2.2.1.2. EDO DRAM

EDO DRAM *(Extended Data Output DRAM)* — память DRAM с увеличенным временем удержания данных на линиях ввода/вывода. В отличие от FPM DRAM, в которой линии ввода/вывода отключаются от шины данных при задании адреса столбца следующей ячейки, в EDO DRAM данные доступны для считывания во время передачи нового адреса. Таким образом, при работе с памятью EDO DRAM возможно одновременное считывание данных и задание адреса следующих, что позволяет повысить быстродействие при чтении на 10—15% по сравнению с быстродействием FPM DRAM и на 30% — по сравнению с DRAM.

Также как и в случае использования FPM DRAM, выигрыш в быстродействии имеет место лишь при обращении к данным, расположенным последовательно. Стоимость микросхем EDO DRAM примерно такая же, как и микросхем DRAM. Причем для использования их в качестве видеопамяти достаточно внести лишь небольшие изменения в алгоритм работы видеоконтроллера. На корпусе микросхем EDO DRAM указывается также среднее время доступа для произвольно расположенных данных.

2.2.2.1.3. BEDO DRAM

BEDO DRAM *(Burst Extended Data Output DRAM)* — пакетная память EDO DRAM. В микросхему BEDO DRAM добавлен специальный счетчик слов, благодаря чему считывание из памяти четырех соседних слов осуществляется автоматически, даже тогда, когда необходимо считать лишь одно из них. На частоте системной шины PC 66 МГц временная диаграмма BEDO DRAM имеет вид 7-1-1-1, в то время как для FPM — 7-3-3-3, а для EDO— 7-2-2-2.

2.2.2.1.4. SDRAM

Низкое быстродействие видеопамяти всегда ограничивало быстродействие видеосистемы и PC в целом. Появились процессоры с более высокой тактовой частотой и новые высокоскоростные шины. Видеопамять же оставалась прежней. До недавнего времени (всего лишь несколько лет назад) лучшей дешевой памятью считалась FPM DRAM с временем доступа 60—70 нс (в профессиональных видеоадаптерах использовалась двухпортовая память типа VRAM, однако стоимость таких адаптеров чуть ли не в 2—3 раза превышала стоимость адаптеров с памятью FPM DRAM).

Для организации согласованной работы процессора и памяти был введен параметр *Wait State* (цикл ожидания). Этот параметр указывает, сколько тактов должен пропустить процессор, пока на выходных линиях памяти гарантированно не появятся запрошенные данные. Необходимость введения этого параметра была обусловлена еще и тем, что все элементы памяти (в том числе и рассмотренные выше) в то время были асинхронными, т. е. данные на их выходах появлялись спустя некоторое время *(время задержки)* с момента изменения сигналов на информационных входах. Это время, как правило, не регламентируется и может изменяться в зависимости от колебаний температуры и по мере старения полупроводниковых элементов.

Таким образом, получается, что, пока данные не считались, процессор простаивает и не может заниматься выполнением других задач. Дополнительные циклы ожидания могут значительно снизить производительность системы в целом. В результате складывается парадоксальная ситуация, когда уже не имеет смысла повышать тактовую частоту процессора и пропускную способность шины без кардинального решения проблемы быстродействия памяти.

Частично решить эту проблему удалось путем создания *синхронной памяти* SDRAM *(Synchronous DRAM).* Принцип работы SDRAM такой же, как и DRAM, за исключением того, что для работы SDRAM необходим дополнительный тактирующий сигнал, частота которого кратна тактовой частоте процессора. Этим сигналом фиксируются моменты обновления информации на входах и выходах элементов памяти, т. е. процессор в зависимости от типа используемой синхронной памяти будет точно «знать», в какой момент времени данные появятся на ее выходных линиях. В этом случае процессор (видеоконтроллер) и память работают синхронно, без циклов ожидания, т. е. все операции по управлению памятью синхронизированы с тактовой частотой процессора. Теперь вместо циклов ожидания процессор может выполнять другие задачи.

Синхронная работа процессора и памяти позволила исключить ситуацию, когда команды чтения/записи совпадают по времени с процедурой регенерации памяти, а также уменьшить время выборки данных за счет реализации трехступенчатой конвейерной адресации, позволяющей осуществить доступ к запрошенным данным до завершения обработки предыдущих. Благодаря этому удалось уменьшить длительность рабочего цикла до 7-10 нс и поднять частоту тактового сигнала до 125 МГц и выше.

Память SDRAM широко используется в современных ускорителях трехмерной графики. На корпусе микросхем памяти SDRAM указывается не время доступа к случайно расположенной ячейке памяти (как для FPM и EDO), а время рабочего цикла памяти с учетом чередования адресов, пакетного режима и трехступенчатой конвейерной адресации. Современные видеоадаптеры оснащаются памятью SDRAM с временем доступа не более 10 нс.

2.2.2.1.5. SGRAM

Память типа SGRAM *(Synchronous Graphics RAM* — синхронная графическая память) имеет много

общего с памятью типа SDRAM. Память SGRAM может работать на частотах 66 МГц и более. По сравнению с DRAM, быстродействие SGRAM возросло более чем в четыре раза. Память SGRAM предназначена для хранения изображений, т. е. для использования в видеоадаптерах.

2.2.2.1.6. MDRAM

MDRAM *(Multibank DRAM)* — мультибанковая динамическая память, разработанная фирмой MoSys Inc. и предназначенная для использования исключительно в системах обработки графики и видео изображений. Главной особенностью памяти MDRAM является ее необычная организация. MDRAM представляет собой набор независимых банков обычной динамической памяти; емкость каждого составляет 32 Кбайт.

Банком памяти называется совокупность микросхем памяти, суммарная разрядность которых равна разрядности шины данных. Например, для организации памяти из 8-разрядных микросхем при 32-разрядной шине данных каждый банк должен содержать 4 микросхемы памяти одинаковой емкости. Обычно используются два банка памяти, чтобы реализовать режим чередования адресов и циклов регенерации.

В результате использования одно- или двухбанковой архитектуры память видеоадаптера может иметь только конкретный объем: 1, 2, 4, 8 или 16 Мбайт. В этом случае видеопамять используется неэффективно. Например, в режиме 1024×768 и 16,7 млн оттенков цветов кадровый буфер должен иметь размер 2,25 Мбайт; установить же придется 4 Мбайт видеопамяти; 1,75 Мбайт использоваться не будут.

Независимость банков памяти MDRAM позволила создать микросхемы памяти различного номинала и, как следствие, исключить избыточность объема, характерную для других традиционных типов памяти. В частности, фирмой MoSys выпускаются модули памяти емкостью 0,5; 0,75; 1; 1,125 и 1,25 Мбайт. Кроме того, MDRAM можно организовать таким образом, что обращение к каждому банку будет независимым. А это — путь к несомненному повышению производительности видеоадаптера в целом, по сравнению с моделями, использующими двухпортовую память—VRAM или WRAM. Память типа MDRAM используется пока только в видеоадаптерах фирмы Tseng Labs (ET6000, ET6100).

2.2.2.2. Двухпортовая видеопамять

В отличие от однопортовой видеопамяти, ограничивающей быстродействие видеоадаптера, использование *двухпортовой* видеопамяти позволяет осуществлять одновременное выполнение операций чтения и записи данных, т. е. к ней одновременно могут обращаться два устройства, например графический процессор и RAMDAC. Двухпортовая видеопамять, хотя и стоит дороже однопортовой, но обеспечивает значительное ускорение работы видеоадаптера, особенно в режимах с высокими разрешением и глубиной цвета. В настоящее время используются две разновидности двухпортовой памяти — VRAM и WRAM.

2.2.2.2.1. VRAM

VRAM *(Video RAM)* — это динамическая память с двумя портами для одновременного выполнения операций чтения и записи. Для работы с такой памятью на плате видеоадаптера устанавливается специальный контроллер. Как следует из ее названия, VRAM была разработана исключительно для видеосистемы PC. Преимущества VRAM особенно заметны при работе в режимах с высоким разрешением (1280×1024 и выше) и 24-битным представлением цвета. Поэтому элементы памяти этого типа используются в основном в профессиональных графических системах, для которых высокая цена не является серьезным недостатком.

2.2.2.2.2. WRAM

Архитектура WRAM *(Window RAM)* очень похожа на архитектуру VRAM, но стоимость изготовления WRAM ниже. Эта память может работать на частоте до 50 МГц, благодаря чему быстродействие повышается на 50% по сравнению с VRAM.

Технология изготовления и схемотехника памяти типа WRAM ориентированы на конкретное применение в видеоадаптерах. В этой памяти имеется режим быстрой записи двухцветных блоков для ускорения пересылки текста и закраски областей пиксельными шаблонами *(Dual-Color Block Write),* а также реализована технология двойной буферизации *FastBit.* В соответствии с технологией FastBit, для двух смежных кадров предусмотрены два кадровых буфера. Пока первый кадр отображается (считывается в RAMDAC), CPU формирует следующий. В итоге обеспечивается быстрая смена кадров, что особенно важно при обработке видео и работе с трехмерной графикой.

Все перечисленные типы видеопамяти считаются традиционными, поскольку широко применяются на практике. Однако в настоящее время появились новые перспективные типы памяти, которые пока не нашли широкого применения, но за ними — будущее. Кратко рассмотрим их.

2.2.2.3. Перспективные типы видеопамяти

2.2.2.3.1. Direct RDRAM

Direct RDRAM *(Direct Rambus DRAM)* — динамическая память с прямым доступом, производства американской фирмы Rambus, на сегодня — одна из самых быстродействующих. В настоящее время память этого типа выпускают несколько компаний, в том числе NEC, Fujitsu и Toshiba.

В микросхемах Direct RDRAM сохранились старые принципы записи/чтения данных. Изменения затронули лишь организацию доступа к ячейкам памяти. Технология Rambus предусматривает наличие 16-разрядной высокоскоростной шины Rambus Channel, тактовая частота которой составляет 400 МГц, а скорость передачи данных — 1,6 Гбайт/с.

2.2.2.3.2. 3D RAM

3D RAM — память для ускорителей трехмерной графики. Разработана фирмой Mitsubishi для совместного использования с Chipset 3Dpro ее же производства. Этот Chipset основан на использовании передовой технологии REALImage фирмы Evans&Sutherland и предназначен для работы с трехмерной графикой. Память 3D RAM является двухпортовой и имеет разветвленную архитектуру, позволяющую осуществлять конвейерную (параллельную) обработку данных. Кроме того, предусмотрено наличие встроенного процессора и скоростных видеобуферов.

2.2.2.3.3. CDRAM

CDRAM *(Cashed DRAM)* — кэшированная динамическая память; также является разработкой фирмы Mitsubishi. Представляет собой комбинацию динамической памяти и небольшого скоростного буфера (с временем доступа 10-15 нс), выполненного на элементах статической памяти, традиционно используемой в качестве кэш-памяти. В частности, имеются модули памяти CDRAM, содержащие 4 или 16 Мбит динамической памяти и 16 Кбит статической. Подобно 3D RAM, CDRAM предназначена для использования в 3D-ускорителях (в качестве памяти для текстур). Кроме того, CDRAM используется в качестве кэш-памяти в скоростных винчестерах с интерфейсом Fibre Channel. CDRAM имеет 128-разрядную внутреннюю шину данных и 16-разрядную внешнюю, благодаря чему данные считываются из памяти со скоростью около 800 Мбайт/с.

2.2.2.3.4. DDR SDRAM

DDR SDRAM *(Double Data Rate SDRAM)* — память SDRAM с удвоенной скоростью передачи данных. Другое название этого типа памяти — SDRAM II. Разработка компании Samsung. Основное отличие этой памяти от ее прототипа SDRAM заключается в том, что чтение данных осуществляется как по фронту, так и по спаду тактового сигнала, что позволяет выполнять два обращения к памяти за время одного цикла обращения к «стандартной» SDRAM.

2.2.2.3.5. ESDRAM

ESDRAM *(Enhanced SDRAM)* — улучшенная память SDRAM. Разработка фирмы Enhanced Memory Systems. Более высокого быстродействия элементов памяти ESDRAM, по сравнению с SDRAM и даже DDR SDRAM, удалось добиться благодаря использованию более совершенной архитектуры и встроенной статической (SRAM) кэш-памяти (подобно тому, как это реализовано в элементах памяти CDRAM). Кроме того, за счет реализации алгоритма «упреждающего чтения» из ячеек SDRAM в более быстрые регистры SRAM, по аналогии с технологией кэширования жестких дисков, значительно повышена скорость считывания информации.

В табл. 2.12 приведены основные характеристики наиболее распространенных типов видеопамяти. Пропускная способность указана для 64-разрядной шины данных.

Таблица 2.12. Основные характеристики видеопамяти различных типов

Тип памяти	Тактовая частота шины данных, МГц	Пропускная способность, Мбайт/с
FPM DRAM	25-33	80
EDO DRAM	40-50	105
VRAM	25-33	120
WRAM	50	180
SDRAM	66-100	166-253
RDRAM	250	300-500
MDRAM	125-166	405-490

2.2.3. Интерфейс между видеоадаптером и материнской платой

Еще одним фактором, сдерживающим производительность видеосистемы и PC в целом, является ограниченная пропускная способность шины ввода/вывода, по которой осуществляется обмен данными между центральным процессором и видеоадаптером. Если пропускная способность шины окажется меньше, чем требуется, ни один из рассмотренных выше способов повышения производительности видеосистемы не даст положительного результата.

Определение шины ввода/вывода было дано в разд. 1.1 при рассмотрении особенностей архитектуры

PC. Помимо чисто конструктивных особенностей (количество проводов, форма контактов, параметры сигналов и т. п.), важную роль в организации обмена данными между периферийными устройствами (например, видеоадаптером) и центральным процессором играет набор правил, или алгоритм обмена, называемый *протоколом.* Совокупность конструктивных особенностей шины и протокола обмена принято называть *интерфейсом (Interface).*

В процессе эволюции IBM PC для нужд видеосистемы использовалось несколько разновидностей шин ввода/вывода, однако широкое распространение получили только четыре из них: ISA, VLB, PCI и AGP. Кратко рассмотрим основные особенности этих шин ввода/вывода.

2.2.3.1. Шина ISA

Видеосистема VGA получила широкое распространение в PC с шиной ISA *(Industrial Standard Architecture* — промышленная стандартная архитектура). Эта шина долгое время была стандартной шиной ввода/вывода, ее слоты расширения имеются на материнских платах и по сей день.

Эта 16-разрядная шина работает на тактовой частоте 8 МГц и впервые была реализована на материнских платах с процессором Intel 80286 (напомним, что разрядность шины данных этого процессора равна 16, а тактовая частота для его первых моделей — 8 МГц). Теоретическая пропускная способность шины ISA составляет 16×8000000 = 122 Мбит/с = 15,2 Мбайт/с. Однако, поскольку далеко не каждый такт сопровождается передачей данных, реальная пропускная способность шины ISA не превышает 3 - 8 Мбайт/с. Этого вполне достаточно для работы сравнительно медленных устройств типа контроллеров портов ввода/вывода, накопителей на гибких дисках, модемов и др. Однако уже для видеоадаптера VGA возможности шины ISA были на пределе. Например, в видеорежиме 12h (640×480/16) частота кадров составляет 60 Гц. Для реализации этого видеорежима необходимо обеспечить поток данных между центральным процессором и видеоадаптером, равный 640×480×4×60=8,8 Мбайт/с. Это значение немного превышает реальную пропускную способность шины. Если же установить разрешение 800×600 и/или 8-битное кодирование цвета, то шина ISA и вовсе перестанет справляться с поставленной задачей. В результате — медленная прорисовка экрана, утомительное ожидание окончания заданной операции.

Таким образом, возможности шины ISA перестали удовлетворять даже скромным требованиям офисных приложений, а для мультимедийных оказались просто непригодными. В частности, для воспроизведения видеофрагмента с удовлетворительным качеством (с разрешением 640×480, 65536 оттенками цветов и частотой кадров не ниже 25—30 кадр/с) необходимо обеспечить поток данных между центральным процессором и видеоадаптером около 17,8 Мбайт/с. При переходе в режим True Color (24 бита на пиксел) с разрешением 800×600 потребуется уже 41,2 Мбайт/с.

Вывод очевиден: шина ISA не в состоянии обеспечить необходимую для нормальной работы видеосистемы и PC в целом пропускную способность. Кстати отметим одну интересно деталь: сотрудники фирмы IBM, разработавшей видеосистему VGA, прекрасно понимали, что возможности шины ISA явно не соответствуют требованиям, предъявляемым к ней видеоадаптером. Поэтому первым компьютером, в котором была использована видеосистема VGA, стал IBM PS/2 с процессором Intel 80386, а не IPM PC (помимо нового процессора, IBM PS/2 был оснащен высокоскоростной 32-разрядной шиной ввода/вывода MCA).

2.2.3.2. Локальная шина VLB

Создание Ассоциацией стандартов по видеоэлектронике (VESA) локальной шины VLB *(VESA Local Bus)* в свое время оказалось наиболее удачным и недорогим решением проблемы повышения пропускной способности шины ввода/вывода.

Локальная шина VESA представляет собой не новое устройство на материнской плате, а скорее расширение шины ISA для обмена видеоданными.

Обмен информацией осуществляется под управлением контроллеров, расположенных на картах, устанавливаемых в слот VLB, напрямую с CPU, в обход стандартной шины ввода/вывода. Шина VLB является 32-разрядной и работает на тактовой частоте процессора. Кроме того, передача данных по этой шине невозможна без использования линий шины ISA, по которым передаются сигналы адресов и управления.

Согласно спецификации VESA, тактовая частота локальной шины не должна превышать 40 МГц. Однако для большинства материнских плат, имеющих процессор с тактовой частотой 50 МГц, проблем обычно не возникало.

С другой стороны, это ограничивало максимальное число подключаемых к шине устройств, так как выводы процессора имеют весьма ограниченную нагрузочную способность. В результате на материнской плате можно установить не более трех слотов VLB. Реально же, как показывает практика, устойчиво работают не более двух устройств. Кроме того, подключение устройств к шине VLB заметно ужесточает тепловой режим работы центрального процессора, так как увеличиваются протекающие через него токи. Неслучайно первые вентиляторы на центральных процессорах типа 486 появились именно на тех материнских платах, которые были оснащены шиной VLB.

В целом же, разработка локальной шины VLB и видеоадаптеров SVGA с графическим ускорителем и

соответствующим интерфейсом явилась важным этапом на пути увеличения производительности видеосистемы. В частности, такими видеоадаптерами комплектовалось большинство компьютеров с процессором 486DX2.

2.2.3.3. Шина PCI

Едва шина VLB успела закрепиться на рынке, как уже появилась новая шина — PCI *(Peripheral Component Interconnect).* Она была разработана фирмой Intel для нового высокопроизводительного процессора Pentium. Шина PCI, в отличие от VLB, является не дальнейшим развитием шины ISA, а шиной совершенно нового типа.

На современных материнских платах тактовая частота шины PCI (спецификация PCI 1.0) задается как 1/2 тактовой частоты системной шины, т. е. при тактовой частоте системной шины 66 МГц тактовая частота шины PCI будет 33 МГц. Поэтому ее пропускная способность составила 132 Мбайт/с (столько же, сколько у шины VLB). Однако с появлением новых версий спецификации PCI (PCI 2.0) тактовая частота была увеличена до 66 МГц, т. е. пропускная способность шины увеличилась вдвое. В настоящее время завершается работа над новой версией спецификации PCI, в соответствии с которой тактовая частота шины составит 100 МГц.

Важной особенностью шины PCI является реализация принципа Bus Mastering — возможность управления шиной (или пересылкой данных) внешним устройством без участия центрального процессора. Во время передачи информации устройство, поддерживающее технологию Bus Mastering, захватывает шину и становится главным. Такой подход позволяет освободить центральный процессор от управления передачей данных для выполнения других задач.

Еще одним свойством шины PCI является ее интеллектуальность, т. е. она в состоянии распознавать аппаратные средства и осуществлять анализ конфигурации системы в соответствии с технологией Plug&Play, разработанной корпорацией Intel.

Разрабатывавшаяся первоначально как локальная, шина PCI быстро завоевала широкую популярность и де-факто стала основной шиной PC с процессорами Pentium. Ведущие производители оборудования для PC подчеркивают, что шина PCI была и останется на ближайшее будущее стандартной шиной ввода/ вывода PC.

Несмотря на все преимущества шины PCI, ее возможностей становится недостаточно в условиях растущей нагрузки на систему. Причина заключается в комбинации 3D-графики и видео, с которыми приходится иметь дело новому поколению графических адаптеров.

Как известно, построение трехмерных изображений требует больших объемов видеопамяти. Особенно велики затраты видеопамяти на хранение *текстур* — растровых картинок, которыми покрываются поверхности трехмерных объектов. Для того чтобы снизить необходимый для хранения текстур объем дорогостоящей видеопамяти, текстуры хранят в оперативной памяти PC, подгружая их в видеопамять ускорителя трехмерной графики по мере необходимости. Такой подход значительно снижает стоимость 3D-акселераторов, но значительно ужесточает требования к скорости обмена данными между видеоадаптером и оперативной памятью PC. Поэтому специально для задач трехмерной графики была создана скоростная локальная шина AGP *(Accelerated Graphic Port).* Практически все современные материнские платы с процессором Pentium II оснащаются шиной AGP. Более подробно особенности этого стандарта будут рассмотрены в гл. 3.

2.2.4. Унификация возможностей видеоадаптеров SVGA. Спецификация VESA

Как уже неоднократно отмечалось, на начальном этапе развития видеоадаптеров SVGA серьезной проблемой было отсутствие единого подхода к реализации расширенных, по отношению к VGA, видеорежимов. Видеоадаптеры SVGA конкурирующих фирм-производителей имели весьма существенные различия в своей конструкции, а именно:

- различную нумерацию расширенных видеорежимов;
- разную регистровую модель (назначение дополнительных, по отношению к архитектуре VGA, регистров, адреса портов ввода/вывода и механизмы их адресации);
- различные методы логической организации видеопамяти (способы разбиения на банки и методы их переключения).

В результате стало невозможным создание универсальной прикладной программы, которая могла бы одинаково успешно работать с различными видеоадаптерами в расширенных видеорежимах (со стандартными режимами VGA проблем не возникало). В разд. 2.1.2 отмечалось, что наиболее надежный (хотя и медленный) *аппаратно независимый* способ работы с видеоадаптером — использование видеосервиса BIOS. Однако отсутствие единой нумерации расширенных видеорежимов не позволяло использовать даже его. Например, для выполнения наиболее простого действия — установки видеорежима с помощью функции 00h прерывания INT 10h — необходимо занести в регистр AL центрального процессора номер видеорежима. Поскольку в каждом видеоадаптере SVGA используется своя собственная нумерация SVGA-режимов, то выполнение данной программы на двух PC с различными видеоадаптерами приведет к совершенно разным результатам. Если же попытаться выполнить более сложные действия (например,

непосредственную запись данных в регистры видеоадаптера, обратиться к видеопамяти в расширенных режимах и т. п.), то крах операционной системы обеспечен.

Наконец, первоначально был неясен точный смысл самого понятия SVGA. Какие видеорежимы следует считать стандартными (т. е. режимами VGA), а какие - расширенными, т. е. SVGA-режимами? Вопрос в первую очередь касался видеорежима 800×600/16, для реализации которого требовалось менее 256 Кбайт видеопамяти, благодаря чему большинство видеоадаптеров VGA могли его поддерживать при соответствующем программировании контроллера ЭЛТ. В частности, имея в составе PC видеоадаптер VGA и работая в ОС Windows, можно установить драйвер «безымянного» видеоадаптера, обозначаемый как **SuperVGA, 800×600,16 colors**. Скорее всего, в этом режиме система будет работать нормально, а проблемы могут возникнуть лишь при частом переключении в режим MS-DOS и обратно.

Основные усилия по разрешению проблемы совместимости приложила ассоциация VESA. Суть предложенных ею мер сводилась к следующему:

- считать видеорежимами SVGA только такие режимы, для реализации которых требуется свыше 256 Кбайт видеопамяти;
- использовать единую для всех видеоадаптеров SVGA нумерацию расширенных видеорежимов;
- использовать одинаковый механизм переключения банков видеопамяти;
- дополнить стандартную Video BIOS специальным *расширением,* в котором хранятся программы, реализующие *расширенные* функции Video BIOS, т. е. управление видеосистемой в расширенных видеорежимах.

При решении проблемы классификации и нумерации видеорежимов специалисты VESA учли то обстоятельство, что практически все PC, оснащаемые видеоадаптерами SVGA, имеют 32-разрядные CPU (80386 и выше). Это значит, что младшие и старшие части регистров общего назначения (например, AL и AH регистра AX) имеют в этих процессорах по 16 разрядов, т. е. вмещают два байта. Следовательно, номер видеорежима, который при вызове функции 00h прерывания INT 10h помещается в AL, для расширенных видеорежимов можно сделать двухбайтным. Двухбайтная нумерация расширенных видеорежимов, названных в дальнейшем *VESA-режимами,* явилась для различных производителей видеоадаптеров SVGA удобным компромиссом: с одной стороны, все они сохранили свою оригинальную нумерацию SVGA-режимов, используя для них однобайтные номера в пределах 14h - 7Fh, с другой — каждый SVGA-режим получил второй, единый для всех видеоадаптеров номер, состоящий из двух байт. Двухбайтные номера VESA-режимов начинаются с номера 100h. Видеоадаптер может устанавливать требуемый видеорежим при указании одного из двух номеров — собственного однобайтного или стандартного двухбайтного.

Для обеспечения совместимости с программами, написанными для обычного видеоадаптера VGA, некоторые из его видеорежимов также включены в состав VESA-режимов. В частности, стандартный режим 12h (640×480/16) получил номер 100h. Режим 800×600/16 в большинстве VGA-совместимых адаптеров имеет номер 6Ah, а в спецификации VESA — 102h. В табл. 2.13 приведен список VESA-режимов, а также собственные однобайтные номера видеорежимов некоторых моделей видеоадаптеров SVGA.

Таблица 2.13. Видеорежимы VESA

Номер	Тип	Разрешение	Глубина цвета/ формат	Палитра	Собственный номер режима видеоадаптера		
					TridentTVGA 8900	Western Digital WD90C30	Realtec RTG3160
100h	Граф.	640×400	8	256	5Ch	5Eh	25h
101h	Граф.	640×480	8	256	5Dh	5Fh	26h
102h	Граф.	800×600	4	16	5Bh	58h	1Fh
103h	Граф.	800×600	8	256	5Eh	5Ch	27h
104h	Граф.	1024×768	4	16	5Fh	5Dh	21h
105h	Граф.	1024×768	8	256	62h	60h	28h
106h	Граф.	1280×1024	4	16	63h	64h	2Ah
107h	Граф.	1280×1024	8	256	64h	—	—
108h	Текст.	80×60*	4	16	—	—	—
109h	Текст.	132×25*	4	16	—	—	—
10Ah	Текст.	132×43*	4	16	—	—	—
10Bh	Текст.	132×50*	4	16	—	—	—
10Ch	Текст.	132×60*	4	16	—	—	—
10Dh	Граф.	320×200	15/1:5:5:5	32 768	7Eh	68h	—
10Eh	Граф.	320×200	16/5:6:5	65 536	7Fh	—	38h
10Fh	Граф.	320×200	24/8:8:8	16 77 7216	6Bh	—	—
110h	Граф.	640×480	15/1:5:5:5	32 768	74h	62h	—
111h	Граф.	640×480	16/5:6:5	65 536	75h	—	3Bh

Таблица 2.13. Видеорежимы VESA (продолжение)

Номер	Тип	Разрешение	Глубина цвета/ формат	Палитра	Собственный номер режима видеоадаптера		
					TridentTVGA 8900	Western Digital WD90C30	Realtec RTG3160
112h	Граф.	640×480	24/8:8:8 или 32/8:8:8:8	16 777 216	6Ch	—	—
113h	Граф.	800×600	15/1:5:5:5	32 768	76h	63h	—
114h	Граф.	800×600	16/5:6:5	65 536	77h	—	3Ch
115h	Граф.	800×600	24/8:8:8 или 32/8:8:8:8	16 777 216	6Dh	—	—
116h	Граф.	1024×768	15/1:5:5:5	32 768	78h	—	—
117h	Граф.	1024×768	16/5:6:5	65 536	79h	—	—
118h	Граф.	1024×768	24/8:8:8 или 32/8:8:8:8	16 777 216	—	—	—
119h	Граф.	1280×1024	15/1:5:5:5	32 768	—	—	—
11Ah	Граф.	1280×1024	16/5:6:5	65 536	—	—	—
11Bh	Граф.	1280×1024	24/8:8:8 или 32/8:8:8:8	16 777 216	—	—	—
11Ch	Граф.	1600×1200	15/1:5:5:5:	32 768	—	—	—
11Dh	Граф.	1600×1200	16/5:6:5	65 536	—	—	—
11Eh	Граф.	1600×1200	24/8:8:8 или 32/8:8:8:8	16 777 216	—	—	—
81FFh	Специ-альный	—	—	—	—	—	—

Примечание. В текстовых режимах вместо разрешения экрана приведен его размер в символьных позициях; о формате задания цвета см. 3.1.2.5.

2.2.4.1. VESA-расширение B1OS

Помимо единой нумерации расширенных видеорежимов, стандартом VESA определены способы их использования. С этой целью стандартный Video BIOS адаптера VGA был дополнен программами, управляющими видеоадаптером в режимах SVGA. Это дополнение называется *VESA-расширением B1OS* или *VBE(VESA BIOS Extension).* Подобно стандартному Video BIOS (см. табл. 2.2), VBE содержит программы, реализующие *дополнительные графические функции,* необходимые в видеорежимах VESA (табл. 2.14).

Таблица 2.14. Графические функции VESA BIOS Extension (для видеорежимов SVGA)

Номер функции	Описание функции
	Информационные функции
4F00h	Проверка наличия в Video BIOS видеоадаптера поддержки VBE
4F01h	Информация о видеорежимах, определенных в VBE
	Управляющие функции
4F02h	Установка видеорежимов, определенных в VBE
4F03h	Информация о текущем видеорежиме VESA (SVGA)
4F04h	Сохранение и восстановление состояния видеосистемы SVGA
4F05h	Определение окна видеопамяти в режиме SVGA
4F06h	Установка/чтение длины строки
4F07h	Установка/чтение начала экрана
4F08h	Установка/чтение формата палитры RAMDAC
4F09h	Установка/чтение регистров палитры RAMDAC
4F0Ah	Интерфейс VBE для защищенного режима работы CPU
	Дополнительные функции VBE
4F10h	Управление энергопотреблением (поддержка стандарта DPMS)
4F11h	Интерфейс плоскопанельных мониторов
4F12h	Расширение интерфейса курсора
4F13h	Расширение аудиоинтерфейса
4F14h	Расширение по усмотрению производителя
4F15h	Управление каналом данных монитора DDC
4F16h	Конфигурирование графической системы SVGA

Помимо управления изображением в этих режимах, функции VBE используются для поддержки канала данных (DDC) монитора, управления энергопотреблением, сопряжения видеоадаптера с плоскопанельными мониторами и др.

1.2.4.1.1. Информационные функции VBE

Для того чтобы упростить использование возможностей конкретного видеоадаптера SVGA, в состав VBE включены специальные ***информационные функции*** (с номерами 4F00h и 4F01h), которых не было в стандартном Video BIOS. Они позволяют получить подробную информацию обо всех видеорежимах, поддерживаемых данным видеоадаптером, что облегчает прикладной программе их корректное использование. Рассмотрим их более подробно в реализации VESA BIOS 2.0:

1.2.4.1.1.1. Прерывание Int 10h, функция 4Fh, подфункция 00h: *ПОЛУЧИТЬ ИНФОРМАЦИЮ О ВЕРСИИ VESA BIOS*

Функция предназначена для считывания информации о версии VESA BIOS (информация представлена в виде структуры, описанной в табл.2 .15).

Перед вызовом прерывания требуется занести в регистры следующие значения:

AX = 4F00h;

ES:DI → адрес буфера объемом 512 байт для сохранения информации о версии VESA BIOS (информация сохраняется в виде структуры, описанной в табл. 2.15).

Таблица 2.15. Формат информации VESA BIOS

Смещение	Размер	Описание
00h	4 байта	Сигнатура 'VESA'
04h	WORD	Номер версии VESA (0200h или 0300h)
06h	DWORD	Указатель на строку OEM (с наименованием микросхемы видеоконтроллера)
0Ah	4 байта	Возможности графического контроллера: бит 0: 0 - 6-разрядный ЦАП, 1 - 8-разрядный; бит 1: 0 - контроллер VGA-совместимый, 1 - несовместимый; бит 2: 0 - обычная работа с RAMDAC, 1 - при программировании больших блоков информации в RAMDAC использовать бит очистки функции 09h; биты 3-31 зарезервированы
0Eh	DWORD	Указатель на список Видеорежимов, поддерживаемых VESA и OEM
12h	WORD	Число 64-килобайтных блоков памяти (объем видеопамяти, деленный на 64 Кбайт)
14h	WORD	Код версии программной реализации VESA
16h	DWORD	Указатель на строку с названием фирмы-изготовителя
1Ah	DWORD	Указатель на строку с названием изделия
1Eh	DWORD	Указатель на строку с кодом версии изделия
22h	222 байта	Зарезервировано для последующих версий
100h	256 байт	Область данных для строк OEM

Подфункцию 00h необходимо вызвать перед началом использования других VESA-функций. Она позволяет определить, имеется ли у видеоконтроллера поддержка команд VESA, и если да - то какой версии. Чтобы имелась возможность использования линейного буфера, версия VESA должна быть не ниже 2.0.

Вначале нужно убедиться, что функция была выполнена: после вызова функции в регистре AL должен находиться код 4Fh. Затем следует проверить наличие сигнатуры 'VESA' в четырех первых байтах возвращенного функцией блока данных, а также убедиться, что байт со смещением 05h от начала блока (старший байт номера версии) содержит значение не меньше 2.

1.2.4.1.1.2. Прерывание Int 10h, функция 4Fh, подфункция 01h: *ПОЛУЧИТЬ ИНФОРМАЦИЮ О ПАРАМЕТРАХ ВИДЕОРЕЖИМА*

Функция позволяет проверить наличие в данной версии BIOS режима с заданным номером и определить его свойства.

Перед вызовом прерывания требуется занести в регистры следующие значения:

AX = 4F01h;
CX — код видеорежима, параметры которого нужно определить;
ES:DI → адрес буфера объемом 256 байт для сохранения информации о видеорежиме (табл. 2.16).

Подфункцию 0lh обычно вызывают перед установкой нового видеорежима, чтобы удостовериться в его реализации в данном видеоконтроллере и в возможности использования линейной адресации видеопамяти в этом режиме. После вызова функции в регистре AL должен находиться код 4Fh, в AH - код 0. После этого нужно проверить седьмой бит слова атрибутов режима (линейная адресация поддерживается, если он равен 1), а затем прочитать адрес линейного видеобуфера из 32-разрядного (двойного) слова со смещением 28h от начала структуры данных. Подфункции 00h и 0lh необходимо использовать, если нет полной уверенности в том, что видеоконтроллер поддерживает используемый в вашей программе видеорежим.

Таблица 2.16. Формат информации о видеорежиме

Смещение	Размер	Описание
00h	WORD	Атрибуты режима: бит 0: 0 - режим не поддерживается, 1 - режим поддерживается; бит 1: всегда 1 (зарезервирован); бит 2: 0 - TTY функции не поддерживаются, 1 —TTY функции поддерживаются; бит 3: 0 — монохромный режим, 1 — цветной режим; бит 4: 0 — текстовый режим, 1 — графический режим; бит 5: 0 — VGA-совместимый режим, 1 — VGA-несовместимый режим; бит 6: 0 — режим оконной адресации видеопамяти поддерживается, 1 —не поддерживается; бит 7: 0 — режим линейной адресации видеопамяти не поддерживается, 1 — поддерживается; биты 8-15 зарезервированы
02h	BYTE	Атрибуты окна A: бит 0: 0 — окно неперемещаемое, 1 — поддерживается перемещение окна; бит 1: 0— чтение из окна запрещено, 1 — разрешено; бит 2: 0— запись в окно запрещена, 1 — разрешена; биты 3-7 зарезервированы
03h	BYTE	Атрибуты окна B (аналогично байту атрибутов окна A)
04h	WORD	Гранулярность (дробность) окна (наименьшая величина в килобайтах, на которую можно переместить окно)
06h	WORD	Размер окна в килобайтах
08h	WORD	Начальный сегмент окна A в адресном пространстве процессора
0Ah	WORD	Начальный сегмент окна B
0Ch	DWORD	FAR-адрес функции позиционирования окна (она эквивалентна функции 4Fh с подфункцией 05h)
10h	WORD	Число байтов, которое приходится на одну строку развертки
12h	WORD	Горизонтальное разрешение экрана в пикселах (в графическом режиме) или знакоместах (в текстовом режиме)
14h	WORD	Вертикальное разрешение экрана в пикселах или знакоместах
16h	BYTE	Ширина знакоместа в пикселах
17h	BYTE	Высота знакоместа в пикселах
18h	BYTE	Число плоскостей памяти
19h	BYTE	Число битов на пиксел
1Ah	BYTE	Число банков памяти

Таблица 2.16 Формат информации о видеорежиме *(продолжение)*

Смещение	Размер	Описание
1Bh	BYTE	Тип модели памяти: 00h - текстовый режим; 01h - графический режим CGA; 02h - графический режим Hercules; 03h - планарный режим; 04h - режим упакованных пикселов; 05h - нецепочечный режим № 4, 256 цветов; 06h - режим Direct Color (HiColor или TrueColor); 07h - режим YUV; 08h - 0Fh — зарезервировано для дополнений VESA; 10h - FFh — зарезервировано для дополнений OEM
1Ch	BYTE	Размер банка памяти в килобайтах
1Dh	BYTE	Число полных видеостраниц (экранов), помещающихся в видеопамяти, минус единица
1Eh	BYTE	Зарезервирован

Поля Direct Color

Смещение	Размер	Описание
1Fh	BYTE	Размер компоненты красного цвета в битах
20h	BYTE	Начальная битовая позиция компоненты красного цвета
21h	BYTE	Размер компоненты зеленого цвета в битах
22h	BYTE	Начальная битовая позиция компоненты зеленого цвета
23h	BYTE	Размер компоненты синего цвета в битах
24h	BYTE	Начальная битовая позиция компоненты синего цвета
25h	BYTE	Размер резервной компоненты в битах
26h	BYTE	Начальная битовая позиция резервной компоненты
27h	BYTE	Атрибуты режима Direct Color

Поля, присутствующие только в VBE 2.0 и позднейших реализациях стандарта

Смещение	Размер	Описание
28h	DWORD	Физический (абсолютный) 32-разрядный адрес буфера кадра
2Ch	DWORD	Указатель на начало заэкранной памяти, то есть на начало второй видеостраницы (введен в версии 2.0, но изъят из версии 3.0; теперь данное поле считается зарезервированным и содержит значение 0)
30h	WORD	Объем заэкранной памяти в килобайтах (введен в версии 2.0, но изъят из версии 3.0; теперь данное поле считается зарезервированным и содержит значение 0)

Плоя, присутствующие только в VBE 3.0 и позднейших реализациях стандарта

Смещение	Размер	Описание
34h	BYTE	Количество видеостраниц в страничном режиме
35h	BYTE	Количество видеостраниц в линейном режиме
36h	BYTE	Размер компоненты красного цвета в битах в линейном режиме
37h	BYTE	Начальная битовая позиция компоненты красного цвета в линейном режиме
38h	BYTE	Размер компоненты зеленого цвета в битах в линейном режиме
39h	BYTE	Начальная битовая позиция компоненты зеленого цвета в линейном режиме
40h	BYTE	Размер компоненты синего цвета в битах в линейном режиме
41h	BYTE	Начальная битовая позиция компоненты синего цвета в линейном режиме
42h	BYTE	Размер резервной компоненты в битах в линейном режиме
43h	BYTE	Начальная битовая позиция резервной компоненты в линейном режиме
44h	DWORD	Максимальная частота вывода пикселов в графических режимах (в Гц)
48h	184 байта	Зарезервировано

1.2.4.1.2. Управляющие функции VBE

***Управляющие функции* VBE** предназначены главным образом для решения задач, характерных именно для режимов SVGA, а именно:

- для реализации универсального способа переключения банков видеопамяти (функция 4F05h);
- для управления работой видеоадаптера в режимах High Color и True Color (функции 4F08h, 4F09h);
- для реализации *линейной адресации* кадрового буфера, доступной в защищенном режиме работы CPU.

Рассмотрим их более подробно в реализации VESA BIOS 2.0.

1.2.4.1.2.1. Прерывание Int 10h, функция 4Fh, подфункция 02h: *УСТАНОВИТЬ ВИДЕОРЕЖИМ С ЗАДАННЫМ НОМЕРОМ*

Функция устанавливает видеорежим с заданным номером и режим адресации видеопамяти (чтобы использовать линейную адресацию, необходимо установить бит 14 в коде режима).

Перед вызовом прерывания требуется занести в регистры следующие значения:

AX - код 4F02h;

BX — код видеорежима.

Код режима имеет следующий формат:

- биты 0 - 8 - номер режима (если бит 8 равен 0, то это режим производителя видеоконтроллера, если 1 — VESA-режим);
- биты 9-13 - зарезервированы для дальнейших расширений и равны 0;
- бит 14 - тип режима адресации видеопамяти: 0 - страничный режим (память адресуется участками по 64 Кбайт), 1 - линейный режим;
- бит 15 - признак очистки видеопамяти при переключении режимов (0 - очищать память, 1 - не очищать).

Стандартизированные номера режимов приведены в табл. 2.13. При работе с функцией установки видеорежима нужно учитывать следующее.

- Режимы с номерами меньше l00h определяются изготовителем видеоконтроллера и не обрабатываются VESA-функциями. Если функции установки видеорежима будет послан код со значением меньше 100h, то она передаст его на обработку прерыванию BIOS 10h.
- Видеоконтроллеры различных производителей могут интерпретировать режимы True Color с номерами 10Fh, 112h, 115h, 118h, 11Bh как 24- или как 32-битовые, то есть данные режимы определены неоднозначно. Для уточнения числа байтов, выделенных на одну точку, нужно вызвать функцию 01h, получить информацию о видеорежиме и проверить размер резервной компоненты (байт со смещением 25h): если 0 — 24-битовый режим, если 8 — 32-битовый.
- Коды режимов с разрешением более 1280×1024 не стандартизированы, и их приходится определять по таблице доступных режимов, заданной изготовителем видеоконтроллера.
- Режим 81FFh служит для обеспечения доступа ко всей имеющейся видеопамяти без изменения ее содержимого.

1.2.4.1.2.2. Прерывание Int 10h, функция 4Fh, подфункция 03h: *ОПРЕДЕЛИТЬ КОД ТЕКУЩЕГО ВИДЕОРЕЖИМА*

Функция возвращает код текущего видеорежима

AX = 4F03h.

Возвращает:

AX = код результата выполнения функции (код возврата);

BX = код текущего видеорежима (см. табл. 2.13).

1.2.4.1.2.3. Прерывание Int 10h, функция 4Fh, подфункция 04h: *СОХРАНИТЬ ИЛИ ВОССТАНОВИТЬ СОСТОЯНИЕ ВИДЕОКОНТРОЛЛЕРА*

Функция сохраняет или восстанавливает режим работы видеоконтроллера. Сохранение состояния заключается том, что функция записывает в выделенный буфер оперативной памяти содержимое всех регистров видеоконтроллера. Содержимое видеопамяти функция не записывает — эту операцию необходимо выполнять отдельно.

Перед вызовом прерывания требуется занести в регистры следующие значения:

AX = 4F04h;

DL = код подфункции:

0 - получить размер буфера, необходимого для сохранения состояния,

1 - сохранить состояние,

2 - восстановить состояние;

CX = флаги, уточняющие, состояние каких именно блоков видеоконтроллера нужно

сохранить или восстановить:

бит 0 — состояние аппаратуры,
бит 1 — состояние данных BIOS,
бит 2 — состояние DAC,
бит 3 — состояние регистров,
биты 4 - 15 — зарезервированы;

ES: BX — указатель на буфер (нужен только в том случае, когда в DL не 0).

Возвращает:

AX = код возврата,
BX = число блоков по 64 байта, необходимых для хранения состояния (если в DL = 0).

Подфункция 04h позволяет запомнить состояние видеоконтроллера перед переходом в какой-либо другой режим, чтобы потом можно было с ее помощью вернуться в исходное состояние.

1.2.4.1.2.4. Прерывание Int 10h, функция 4Fh, подфункция 05h: *УПРАВЛЕНИЕ ОКНАМИ ВИДЕОПАМЯТИ*

Функция предназначена для управления отображением видеопамяти на адресное пространство процессора при использовании страничного доступа (через окно 64 Кбайт).

Перед вызовом прерывания требуется занести в регистры следующие значения:

AX = 4F05h;

BH = код выполняемой операции:

0 — установить окно видеопамяти,
1 — получить номер текущего окна видеопамяти;

BL = номер окна (0 — окно A, 1 — окно B);

DX = номер окна в видеопамяти (в единицах дробности); задается только при BH = 0.

Возвращает:

AX = код возврата,
DX — адрес окна в видеопамяти в единицах дробности (выдается только при BH =1).

Подфункция 05h применяется в оконном режиме адресации, когда в каждый момент времени процессору доступен только один участок (сегмент) видеопамяти размером в 64 Кбайт. Ее необходимо вызывать каждый раз, когда требуется перейти к работе с другим участком.

1.2.4.1.2.5. Прерывание Int 10h, функция 4Fh, подфункция 06h: *ПОЛУЧИТЬ ИЛИ УСТАНОВИТЬ ДЛИНУ ЛОГИЧЕСКОЙ СТРОКИ РАЗВЕРТКИ*

Функция читает текущую длину логической строки либо устанавливает ее новое значение.

Перед вызовом прерывания требуется занести в регистры следующие значения:

AX = 4F06h;

BL = код выполняемой операции

0 — установить длину строки развертки в пикселах,
1 — получить текущую длину строки развертки в пикселах,
2 — установить длину строки развертки в байтах,
3 — получить максимально возможную длину строки развертки в пикселах;

CX = требуемая ширина строки (задается только для операций 0 и 2: при BL = 0 — в пикселах, при BL = 2 — в байтах).

Возвращает:

AX = код возврата;
BX = число байт на строку развертки;
CX = число пикселов на строку развертки (округленное до ближайшего целого пиксела);
DX = максимальное число строк развертки.

Подфункция 06h используется для изменения организации видеопамяти с целью упрощения адресации (путем выравнивания длины строки на величину, равную 2^N = 1024 или 2048). Видеоконтроллеры позволяют делать длину строки в видеопамяти (логическую) больше реальной (физической) длины строки экрана за счет реорганизации неиспользуемых областей памяти. Обычно невидимая область располагается под видимой (рис. 2.21, *слева),* однако после увеличения логической длины строки происходит перераспределение памяти — появляется невидимый участок справа, а высота нижнего неотображаемого участка уменьшается (рис. 2.21, *справа).*

Выравнивание логической длины строки на 2^N позволяет:

- упростить вычисление адресов памяти (используя сдвиги и логические операции вместо умножения и сложения);
- реализовать (при необходимости) плавную прокрутку изображения по вертикали аппаратными средствами видеоконтроллера;

- существенно упростить контроль границ сегментов при оконном режиме адресации видеопамяти.

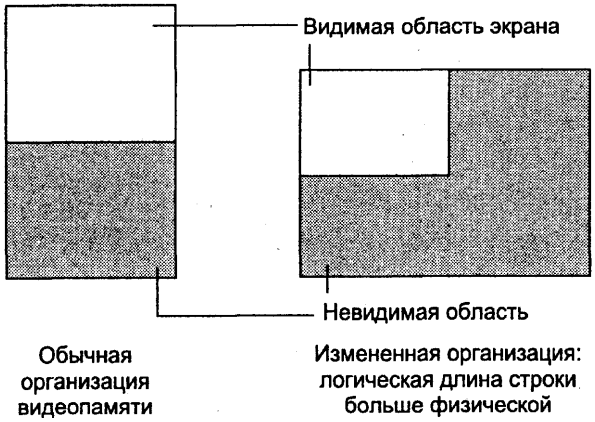


Рис. 2.21. Реорганизация видеопамяти после изменения логической длины строки

Для пояснения последнего пункта рассмотрим обычную организацию видеопамяти в режиме оконной адресации. Если длина строки в пикселах не кратна 2^N, то сегмент будет содержать нецелое число экранных строк, и границы сегментов не будут совпадать с границами экрана (рис. 2.22, *слева).* В результате при выводе изображения на экран необходимо контролировать не только нижнюю, но и правую границу сегмента (вывод выполняется по точкам, слева направо и сверху вниз). Такую операцию контроля приходится включать внутрь всех циклов вывода изображения, что сильно тормозит работу: вывод точки на экран выполняется одной простой командой пересылки данных MOV, и включение любых других команд (в том числе для проверки границ) в цикл вывода точек замедляет его выполнение в несколько раз. Замедление особенно заметно на современных быстрых видеоконтроллерах, у которых скорость записи в память измеряется десятками мегабайтов в секунду.

Рис. 2.22. Выравнивание границ сегментов после изменения логической длины строки

Если выровнять длину строки в пикселах на 2^N, то сегмент будет состоять из целого числа экранных строк, а начало каждого сегмента будет совпадать с началом строки экрана (рис. 2.22, *справа).* Операция контроля пересечения границы очередного сегмента при этом проводится во внешнем цикле вывода строки изображения, а не во внутреннем цикле рисования точек. Объем памяти при низких разрешениях (при длине строки 640 или 800 точек) обычно не является препятствием для операции выравнивания даже для видеоконтроллеров нижнего ценового уровня, так как самые примитивные современные карты имеют память емкостью 4 - 8 Мбайт.

1.2.4.1.2.6. Прерывание Int 10h, функция 4Fh, подфункция 07h: *ПОЛУЧИТЬ ИЛИ УСТАНОВИТЬ КООРДИНАТЫ ЛЕВОГО ВЕРХНЕГО УГЛА ЭКРАНА*

Функция читает или устанавливает номер пиксела, отображаемого в левом верхнем углу экрана.

Перед вызовом прерывания требуется занести в регистры следующие значения:

AX = 4P07h;

BH = 0 (регистр зарезервирован для последующих расширений);

BL = код выполняемой операции

00h — установить начало экрана,

01h — получить начало экрана,

80h — установить начало экрана в процессе обратного хода луча по кадру;

CX = первый отображаемый пиксел на строке развертки (задается только при BL = 00h или 80h);

DX = первая отображаемая строка развертки (задается только при BL = 00h или 80h).

Возвращает:

AX = код возврата;

CX = первый отображаемый пиксел на строке развертки (только при BL = 01h);

DX = первая отображаемая строка развертки (только при BL = 01h).

Подфункция 07h имеет две основные области применения: прокрутка экрана и *переключение экранов*. В обоих случаях происходит перемещение по видеопамяти области, отображаемой на экран, но в первом случае — плавно, во втором — скачкообразно. Такое перемещение возможно благодаря тому, что система управления памятью видеоконтроллера позволяет произвольно выбирать точку, с которой начинается вывод на экран (позицию левого верхнего угла экрана относительно видеопамяти).

Для реализации прокрутки изображения влево необходимо перед выводом очередного кадра прибавлять единицу к номеру начальной точки, а для реализации прокрутки вправо — вычитать единицу. Для реализации прокрутки вверх необходимо к номеру начальной точки прибавлять число точек в строке, для реализации прокрутки вниз — вычитать его.

Для выполнения переключения экранов (окон) необходимо вычислить смещение второго окна относительно начала видеопамяти. Оно равно числу точек в строке изображения, помноженному на число строк на экране и на число байтов, приходящихся на одну точку. Далее для каждого нового кадра происходит поочередная установка номера первого выводимого пиксела (0 или величина смещения второго окна). Пока одно окно выводится на экран, происходит перезапись информации в другом окне. Таким образом можно исключить эффекты типа «снега» и разрезания изображения, возникающие при одновременной записи в участок видеопамяти и выводе на экран информации из этого же участка.

1.2.4.1.2.7. Прерывание Int 10h, функция 4Fh, подфункция 08h: *ПОЛУЧИТЬ ИЛИ ИЗМЕНИТЬ ФОРМАТ РЕГИСТРОВ ПАЛИТРЫ*

Функция читает или устанавливает разрядность регистров палитры (6 или 8 бит). Основное назначение функции — расширение цветовой палитры в режиме 256 цветов с 2^{18} до 2^{24} оттенков. В настоящий момент функция устарела, так как для работы с широкой цветовой палитрой гораздо эффективнее использовать режим True Color. Единственной областью ее применения, вероятно, являются графические редакторы, работающие с форматом файлов PCX.

Перед вызовом прерывания требуется занести в регистры следующие значения:

AX = 4F08h;
BL — код выполняемой операции
0 — установить формат регистров палитры,
1 — получить формат регистров палитры;
BH (при BL = 0) = требуемая разрядность регистров ЦАП в битах (допустимы только 6 и 8).

Возвращает:

AX — код возврата;
BH — текущая разрядность регистров ЦАП в битах.

ПРИМЕЧАНИЕ Функция выдает код ошибки 3 при вызове в режимах Direct Color или YUV.

1.2.4.1.2.8. Прерывание Int 10h, функция 4Fh, подфункция 09h: *СОХРАНИТЬ ИЛИ ИЗМЕНИТЬ СОДЕРЖИМОЕ РЕГИСТРОВ ЦАП*

Функция считывает или устанавливает содержимое таблицы регистров цифро-аналогового преобразователя (DAC), которая ставит в соответствие кодам цвета реальные значения по красному, зеленому и синему компонентам (в 16- и 256-цветных режимах). Это доработанный вариант подфункций 12h и 17h функции 10h прерывания Int 10h VGA BIOS. Функция 09h позволяет перепрограммировать таблицу палитры (в режиме 256 цветов) с целью получения набора оттенков, более подходящего для решения стоящих перед программистом задач, чем стандартный набор. Например, можно организовать плавные цветовые переходы.

Перед вызовом прерывания требуется занести в регистры следующие значения:

AX = 4F09h;
BL = код выполняемой операции
00h — установить палитру,
0lh — прочитать (сохранить) палитру,
02h — установить дополнительную (вторичную) палитру,
03h — прочитать дополнительную (вторичную) палитру,
80h — установить палитру в процессе обратного хода луча по кадру);
CX = число загружаемых (или сохраняемых) регистров палитры (максимум 256);
DX = номер первого из загружаемых (сохраняемых) регистров;
ES:DI = указатель на таблицу, из которой загружается или в которой сохраняется содержимое регистров палитры.

Формат строки таблицы палитры следующий: байт выравнивания (нулевой), байт интенсивности красного цвета, байт интенсивности зеленого цвета, байт интенсивности синего цвета.

1.2.4.1.2.9. Прерывание Int 10h, функция 4Fh, подфункция 0Ah: *ПОЛУЧИТЬ ТАБЛИЦУ ДОСТУПА К ИНТЕРФЕЙСУ ЗАЩИЩЕННОГО РЕЖИМА*

Функция возвращает указатель на таблицу доступа к интерфейсу 32-разрядного защищенного режима. Перед вызовом прерывания требуется занести в регистры следующие значения:

AX = 4F0Ah;
BX = 00h.

Возвращает:

AX = код возврата;
ES:DI — указатель на таблицу доступа к интерфейсу защищенного режима;
CX = размер таблицы доступа в байтах, включая размер кода для работы в защищенном режиме.

Таблица доступа к интерфейсу защищенного режима содержит четыре слова-указателя, вслед за которыми располагается программный код для работы в защищенном режиме. Указатели размещены в таблице доступа в следующем порядке:

- слово со смещением 00h содержит смещение кода функции 05h (для защищенного режима) от начала таблицы доступа;
- слово со смещением 02h содержит смещение кода функции 07h (для защищенного режима) от начала таблицы доступа;
- слово со смещением 04h содержит смещение кода функции 09h (для защищенного режима) от начала таблицы доступа;
- слово со смещением 06h содержит смещение таблицы портов и описателей области памяти, для доступа к которым нужно устанавливать соответствующий уровень привилегий, от начала таблицы доступа (значение 0000h в данном слове указывает на отсутствие таблицы).

Таблица портов и областей памяти устроена следующим образом: в начале находится список 16-разрядных адресов портов ввода вывода, который заканчивается терминатором FFFFh. За терминатором списка портов следуют 32-разрядный базовый адрес области памяти, 16-разрядный лимит сегмента и терминатор описателя области памяти FFFFh.

Для обеспечения доступа к программному коду в защищенном режиме нужно либо скопировать все содержимое таблицы доступа в 32-разрядный сегмент, либо настроить селекторы и порты ввода-вывода прямо на соответствующую область ПЗУ.

1.2.4.1.3. Дополнительные функции VBE

***Дополнительные функции* VBE** позволяют наиболее полно использовать возможности видеосистемы современного мультимедийного PC. При помощи функции 4F10h операционная система реализует энергосберегающие функции монитора согласно стандарту DPMS, также разработанному VESA (см. [2], разд. 2.3.2). Напомним, что управление состоянием монитора, согласно данному стандарту, осуществляется путем отключения сигналов синхронизации, что и позволяет сделать функция 4F10h. С помощью функции 4F13h можно управлять встроенной акустической системой мультимедийного монитора. При помощи функции 4F15h реализуется поддержка мониторов Plug&Play, использующих для обмена данными специальный канал DDC (см. [2], разд. 2.4).

Появление VBE стало важным этапом в развитии видеоадаптеров, позволившим говорить об SVGA как о стандарте. Первоначально видеоадаптеры SVGA снабжались специальными драйверами, которые при загрузке «дописывали» VBE в память, дополняя обычный Video BIOS (такой прием характерен для спецификации VESA версии 1.0). При отсутствии таких драйверов можно было воспользоваться программой Scientific Display Doctor, поддерживающей большинство распространенных видеоадаптеров. В современных видеоадаптерах SVGA VBE добавлено в ROM Video BIOS, поэтому для их работы не требуется загрузка дополнительных драйверов.

Контрольные вопросы.

1. Какая фирма первой выпустила видеоадаптер с памятью 512 Кбаит и какими особенностями он обладал?
2. Какими особенностями отличался RAMDAC фирмы Sierra Semiconductors?
3. Какие изменения произошли в организации памяти и внутренних шин в видеоадаптерах, использующих 24-разрядный RAMDAC с прямым обращением к ЦАП?
4. На какое адресное пространство отображается видеопамять при использовании в ней режима линейной адресации)для ISA и PCI)?
5. Почему при выше означенных нововведениях проблема низкого быстродействия видеосистемы принципиально на была решена?
6. Для чего в видеоадаптерах стали использовать графические сопроцессоры?
7. Какой элемент видеоадаптера стал "узким местом" после того, как стали использовать мощные графические сопроцессоры и скоростные шины ввода/вывода (VLB и PCI)?
8. Какая организация внесла основной вклад в стандартизацию систем SVGA?
9. Что понимают под графическим ускорителем (акселератором)?

10. За счет чего увеличивается производительность видеосистемы и ПК в целом при использовании графического ускорителя?
11. Чем отличается графический контроллер от графического ускорителя?
12. Что понимается под графическим примитивом и высокоуровневыми командами, посылаемыми в видеоадаптер?
13. Перечислите основные графические функции, при выполнении которых необходимо использовать аппаратные ускорение.
14. В чем заключается особенность задания графических примитивов в векторном виде ?
15. Для чего используются команды прорисовки?
16. Охарактеризуйте операции переноса блоков изображения из одной части экрана в другую?
17. Охарактеризуйте графическую функцию аппаратной поддержки окон.
18. В чем состоит суть графической функции аппаратного курсора?
19. Что такое Sprite и в чем отличие операций перемещения курсора от операций BitBlt?
20. Какие виды масштабирования используются для растровых изображений?
21. Как борятся с эффектами пикселизации и апиасинга при масштабировании?
22. Чем отличается функция панорамирования от прокрутки экрана (операций типа BitBlt)?
23. И чем суть преобразования цветового пространства?
24. Где используется формат YUV?
25. В чем суть функции декомпрессии сжатых видеоданных?
26. Какие дополнительные функции может поддерживать видеоадаптер?
27. Какие функции были реализованы в первом ускорителе (акселераторе Windows), какие - в последующих и какие в последних вариантах?
28. Какие функции реализует типовой современный графический ускоритель двумерных графических данных (2D)?
29. На какие группы первоначально делились видеоадаптеры с точки зрения конкретного способа реализации аппаратной поддержки графических функций?
30. В чем отличие графического сопроцессора от графического процессора?
31. Что представляет из себя графический ускоритель?
32. В чем основное отличие видеоадаптера с графическим акселератором от стандартного видеоадаптера VGA?
33. Какие блоки входят в состав акселератора, и что является результатом его работы?
34. В чем основные отличия графического сопроцессора от графического акселератора?
35. Какими устройствами используется видеопамять?
36. Что представляет собой элемент динамической памяти и какие фазы можно выделить в цикле его работы?
37. Почему время доступа к динамической памяти больше времени рабочего цикла?
38. Как интерпретируется временная диаграмма работы памяти в пакетном режиме?
39. Что представляет собой элементарная ячейка хранения статической памяти(SRAM)?
40. Охарактеризуйте FPM-DRAM.
41. Охарактеризуйте EDO-DRAM.
42. Охарактеризуйте BEDO-DRAM.
43. Каковы временные диаграммы FPM, EDO и BEDO при частоте шины 66 МГц?
44. Охарактеризуйте SDRAM; за счет чего у нее длительность рабочего цикла уменьшена до 7 - 10 нс?
45. Охарактеризуйте SGRAM.
46. Охарактеризуйте MDRAM и проблемы ее использования.
47. В чем особенность двухпортовой памяти.
48. Охарактеризуйте VRAM.
49. Охарактеризуйте WRAM.
50. Охарактеризуйте Direct RDRAM.
51. Охарактеризуйте 3D-RAM.
52. Охарактеризуйте CD RAM.
53. Охарактеризуйте DDR SDRAM.
54. Охарактеризуйте ES DRAM.
55. Каковы скоростные параметры шины ISA, и какие видеорежимы она может обеспечить?
56. Охарактеризуйте шину VLB.
57. Охарактеризуйте шину PCI.
58. Какими параметрами отличались первые SVGA-адаптеры различных изготовителей?
59. В чем суть мер, предложенных ассоциацией VESA по унификации видеорежимов SVGA?
60. В чем суть двухбайтовой нумерации расширенных режимов SVGA (VESA-режимов)?
61. Какие номера имеют режимы 640×480/16 и 800×600/16?
62. Что подразумевается под VBE?
63. Какие функции, помимо управления изображением, реализованы в VBE?
64. Охарактеризуйте видеосистемы VESA.

65. Охарактеризуйте информационные функции VBE.
66. Охарактеризуйте формат информации VBE, получаемой подфункцией 00h функции 4Fh прерывания Int 10h.
67. Охарактеризуйте первые 1Eh байт информации о видеорежиме, возвращаемой подфункцией 01h функции 4Fh прерывания Int 10h.
68. Охарактеризуйте поля Direct Color информации о видеорежиме, возвращаемой подфункцией 01h функции 4Fh прерывания Int 10h (1Fh - 27h).
69. Охарактеризуйте поля информации о видеорежиме, возвращаемой подфункцией 01h функции 4Fh прерывания Int 10h, присутствующие только в VBE 2.0 и более поздних реализациях стандарта (28h - 31h).
70. Охарактеризуйте поля информации о видеорежиме, возвращаемой подфункцией 01h функции 4Fh прерывания Int 10h, присутствующие только в VBE 3.0 и более поздних реализациях стандарта (34h - 47h).
71. Охарактеризуйте управляющие функции VBE.
72. Охарактеризуйте подфункции 02h и 03h функции 4Fh прерывания Int 10h.
73. Охарактеризуйте подфункцию 04h функции 4Fh прерывания Int 10h.
74. Охарактеризуйте подфункцию 05h функции 4Fh прерывания Int 10h.
75. Охарактеризуйте подфункцию 06h функции 4Fh прерывания Int 10h.
76. Охарактеризуйте подфункцию 07h функции 4Fh прерывания Int 10h.
77. Охарактеризуйте подфункцию 08h функции 4Fh прерывания Int 10h.
78. Охарактеризуйте подфункцию 09h функции 4Fh прерывания Int 10h.
79. Охарактеризуйте подфункцию 0Ah функции 4Fh прерывания Int 10h.
80. Охарактеризуйте дополнительные функции VBE.
81. Как поддерживалась VBE при отсутствии ее функций в Video BIOS?

В данном разделе использована информация из [1, 5]. Информацию о современных типах памяти и скоростных шинах ввода-вывода, используемых в современных ПК, можно также найти в [9, 10].

2.2.5. Пример использования функций VESA BIOS.

В программе VESA_BIOS_Test, приведенной в листинге 2.2, используются подфункции 00h и 01h функции 4Fh прерывания int 10h VESA BIOS, чтобы проверить наличие и версию VESA BIOS, а затем поочередно отобразить на экран параметры всех имеющихся видеорежимов.

Как отмечалось в разделе 2.2.4 подфункцию 00h необходимо вызвать перед началом использования других VESA-функций. Она позволяет определить, имеется ли у видеоконтроллера поддержка команд VESA, и если да - то какой версии. Чтобы имелась возможность использования линейного буфера, версия VESA должна быть не ниже 2.0.

Вначале нужно убедиться, что функция была выполнена: после вызова функции в регистре AL должен находиться код 4Fh. Затем следует проверить наличие сигнатуры 'VESA' в четырех первых байтах возвращенного функцией блока данных, а также убедиться, что байт со смещением 05h от начала блока (старший байт номера версии) содержит значение не меньше 2.

Подфункцию 0lh обычно вызывают перед установкой нового видеорежима, чтобы удостовериться в его реализации в данном видеоконтроллере и в возможности использования линейной адресации видеопамяти в этом режиме. После вызова функции в регистре AL должен находиться код 4Fh, в AH - код 0. После этого нужно проверить седьмой бит слова атрибутов режима (линейная адресация поддерживается, если он равен 1), а затем прочитать адрес линейного видеобуфера из 32-разрядного (двойного) слова со смещением 28h от начала структуры данных.

Подфункции 00h и 0lh необходимо использовать, если нет полной уверенности в том, что видеоконтроллер поддерживает используемый в вашей программе видеорежим.

Основной модуль программы VESA_BIOS_Test, функционирует в текстовом режиме и использует дополнительные процедуры вывода данных на экран, процедуры перевода чисел, вспомогательную процедуру ShowVESAString для вывода текстовых строк из области данных VESA BIOS, файлы мнемонических обозначений управляющих клавиш и цветовых кодов, а также файлы макросов.

Первые две группы процедур общего назначения и файлы оформлены в виде подключаемых файлов (include-файлов), приведенных в листингах 2.3, 2.4, 2.5 и 2.6. Листинг процедуры ShowVESAString представлен в рамках листинга 2.2.

Листинг 2.2. Проверка наличия VESA BIOS, а также режима с линейной адресацией, палитрой 256 цветов, разрешением 640×480 и получение его параметров

```
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"

DATASEG
; Код исследуемого видеорежима
GraphicsMode DW ?
; Место для хранения информации VESA BIOS
VESA_BIOS DB 512 DUP(?)
; Место для хранения информации о параметрах видеорежима
VESA_info DB 256 DUP(?)
; Текстовые сообщения
Txt0 DB LIGHTCYAN,0,33,"ТЕСТ VESA BIOS",0
     DB YELLOW,24,29,"Нажмите любую клавишу",0
Txt1 DB 2,29,"Обнаружен драйвер VESA",0
     DB 4,29,"Сигнатура:",0
     DB 5,21,"Номер версии VESA:",0
     DB 6,31,"Имя OEM:",0
     DB 7,22,"Объем памяти, Мб:",0
     DB 12,25,"Номера поддерживаемых режимов:",0
Txt2 DB 4,27,"Параметры режима №     h:",0
     DB 6,13,"Атрибуты режима:",0
     DB 7,13,"Атрибуты окна A:",0
     DB 8,13,"Атрибуты окна B:",0
     DB 9,2,"Гранулярность окна (Кбайт):",0
     DB 10,9,"Размер окна (Кбайт):",0
     DB 11,4,"Начальный сегмент окна A:",0
     DB 12,4,"Начальный сегмент окна B:",0
     DB 13,5,"Адрес функции поз. окна:",0
     DB 14,2,"Байтов на строку развертки:",0
     DB 15,3,"Горизонтальное разрешение:",0
     DB 16,5,"Вертикальное разрешение:",0
     DB 17,11,"Ширина знакоместа:",0
     DB 18,11,"Высота знакоместа:",0
     DB 19,5,"Число плоскостей памяти:",0
     DB 6,46,"Число битов на пиксел:",0
     DB 7,49,"Число банков памяти:",0
     DB 8,51,"Тип модели памяти:",0
     DB 9,44,"Размер банка памяти (Кб):",0
     DB 10,43,"Число полных видеостраниц:",0
     DB 11,43,"Размер красной компоненты:",0
     DB 12,40,"Нач. поз. красной компоненты:",0
     DB 13,43,"Размер зеленой компоненты:",0
     DB 14,40,"Нач. поз. зеленой компоненты:",0
     DB 15,45,"Размер синей компоненты:",0
     DB 16,42,"Нач. поз. синей компоненты:",0
     DB 17,41,"Размер резервной компоненты:",0
     DB 18,43,"Нач. поз. рез. компоненты:",0
     DB 19,41,"Линейный адрес буфера кадра:",0
Err1 DB 25,0,"Команды VESA не поддерживаются",0
ENDS

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

CODESEG
;****************************
;* Основной модуль программы *
;****************************
PROC VESA_BIOS_Test
        mov     AX,DGROUP
        mov     DS,AX
```

Листинг 2.2 (продолжение)

```
        mov     [CS:MainDataSeg],AX
; Установить текстовый режим
        mov     AX,3
        int     10h
; Скрыть курсор - убрать за нижнюю границу экрана
        mov     [ScreenString],25
        mov     [ScreenColumn],0
        call    SetCursorPosition
; Вывести заголовок
        MShowColorText 2,Txt0

; ПРОВЕРИТЬ НАЛИЧИЕ VESA BIOS
; Получить информацию о наличии VESA BIOS
        mov     AX,DS
        mov     ES,AX
        mov     AX,4F00h
        mov     DI,offset VESA_BIOS
        int     10h


    ; Проверить результат выполнения функции
            cmp     AX,4Fh
            jne     @@Err
    ; Вывести поясняющий текст
            mov     [TextColorAndBackground],LIGHTGREEN
            MShowText 6,Txt1
            mov     [TextColorAndBackground],WHITE
    ; Вывести сигнатуру
            mov     SI,offset VESA_BIOS
            MShowASCIIField 4,40,0,4
    ; Вывести номер версии
            MShowHexWord 5,40,<[word ptr VESA_BIOS+4]>
    ; Вывести имя OEM
            call    ShowVESAString
    ; Вывести объем памяти
            mov     AX,[word ptr VESA_BIOS+12h]
            shr     AX,4  ;разделить на 16
            MShowDecWord 7,40,AX
    ; Вывести список номеров режимов
            mov     [ScreenString],14
            mov     [ScreenColumn],0
            mov     BX,[word ptr VESA_BIOS+0Eh]
            mov     AX,[word ptr VESA_BIOS+10h]
            mov     ES,AX
            mov     CX,128
    @@j0:   mov     AX,[ES:BX]
            cmp     AX,0FFFFh
            je      @@j1
            inc     [ScreenColumn]
            call    ShowHexWord
            add     BX,2
            loop    @@j0
    ; Ожидать нажатия любой клавиши
    @@j1:   call    GetChar

    ; ПОКАЗАТЬ ПАРАМТРЫ ДОСТУПНЫХ ВИДЕОРЕЖИМОВ


    ; Загрузить адрес массива номеров
            mov     BX,[word ptr VESA_BIOS+0Eh]
    @@NextMode:
    ; Загрузить кодовый номер очередного режима
            mov     AX,[word ptr VESA_BIOS+10h]
            mov     ES,AX
            mov     AX,[ES:BX]
            mov     [GraphicsMode],AX
            cmp     AX,0FFFFh
            je      @@End
    ; Очистить экран
            call    ClearScreen
    ; Вывести заголовок
            MShowColorText 2,Txt0
    ; Получить параметры видеорежима
            mov     AX,DS
            mov     ES,AX
            mov     AX,4F01h
            mov     CX,[GraphicsMode]
            mov     DI,offset VESA_info
            int     10h
    ; Установить зеленый цвет символов и черный фон
            mov     [TextColorAndBackground],LIGHTGREEN
    ; Вывести текстовые сообщения на экран
            MShowText 29,Txt2
    ; Вывести номер режима
            MShowHexWord   4,46,[GraphicsMode]
    ; Установить белый цвет символов и черный фон
            mov     [TextColorAndBackground],WHITE
    ; Вывести значения полей на экран
            ; Атрибуты режима
            MShowHexWord   6,30,<[word ptr VESA_info]>
            ; Атрибуты окна A
            MShowHexByte   7,30,[VESA_info+02h]
            ; Атрибуты окна B
            MShowHexByte   8,30,[VESA_info+03h]
            ; Гранулярность окна
            MShowDecWord   9,30,<[word ptr VESA_info+04h]>
            ; Размер окна
            MShowDecWord  10,30,<[word ptr VESA_info+06h]>
            ; Начальный сегмент окна A
            MShowHexWord  11,30,<[word ptr VESA_info+08h]>
            ; Начальный сегмент окна B
            MShowHexWord  12,30,<[word ptr VESA_info+0Ah]>
            ; Адрес функции позиционирования окна
            MShowHexDWord 13,30,<[dword ptr VESA_info+0Ch]>
            ; Байт на одну строку развертки
            MShowDecWord  14,30,<[word ptr VESA_info+10h]>
            ; Горизонтальное разрешение
            MShowDecWord  15,30,<[word ptr VESA_info+12h]>
            ; Вертикальное разрешение
            MShowDecWord  16,30,<[word ptr VESA_info+14h]>
            ; Ширина знакоместа
            MShowDecByte  17,30,[VESA_info+16h]
            ; Высота знакоместа
            MShowDecByte  18,30,[VESA_info+17h]
            ; Число плоскостей памяти
            MShowDecByte  19,30,[VESA_info+18h]
            ; Число битов на пиксел
            MShowDecByte   6,70,[VESA_info+19h]
            ; Число банков памяти
            MShowDecByte   7,70,[VESA_info+1Ah]
            ; Тип модели памяти
                MShowDecByte   8,70,[VESA_info+1Bh]
                ; Размер банка памяти
                MShowDecByte   9,70,[VESA_info+1Ch]
                ; Число полных видеостраниц
                MShowDecByte  10,70,[VESA_info+1Dh]
                ; Размер красной компоненты
                MShowDecByte  11,70,[VESA_info+1Fh]
                ; Начальная позиция красной компоненты
                MShowDecByte  12,70,[VESA_info+20h]
                ; Размер зеленой компоненты
                MShowDecByte  13,70,[VESA_info+21h]
                ; Начальная позиция зеленой компоненты
                MShowDecByte  14,70,[VESA_info+22h]
                ; Размер синей компоненты
                MShowDecByte  15,70,[VESA_info+23h]
                ; Начальная позиция синей компоненты
                MShowDecByte  16,70,[VESA_info+24h]
                ; Размер резервной компоненты
                MShowDecByte  17,70,[VESA_info+25h]
                ; Начальная позиция резервной компоненты
                MShowDecByte  18,70,[VESA_info+26h]
                ; Линейный адрес буфера кадра
                MShowHexDWord 19,70,<[dword ptr VESA_info+28h]>
                call    GetChar
                add     BX,2
```

Листинг 2.2 (продолжение)

```
        jmp     @@NextMode
; КОНЕЦ ПРОГРАММЫ
@@End:  ; Установить текстовый режим
        mov     AX,3
        int     10h
        ; Выход в DOS
        mov     AH,4Ch
        int     21h

; СООБЩЕНИЯ ОБ ОШИБКАХ
@@Err:  MFatalError Err1
ENDP VESA_BIOS_Test

;***********************************************
;* ВЫВОД СТРОКИ ИЗ ОБЛАСТИ ДАННЫХ VESA НА ЭКРАН *
;***********************************************
PROC ShowVESAString near
        pusha
        push    ES
; Настраить регистр ES на видеопамять
        mov     AX,0B800h
        mov     ES,AX
        cld
; Вычислить адрес строки в видеопамяти
        mov     DI,6*160+40*2
; Загрузить атрибут цвета в AH
        mov     AH,[TextColorAndBackground]
; Установить указатель на строку
        push    DS
        mov     BX,offset VESA_BIOS + 06h
        mov     SI,[BX]
        mov     BX,[BX+2]
        mov     DS,BX
        ; Вывести строку
@@L1:   lodsb
        and     AL,AL
        jz      @@L2
        stosw
        jmp     @@L1
@@L2:   pop     DS
        pop     ES
        popa
        ret
ENDP ShowVESAString
ENDS

; Подключить процедуры вывода данных на экран
include "list1_02.inc"
; Подключить "математические" процедуры для перевода
; чисел из двоичного кода в десятичный
include "list2_05.inc"

END
```

ВНИМАНИЕ ———————————————————————————————————————

Поскольку программа из листинга 2.2 не переключает видеоконтроллер в заданный режим, а только проверяет его наличие, запускать тест можно на компьютере любой конфигурации. Однако для получения информации о свойствах VESA-режимов нужно, чтобы видеоконтроллер имел встроенную или программную (в виде специальной программы-драйвера) поддержку VESA BIOS.

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

В программе VESA_BIOS_Test (листинг 2.2) используется несколько вспомогательных процедур, не относящихся непосредственно к функциям VESA BIOS, но необходимых для отображения информации на экране и ввода с клавиатуры. Подобные процедуры общего назначения объединены в отдельную группу, оформленную в виде подключаемого файла (include-файла), приведенного в листинге 2.3. Данный файл может быть использован при реализации тестовых программ для других подсистем PC.

Назначение подпрограмм, включенных в листинг 2.3, следующее:

- процедура ShowASCIIChar осуществляет вывод символа в ASCII-коде в заданную позицию экрана;
- процедура ShowHexByte отображает в заданную позицию экрана содержимое регистра AL (байт данных) в шестнадцатеричном коде;
- процедура ShowHexWord отображает в заданную позицию экрана содержимое регистра AX (слово данных) в шестнадцатеричном коде;
- процедура ShowHexDWord отображает в заданную позицию экрана содержимое регистра EAX (двойное слово) в шестнадцатеричном коде;
- процедура ShowBinByte отображает в заданную позицию экрана содержимое регистра AL (байт данных) в двоичном коде;
- процедура ShowBinDWord отображает в заданную позицию экрана содержимое регистра EAX (двойное слово) в двоичном 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 ShowASCIIField служит для вывода на экран текстового поля фиксированного размера;
- 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 ClearScreen осуществляет очистку экрана;
- процедура Beep предназначена для выдачи оператору звукового сигнала (обычно в случае какой-либо ошибки);
- 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 2.3. 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 КОДЕ ASCII        *
;* Подпрограмма выводит содержимое регистра AL в    *
;* коде ASCII в указанную позицию экрана.           *
;* Координаты позиции передаются через глобальные   *
;* переменные ScreenString и ScreenColumn. После    *
;* выполнения операции вывода происходит автомати-  *
;* ческое приращение значений этих переменных.      *
;*****************************************************
PROC ShowASCIIChar near
        pusha
        push    DS
        push    ES
        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 номер колонки (дважды)
        add     AX,[ScreenColumn]
        add     AX,[ScreenColumn]
        ; Переписать результат в индексный регистр
        mov     DI,AX
        pop     AX
        mov     AH,[TextColorAndBackground]
        stosw

; Подготовка для вывода следующих байтов
        ; Перевести текущую позицию на 2 символа влево
        inc     [ScreenColumn]
        ; 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 ShowASCIIChar

;*****************************************************
;*    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
```

Листинг 2.3 (продолжение)

```
        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 16-РАЗРЯДНОГО СЛОВА НА ЭКРАН       *
;*                В ШЕСТНАДЦАТЕРИЧНОМ 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 ShowHexWord NEAR
        xchg    AL,AH
        call    ShowHexByte
        xchg    AL,AH
        call    ShowHexByte
        ret
ENDP ShowHexWord

;*****************************************************
;*          ВЫВОД 32-РАЗРЯДНОГО СЛОВА НА ЭКРАН       *
;*                В ШЕСТНАДЦАТЕРИЧНОМ 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 ShowHexDWord NEAR
        rol     EAX,8
        call    ShowHexByte
        rol     EAX,8
        call    ShowHexByte
        rol     EAX,8
        call    ShowHexByte
        rol     EAX,8
        call    ShowHexByte
        ret
ENDP ShowHexDWord

;*************************************************
;*      ВЫВОД БАЙТА НА ЭКРАН В ДВОИЧНОМ КОДЕ     *
;* Подпрограмма выводит содержимое регистра AL   *
;* в двоичном коде в указанную позицию экрана.   *
;* Координаты позиции передаются через глобальные *
;* переменные ScreenString и ScreenColumn. После  *
;* выполнения операции вывода происходит автома-  *
;* тическое приращение значений этих переменных.  *
;*************************************************
PROC ShowBinByte near
        pusha
        push    DS
        push    ES
        ; Копируем отображаемый байт в BL
        mov     BL,AL
        mov     AX,[CS:MainDataSeg]
        mov     DS,AX
        cld
        ; Загрузить адрес "текстовой" видеопамяти 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 дважды номер колонки
        add     AX,[ScreenColumn]
        add     AX,[ScreenColumn]
        ; Переписать результат в индексный регистр
        mov     DI,AX

; Отобразить разряды числа (начиная со старшего)
        mov     AH,[TextColorAndBackground]
        mov     CX,8    ;счетчик разрядов
@@L0:   mov     AL,'0'
        ; Выделить очередной разряд числа
        rol     BL,1
        jnc     @@L1
        mov     AL,'1'
        ; Вывести разряд числа на экран
@@L1:   stosw
        loop    @@L0

; Подготовка для вывода следующих байтов
        ; Перевести текущую позицию на 8 символов влево
```

Листинг 2.3 (продолжение)

```
        add     [ScreenColumn],8
        ; Проверить пересечение правой границы экрана
        cmp     [ScreenColumn],80
        jb      @@End
        ; Если достигнута правая граница экрана -
        ; перейти на следующую строку
        sub     [ScreenColumn],80
        inc     [ScreenString]
; Конец подпрограммы
@@End:  pop     ES
        pop     DS
        popa
        ret
ENDP ShowBinByte
;*****************************************************
;* ВЫВОД 16-РАЗРЯДНОГО СЛОВА НА ЭКРАН В ДВОИЧНОМ 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 ShowBinWord NEAR
        rol     AX,8
        call    ShowBinByte
        inc     [ScreenColumn]
        rol     AX,8
        call    ShowBinByte
        ret
ENDP ShowBinWord

;*****************************************************
;* ВЫВОД 32-РАЗРЯДНОГО СЛОВА НА ЭКРАН В ДВОИЧНОМ 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 ShowBinDWord NEAR
        rol     EAX,8
        call    ShowBinByte
        inc     [ScreenColumn]
        rol     EAX,8
        call    ShowBinByte
        inc     [ScreenColumn]
        rol     EAX,8
        call    ShowBinByte
        inc     [ScreenColumn]
        rol     EAX,8
        call    ShowBinByte
        ret
ENDP ShowBinDWord

;*****************************************************
;*          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;    *
```

Листинг 2.3 (продолжение)

```
;* 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:
```

Листинг 2.3 (продолжение)

```
        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
```

2.2.5.2. Мнемоническое обозначение управляющих клавиш и цветовых кодов

Приведенная в листинге 2.3 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. 2.17, традиционно имеют как ASCII-коды, так и скан-коды, причем значения этих кодов не совпадают. 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).

Таблица 2.17. 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

В листинге 2.4. 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 1.3 включены мнемонические обозначения цветовых оттенков, которые будут применяться в программах, работающих в 16-цветных и 256-цветных режимах

Листинг 2.4. 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
```

Листинг 2.4.(продолжение)

```
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
```

2.2.5.3. Макрокоманды для вывода данных на экран.

Листинг 2.5 содержит набор макрокоманд, делающих более наглядными и компактными участки программного кода, в которых осуществляются операции ввода-вывода.

Листинг 2.5. Макрокоманды для вывода данных на экран (подключаемый файл list1_04.inc) .

```
; МАКРОКОМАНДЫ ТЕКСТОВОГО РЕЖИМА
;*************************************
;*    ВЫВОД БАЙТА В ДВОИЧНОМ КОДЕ    *
;* Параметры:                        *
;* SString - номер строки экрана;    *
;* SColumn - номер колонки экрана;   *
;* BData - отображаемый байт данных. *
;*************************************
MACRO MShowBinByte SString,SColumn,BData
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     AL,BData
        call    ShowBinByte
ENDM
;*********************************************
;* ВЫВОД 16-РАЗРЯДНОГО СЛОВА В ДВОИЧНОМ КОДЕ *
;* Параметры:                                *
;* SString - номер строки экрана;            *
;* SColumn - номер колонки экрана;           *
;* WData - отображаемое слово данных.        *
;*********************************************
MACRO MShowBinWord SString,SColumn,WData
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     AX,WData
        call    ShowBinWord
ENDM
;*********************************************
;* ВЫВОД 32-РАЗРЯДНОГО СЛОВА В ДВОИЧНОМ КОДЕ *
;* Параметры:                                *
;* SString - номер строки экрана;            *
;* SColumn - номер колонки экрана;           *
;* DData - отображаемое слово данных.        *
;*********************************************
MACRO MShowBinDWord SString,SColumn,DData
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     EAX,DData
        call    ShowBinDWord
ENDM
;****************************************
;* 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 16-РАЗРЯДНОГО СЛОВА В ШЕСТНАДЦАТЕРИЧНОМ КОДЕ *
;* Параметры:                                        *
;* SString - номер строки экрана;                    *
;* SColumn - номер колонки экрана;                   *
;* WData - отображаемое слово данных.                *
;*****************************************************
MACRO MShowHexWord SString,SColumn,WData
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     AX,WData
        call    ShowHexWord
ENDM
;*****************************************************
;* ВЫВОД 32-РАЗРЯДНОГО СЛОВА В ШЕСТНАДЦАТЕРИЧНОМ КОДЕ *
;* Параметры:                                        *
;* SString - номер строки экрана;                    *
;* SColumn - номер колонки экрана;                   *
;* DData - отображаемое слово данных.                *
;*****************************************************
MACRO MShowHexDWord SString,SColumn,DData
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     EAX,DData
        call    ShowHexDWord
ENDM

;*************************************
;*   ВЫВОД БАЙТА В ДЕСЯТИЧНОМ КОДЕ   *
;* Параметры:                        *
;* SString - номер строки экрана;    *
;* SColumn - номер колонки экрана;   *
;* BData - отображаемый байт данных. *
;*************************************
MACRO MShowDecByte SString,SColumn,BData
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     AL,BData
        call    ShowDecByte
ENDM
```

Листинг 2.5 (продолжение)

```
;*********************************************
;* ВЫВОД 16-РАЗРЯДНОГО СЛОВА В ДЕСЯТИЧНОМ КОДЕ *
;* Параметры:                                  *
;* SString - номер строки экрана;              *
;* SColumn - номер колонки экрана;             *
;* WData - отображаемое слово данных.          *
;*********************************************
MACRO MShowDecWord SString,SColumn,WData
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     AX,WData
        call    ShowDecWord
ENDM
;*********************************************
;* ВЫВОД 32-РАЗРЯДНОГО СЛОВА В ДЕСЯТИЧНОМ КОДЕ *
;* Параметры:                                  *
;* SString - номер строки экрана;              *
;* SColumn - номер колонки экрана;             *
;* DData - отображаемое слово данных.          *
;*********************************************
MACRO MShowDecDWord SString,SColumn,DData
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     EAX,DData
        call    ShowDecDWord
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 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 СООБЩЕНИЯ ОБ ОШИБКЕ       *
;* Параметры:                            *
;* TOffset - смещение строки сообщения. *
;*****************************************
MACRO MFatalError TOffset
        mov     SI,offset TOffset
        call    FatalError
ENDM
; МАКРОКОМАНДЫ ГРАФИЧЕСКОГО РЕЖИМА
;*****************************
;*   ВЫВОД ТЕКСТОВОЙ СТРОКИ   *
;* Параметры:                 *
;* TOffset - смещение строки. *
;*****************************
MACRO MGShowString TOffset
        mov     SI,offset TOffset
        call    GShowString
ENDM
;*********************************************
;*        ВЫВОД ОДНОЦВЕТНОГО ТЕКСТА          *
;* Параметры:                                 *
;* TStrings - количество строк в тексте;      *
;* TOffset - смещение первой строки текста. *
;*********************************************
MACRO MGShowText TStrings,TOffset
        mov     CX,TStrings
        mov     SI,offset TOffset
        call    GShowText
ENDM
```

2.2.5.4. Перевод чисел из десятичного кода в двоичный и наоборот.

Перевод чисел из одной системы счисления в другую 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), однако об этом не упоминается ни в одном учебнике!

Математический сопроцессор понимает семь различных форматов представления чисел (параметры которых кратко описаны в табл. 2.18) 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. 2.19.

Таблица 2.18. 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 $(10^{18}-1)$

Таблица 2.19. 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 Целые числа в двоичном коде можно хранить как в целочисленных, так и в вещественных форматах.

Рассмотрим порядок перевода целого десятичного числа, записанного в ASCII-коде, в двоичный код.

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.

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. Рассмотрим порядок действий.

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 К, умножение выполняется н, 10^{K-N} , где N — количество знаков мантиссы числа после запятой.

При переводе вещественного числа (X) из двоичного кода в десятичный (D) также приходится выполнять дополнительные преобразования. После выполнения перевода число должно состоять из нормализованной мантиссы (M) и порядка (P):

$$D=M\times10^{P}$$

Порядок выполнения операций следующий:

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 $\log_2 X$ и умножение результата на Y). Десятичный логарифм вычисляется по формуле:

$$\log_{10}X = \log_{10}2\times\log_2 X$$

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 2.6. Назначение подпрограмм, включенных в листинг 2.6, 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 String_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 Doub1eF1oat_to_String 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 DoubleFloat_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 2.6. 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
```

Листинг 2.6 (продолжение)

```
        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
        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
```

Листинг 2.6 (продолжение)

```
; 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]
        ; 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?
```

Листинг 2.6 (продолжение)

```
        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
@@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]
```

Листинг 2.6 (продолжение)

```
; Нормализовать десятичную мантиссу
        fninit                  ;сброс сопроцессора
        fld     [Data_Abs]      ;загрузить модуль числа
        mov     BX,[MaxPositions]
        sub     BX,[Data_Log10]
        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],80l
        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
```

Листинг 2.6 (продолжение)

```
        ; (сдвигаем младшие 4 байта)
        shl     [dword ptr Data_BCD],4
        ; Загружаем следующий символ в AL
        lodsb
        loop    @@ASCIItoBCDConversion
        ; 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
```

Листинг 2.6 (продолжение)

```
        pop     ES
        popa
        ret
ENDP ShowDataString

;*************************************************
;*          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
```

Листинг 2.6 (продолжение)

```
        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
        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.

2.2.5.5. Программа тестирования функций перевода целых и вещественных чисел.

Программа MathFunctionsTest, приведенная в листинге 2.7, предназначена для тестирования функций перевода целых и вещественных чисел. Тестирование выполняется по методу *обратной связи:* введенное оператором число переводится в двоичный код, а затем из двоичного кода переводится обратно в десятичный и отображается на экран.

Листинг 2.7. Тестирование подпрограмм перевода вещественных чисел

```
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
Txt1 DB LIGHTMAGENTA,0,18
     DB "Тестирование подпрограмм перевода чисел",0
     DB LIGHTGREEN,9,0,"Введите целое число:",0
Txt2 DB LIGHTGREEN,10,0
     DB "Число в шестнадцатеричном коде:",0
     DB LIGHTGREEN,11,0
     DB "Обратный перевод в десятичный код:",0
     DB LIGHTGREEN,13,0
     DB "Введите вещественное число:",0
Txt3 DB LIGHTGREEN,14,0,"Число в научном формате:",0
     DB LIGHTGREEN,15,0,"Число в обычном формате:",0
     DB YELLOW,24,29,"Нажмите любую клавишу",0
ENDS

CODESEG
```

Листинг 2.7 (продолжение)

```
;*****************************
;* Основной модуль программы *
;*****************************
PROC MathFunctionsTest
        mov     AX,DGROUP
        mov     DS,AX
        mov     [CS:MainDataSeg],AX
; Установить текстовый режим и очистить экран
        mov     AX,3
        int     10h
; Вывести заголовок
        MShowColorText 2,Txt1
; Задать цвет символов числа
        mov     [TextColorAndBackground],LIGHTGREY

; Ввести и вывести целое число
        ; Ввести целое число
        mov     [ScreenString],9
        mov     [ScreenColumn],21
        call    GetInteger
        ; Вывести комментарии
        MShowColorText 3,Txt2
        ; Вывести число в шестнадцатеричном коде
        MShowHexDWord 10,32,[Data_Int32]
        ; Вывести число в десятичном коде
        mov     [ScreenString],11
        mov     [ScreenColumn],35
        call    Int32_to_String
        call    ShowDataString
; Ввести и вывести вещественное число
        ; Ввести целое число
        mov     [ScreenString],13
        mov     [ScreenColumn],28
        call    GetFloat
        ; Вывести комментарии
        MShowColorText 3,Txt3
        ; Вывести число в научном формате
        mov     [ScreenString],14
        mov     [ScreenColumn],25
        call    DoubleFloat_to_ExpForm
        call    ShowDataString
        ; Вывести число в обычном формате
        mov     [ScreenString],15
        mov     [ScreenColumn],25
        call    DoubleFloat_to_String
        call    ShowDataString

        mov     [ScreenString],25
        mov     [ScreenColumn],0
        call    SetCursorPosition
        call    GetChar

; Переустановить текстовый режим и очистить экран
        mov     AX,3
        int     10h
; Выход в DOS
        mov     AH,4Ch
        int     21h
ENDP MathFunctionsTest
ENDS
; Подключить процедуры вывода данных на экран
include "list1_02.inc"
; Подключить процедуры перевода чисел
include "list2_05.inc"

END
```

Контрольные вопросы.

1. Когда обычно используют подфункции 00h и 01h функции 4Fh прерывания Int 10h VESA BIOS?
2. Охарактеризуйте программу, приведенную в листинге 2.1.
3. Охарактеризуйте процедуры, представленные в листинге 2.2.
4. Охарактеризуйте содержимое листинга 2.3.
5. Перечислите основные макросы, представленные в листинге 2.4.
6. Какие форматы данных понимает математический сопроцессор?
7. Перечислите команды передачи данных, которые можно использовать для перевода чисел в различные форматы.
8. Каков порядок перевода десятичного числа, записанного в ASCII коде в двоичный код и наоборот?
9. Каков порядок перевода вещественных чисел из десятичного формата в двоичный?
10. Охарактеризуйте порядок перевода вещественного числа, записанного в показательном (научном) формате в двоичный код.
11. Как сопроцессор вычисляет логарифмы?
12. Охарактеризуйте назначение подпрограмм, представленных в листинге 2.5.
13. Охарактеризуйте программу, представленную в листинге 2.6.
14. Охарактеризуйте программу, представленную в листинге 2.7.

В этом разделе использованы материалы из [5].

2.2.6. Особенности работы с видеоадаптером в различных режимах.

2.2.6.1. Особенности работы в текстовом режиме

При включении IBM-совместимого компьютера видеоконтроллер обычно начинает работу с текстового 16-цветного режима с разрешением 80×25 символов (код режима 03h). В этом режиме для работы с видеопамятью выделено окно размером 32 Кбайт в первом мегабайте адресного пространства процессора (начальный линейный адрес окна B8000h, см раздел 2.1.1.1.3).

Текстовый режим до сих пор широко применяется, так как для решения многих задач он является вполне достаточным, отличается простотой, надежностью и невысокими требованиями к аппаратуре компьютера. Основные операции в текстовом режиме: вывод символов, управление курсором и загрузка шрифта.

Видеопамять в текстовом режиме с точки зрения процессора организована следующим образом: на каждый символ приходится по два байта информации, причем первый байт хранит ASCII-код символа, а второй - цвет символа и цвет фона знакоместа этого символа, как показано на рис. 2.23.

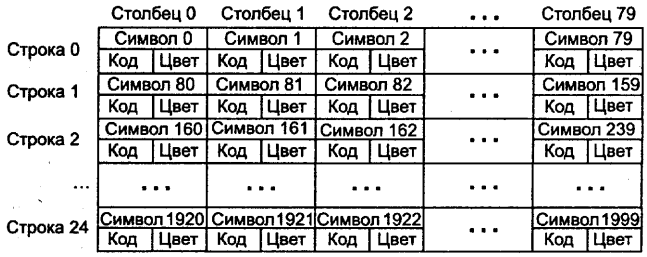


Рис. 2.23. Отображение видеопамяти на экран в текстовом 16-цветном режиме с разрешением 80×25 символов

Для вывода информации в видеопамять обычно используется дополнительный сегментный регистр данных ES, в который перед началом записи в видеопамять нужно занести число, равное абсолютному начальному адресу буфера, поделенному на 16.

Чтобы вывести символ в заданное знакоместо, нужно помножить номер строки знакоместа на 160 (длину строки в байтах) и прибавить номер столбца знакоместа, после чего записать результат в индексный регистр. Далее в соответствующий байт заносится с помощью косвенной адресации (относительно сегмента ES и избранного индексного регистра) ASCII-код символа. При необходимости значение индексного регистра инкрементируется, и в следующий байт записывается код цвета символа и фона.

Формат кодирования цвета символа показан на рис. 2.24 (см. также рис. 2.15 в разделе 2.1.12.1).

• биты 0-3 — цвет символа;
• биты 4-6 — цвет фона;
• бит 7 — мерцание (0 — обычный текст, 1 — мигающий текст).

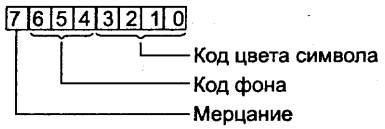


Рис. 2.24. Формат байта описания цвета символа и фона в текстовом режиме

Расшифровка цветовых кодов дана в табл. 2.20 и более подробно в табл. 1.3 раздела 1.3.2 (для фона возможна установка только «темных» цветов с кодами от 0 до 7).

Таблица 2.20. Расшифровка цветовых кодов для текстового режима

Код	Цвет	Ко	Цвет
0	Черный	8	Серый
1	Синий	9	Голубой
2	Зеленый	10	Салатный
3	Бирюзовый	11	Светло-бирюзовый
4	Красный	12	Розовый
5	Фиолетовый	13	Светло-фиолетовый
6	Коричневый	14	Желтый
7	Светло-серый	15	Белый

Приведем пример кода программы, в котором осуществляется вывод символа # в 15-й позиции третьей строки экрана, желтым цветом по синему фону.

```
;Загрузить в ES адрес текстовой видеопамяти
        mov     AX,0B800h
        mov     ES,AX
;Умножить номер строки на длину строки в байтах
        mov     AL,160
        mov     AH,3
        mul     AH
;Прибавить к произведению номер колонки
        add     AX,15
;Переписать полученное смещение
;в индексный регистр
        mov     BX,AX
;Записать код символа в видеопамять
        mov     byte ptr ES:[BX],'#'
;Записать код цвета в следующий байт
        inc     BX
        mov     byte ptr ES:[BX],01Eh
```

Кроме вывода символов, в текстовом режиме часто применяются еще два вида операций — перемещение курсора и переключение видеостраниц. Управление курсором осуществляется при помощи регистров видеоконтроллера или функций BIOS. Установить размер прямоугольника текстового курсора можно при помощи регистров начальной и конечной линий курсора (см. табл. 2.3) или при помощи функции 01h прерывания Int 10h (см. раздел 2.1.2.2). Перемещение курсора по тексту производится путем записи значения смещения курсора относительно начала видеопамяти в регистры старшего и младшего байта адреса курсора (см. табл. 2.3). Позиционировать курсор можно и с помощью функции 02h прерывания Int 10h, но координаты курсора при вызове прерывания задаются в виде номера строки и колонки относительно начала видеостраницы (см. раздел 2.1.2.2). Если необходимо удалить курсор с экрана, то это можно сделать несколькими способами, наиболее универсальным из которых является перемещение курсора за нижнюю границу экрана (на 25-ю строку).

Переключение видеостраниц осуществляется путем записи смещения левого верхнего угла видеостраницы относительно начала видеопамяти в регистры старшего и младшего байт начального адреса. Эту операцию можно также выполнить при помощи функции 05h прерывания Int 10h. Видеоконтроллер в текстовом режиме обеспечивает 8 видеостраниц, но используется обычно только основная (нулевая) страница. Между тем, дополнительные страницы позволяют выдавать сообщения оператору, не разрушая текст на основной странице. Их можно применять для отображения окна подсказки, сообщений об ошибках, вспомогательных меню, а также для вывода дампа памяти и регистров процессора при отладке программ.

2.2.6.2.Работа в современных графических режимах

Современные видеоконтроллеры отличаются от VGA-контроллеров тем, что обеспечивают работу с высокими разрешениями и позволяют использовать линейную адресацию видеопамяти. Ниже рассматриваются видеорежимы, реализующие обе указанные возможности.

2.2.6.2.1. Организация видеопамяти в 256-цветных режимах

Прототипом этой группы послужил режим VGA с кодом 13h. В 256-цветных режимах каждой точке изображения на экране монитора соответствует один байт видеопамяти, в который записывается код цвета точки. Этот код не используется непосредственно, а служит индексом в специальном массиве, содержащем 256 строк по 3 элемента — таблице цветов ЦАП. Каждый из трех элементов строки таблицы задает интенсивность одного из основных цветов электронно-лучевой трубки (красного, зеленого или синего). Значения интенсивностей, выбранные из строки, соответствующей хранящемуся в видеопамяти коду, поступают в ЦАП.

Видеопамять отображается на экран слева направо и сверху вниз, как показано на рис. 2.25 (обратите внимание: экранная ось Y при этом направлена вниз).

	Столбец 0	Столбец 1	Столбец 2		Столбец 639
Строка 1	Байт 0	Байт 1	Байт 2	...	Байт 639
Строка 2	Байт 640	Байт 641	Байт 642	...	Байт 1279
Строка 3	Байт 1280	Байт 1281	Байт 1282	...	Байт 1283
	...	...	...	...	...
Строка 479	Байт 306560	Байт 306561	Байт 306562	...	Байт 307199

Рис. 2.25. Отображение видеопамяти на экран в 256-цветном режиме с разрешением 640×480 точек

2.2.6.2.2. Организация видеопамяти в режимах типа DirectDraw

В режимах группы DirectDraw (HiColor и TrueColor) информации поступает на цифро-аналоговые преобразователи непосредственно из видеопамяти. Соответственно, красная, зеленая и синяя составляющие цвета точки представлены отдельными полями в выделенной для хранения точки области видеопамяти (от 2 до 4 байт на точку).

В режимах HiColor точка кодируется 16-разрядным словом, причем существует два варианта

представления цвета, показанные на рис. 2.26: HiColorl5 (формат 1:5:5:5) и HiColorl6 (формат 5:6:5). Знаком X на рисунке обозначена зарезервированная (неотображаемая) область данных. Из видеопамяти на экран информация отображается слева направо и сверху вниз.

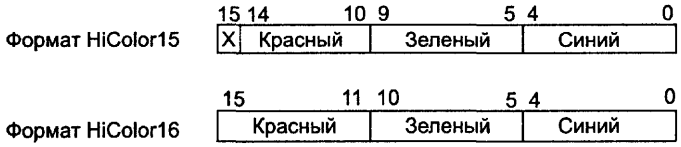


Рис. 2.26. Форматы данных группы HiColor

В режимах TrueColor для хранения каждого компонента цвета точки выделено по одному байту видеопамяти. Существует два формата представления данных, показанные на рис. 4.7: трехбайтный TrueColor24 и четырехбайтный TrueColor32. Дополнительный байт (на рисунке он обозначен символом X) добавлен в режим TrueColor32 для выравнивания — на экране он не отображается. Дело в том, что процессор может передавать данные видеоконтроллеру только байтами, 16-разрядными и 32-разрядными словами. Чтобы не задеть соседние точки, в 24-битном режиме информацию приходится пересылать по байтам (каждый компонент цвета — отдельно), что в три раза снижает скорость передачи данных. Поскольку такая потеря темпа недопустима во многих приложениях (например, в играх), разработчики решили пожертвовать памятью (в режиме TrueColor32 четверть объема памяти расходуется впустую). Это стандартный прием обмена памяти на производительность, который часто используют инженеры.

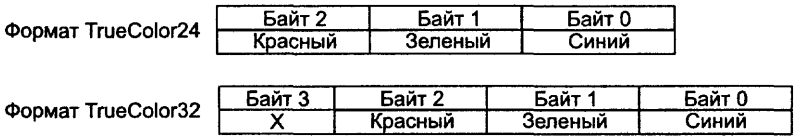


Рис. 2.27. Форматы данных группы TrueColor

Следует отметить, что на рис. 2.27 расположение информационных разрядов показано в порядке, принятом в литературе по программированию: старший разряд слева, младший справа. Однако при отображении памяти на экран, как уже было указано, традиционно принят порядок слева направо и сверху вниз. Например, в режиме TrueColor32 1024×768 память отображается на экран в порядке, показанном на рис. 2.28.

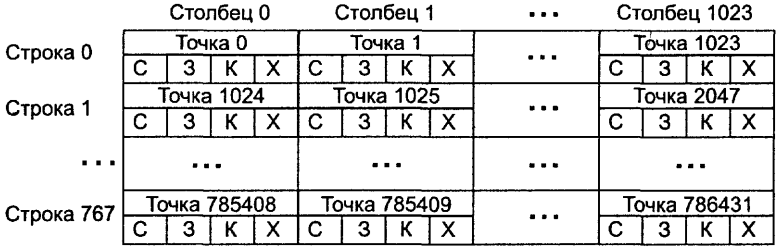


Рис. 2.28. Отображение видеопамяти на экран в режиме TrueColor32 с разрешением 1024×768 точек

В общем, можно сказать, что развитие видеоконтроллеров направлено в сторону максимального упрощения (для программистов) работы с видеопамятью, даже за счет значительного ее перерасхода (рис. 2.29).

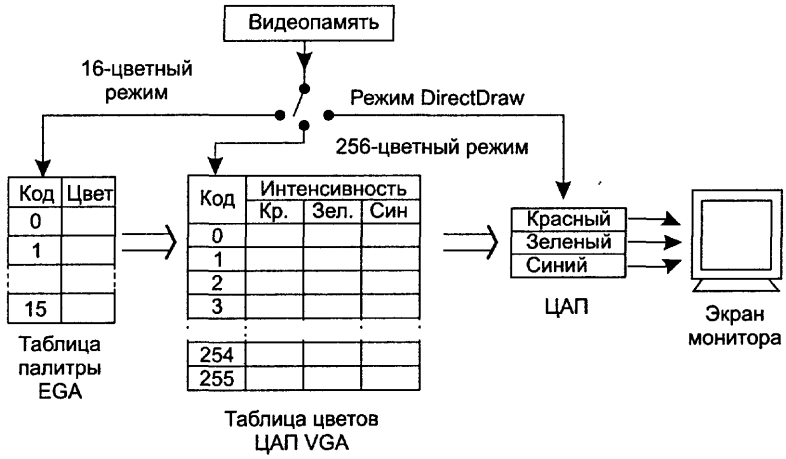


Рис. 2.29. Прохождение информации о цвете пиксела от видеопамяти до экрана монитора в различных графических режимах

Режимы DirectDraw реализуют прямое кодирование цвета без всяких промежуточных преобразований, а последний вариант (True-Color) обеспечивает простой доступ к каждому из трех основных цветовых компонентов. Такая организация видеопамяти минимизирует вычисления, ускоряя расчет освещенности объектов при использовании трехмерной графики, но может заметно снижать скорость вывода двухмерного изображения (текста, графика, схемы, чертежа) из-за резкого увеличения объема передаваемой информации.

2.2.6.2.3. Режимы адресации и распределение видеопамяти

Как было указано выше, существует два основных метода адресации видеопамяти в графических режимах: устаревший сегментный и современный линейный. В режиме сегментации видеопамять поделена на 64-килобайтные кусочки — окна, причем в каждый момент времени для работы доступен только один такой сегмент. Доступ ко всем сегментам осуществляется через 64-килобайтный участок адресного пространства процессоров x86 с абсолютным адресом A0000h. Режим сегментации имеет два серьезных недостатка, заметно снижающих скорость работы видеосистемы:

- при выводе изображения необходимо постоянно контролировать пересечение границ сегментов, что требует выполнения процессором ряда дополнительных команд;
- при пересечении границ сегментов приходится выполнять их переключение посредством программных процедур VESA BIOS, поскольку механизм переключения при помощи регистров видеоконтроллера так и не был стандартизирован.

Указанные недостатки режима сегментации практически незаметны при выводе статических изображений (рисунков, фотографий, Карт, чертежей), однако весьма заметно проявляют себя при использовании анимации.

Линейная адресация обеспечивает гораздо более простой и удобный режим работы — вся видеопамять представляет собой с точки зрения процессора единый (непрерывный) участок адресного пространства. В линейном режиме нет никаких искусственных границ и, соответственно, не нужно тратить время на контроль их пересечения. Основным недостатком метода линейной адресации является невозможность его использования в реальном режиме DOS и режиме виртуальных процессоров. Линейная адресация доступна только в защищенном режиме и режиме линейной адресации данных.

Видеорежим и способ адресации видеопамяти задаются одновременно — кодом номера режима при вызове функции переключения видеорежимов VESA BIOS (прерывание Int 10h, функция 4Fh, подфункция 02h). Прежде чем устанавливать режим, желательно убедиться, что видеоконтроллер его поддерживает, для чего нужно использовать подфункцию 01h функции 4Fh прерывания Int 10h. (см.листинг 2.2).

Распределением видеопамяти управляет программист. При работе со статическим изображением обычно используется только первая видеостраница, а организация памяти сохраняется в том виде, который устанавливается по умолчанию при включении видеорежима (логическая длина видеостроки в пикселах равна физическому разрешению экрана).

При работе с анимацией часто используют две видеостраницы, а память реорганизуют с целью максимального ускорения вывода изображения. Пока процессор перерисовывает изображение в одной области памяти, видеоконтроллер осуществляет вывод предыдущего кадра на экран из другой области (рис. 2.30). Для реорганизации памяти применяются подфункции 06h и 07h функции 4Fh прерывания Int 10h. Подфункция 06h позволяет сделать логическую длину строки больше физической и используется для выравнивания длины строки на 2^N. Выравнивание позволяет ускорить вычисление координат, упростить контроль границ и создать невидимую вертикальную защитную полосу. Подфункция 07h позволяет выбирать положение страниц в видеопамяти и переключать страницы.

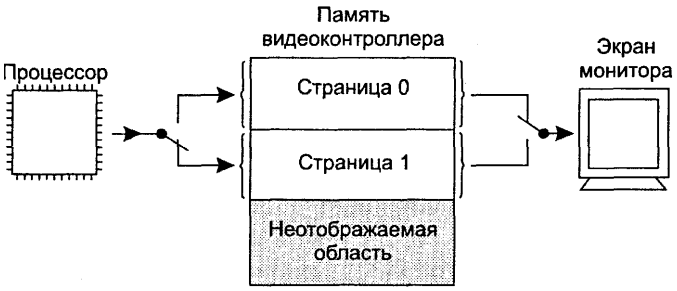


Рис. 2.30. Распределение видеопамяти при использовании механизма переключения страниц

2.2.6.2.4. Рисование линий при линейной адресации памяти

Сегментация видеопамяти существенно замедляет рисование линий в режимах с высоким разрешением, особенно при использовании для переключения страниц прерываний VESA BIOS: при рисовании прямых под большим углом к горизонтали переключение приходится осуществлять через каждые несколько десятков точек. Переключение через регистры видеоконтроллера могут выполнять только Windows-

драйверы, написанные программистами фирмы-изготовителя, поскольку документация контроллера доступна только для них. Драйверы для других операционных систем обычно не предоставляются, и, следовательно, вывод линий осуществляется в режиме сегментации очень медленно.

Линейная адресация позволяет избежать переключения страниц. Модель памяти, как было показано в предыдущих разделах, при линейной адресации очень простая: видеоконтроллер воспринимает свою оперативную память как прямоугольный массив (N строк, M столбцов), а процессор видит память контроллера как линейный массив размером N×M×K байт, где K — число байтов на точку изображения.

Рисование линии базируется на операции вывода точки. Вывод точки в современных видеорежимах заключается в том, что процессор записывает по соответствующему адресу видеопамяти код цвета точки, например, при помощи команды MOV. При использовании линейной адресации для обращения к памяти можно применять в качестве индексного любой из регистров общего назначения, а в качестве сегментного — регистр данных, настроенный на видеопамять (в защищенном режиме) или на линейную адресацию. Например, если для включения линейной адресации применялась процедура, описанная ранее в разделе «Функции VESA BIOS», вывод точки белого цвета в 256-цветном режиме может быть осуществлен следующей командой:

```
mov     [byte ptr GS:EDI],0Fh
```

Здесь 0Fh — код белого цвета, а регистр EDI — абсолютный адрес точки, то есть сумма начального адреса видеопамяти и смещения точки от начала видеопамяти.

В режиме HiColor при выполнении такой же операции нужно записать не байт, а 16-иазрядное слово:

```
mov     [word ptr GS:EDI],0FFFFh
```

В режиме TrueColor32 вывод точки белого цвета заключается в записи соответствующего 32-разрядного кода:

```
mov     [word ptr GS:EDI],0FFFFFFh
```

Процесс рисования горизонтальной линии заключается в циклическом выводе кода цвета линии в ячейки памяти, соответствующие соседним точкам (после вывода каждой точки к значению индексного регистра прибавляется количество байтов на точку). Для 256-цветного режима указанный процесс выглядит так:

```
; Установить цвет линии
        mov     AL,[LineColor]
; Записать адрес начальной точки в индексный регистр
        mov     EDI,[StartPixelAddress]
; Задать длину линии в пикселах
        mov     CX,[LineLength]
@@L:    mov     [GS:EDI],AL ;нарисовать пиксел
        inc     EDI
        loop    @@L
```

В режиме TrueColor32 та же операция выполняется следующим образом:

```
; Установить цвет линии
        mov     EAX, [LineColor]
; Записать адрес начальной точки в индексный регистр
        mov     EDI,[StartPixelAddress]
; Задать длину линии в пикселах
        mov     CX,[LineLength]
@@L:    mov     [GS:EDI],EAX ;нарисовать пиксел
        add     EDI,4
        loop    @@L
```

Процесс рисования вертикальной линии отличается от описанного выше тем, что для перехода на одну точку вниз к содержимому индексного регистра прибавляется длина логической строки в байтах (которая вычисляется как произведение длины строки в пикселах на размер пиксела в байтах). Например, для 256-цветного режима цикл рисования выглядит так:

```
; Установить цвет линии
        mov     AL,[LineColor]
; Записать адрес начальной точки в индексный регистр
        mov     EDI,[StartPixelAddress]
; Задать длину линии в пикселах
        mov     CX,[LineLength]
@@L:    mov     [GS:EDI],AL  ;нарисовать пиксел
        ; Прибавить длину строки в байтах
        add     EDI,LogicalStringLength
        loop    @@L
```

Так же легко реализуется алгоритм рисования наклонных линий, имеющих постоянный размер шага, то есть состоящих из отрезков одинаковой длины. Например, при выводе диагональной линии под углом 45

градусов одновременно выполняется перемещение на одну точку по горизонтали и по вертикали.

В режиме HiColor для горизонтальной линии получается следующий цикл:

```
; Установить цвет линии
        mov     AX,[LineColor]
; Записать адрес начальной точки в индексный регистр
        mov     EDI,[StartPixelAddress]
; Задать длину линии в пикселах
        mov     CX,[LineLength]
@@L:    mov     [GS:EDI],AX
        ; Прибавить сумму длины строки
        ; и размера пиксела в байтах
        add     EDI,(LogicalStringLength+1)*2
        loop    @@L
```

Для вывода произвольных линий можно использовать целочисленный алгоритм Брезенхема, описанный во всех книгах по компьютерной графике (см., например, [12], раздел 7.3.2). Наиболее простое описание алгоритмов рисования линий, окружностей и эллипсов приведено в книге [13], но перевод с языка С на ассемблер теперь выполняется гораздо проще, чем десять лет назад, так как уже не нужно манипулировать с масками точек (основная «головная боль» в 16-цветных режимах), а можно использовать 32-разрядные регистры процессора и линейную адресацию видеопамяти. Листинги примеров рисования линий, в том числе по алгоритму Брезенхема, будут показаны ниже — вместе с выводом текста.

2.2.6.2.5. Вывод текста и статических изображений в графических режимах

Функции BIOS VGA не обеспечивают приемлемой скорости вывода символов на экран, поэтому в графических режимах текст выводится напрямую в видеопамять (растровое разложение символа передается по точкам).

Наиболее простым способом вывода на экран текста и статических изображений является метод масок. Вообще говоря, для создания изображения объекта необходимо два массива: массив, содержащий рисунок объекта, и массив, содержащий маску прозрачности, то есть указание, какие точки рисунка должны отображаться на экране, а в каких надлежит сохранить фон изображения. Однако с целью экономии памяти оба массива объединяют и именуют полученный результат маской объекта (рис. 2.31). Таким образом, маска представляет собой растровое изображение объекта, совмещенное с шаблоном прозрачности.

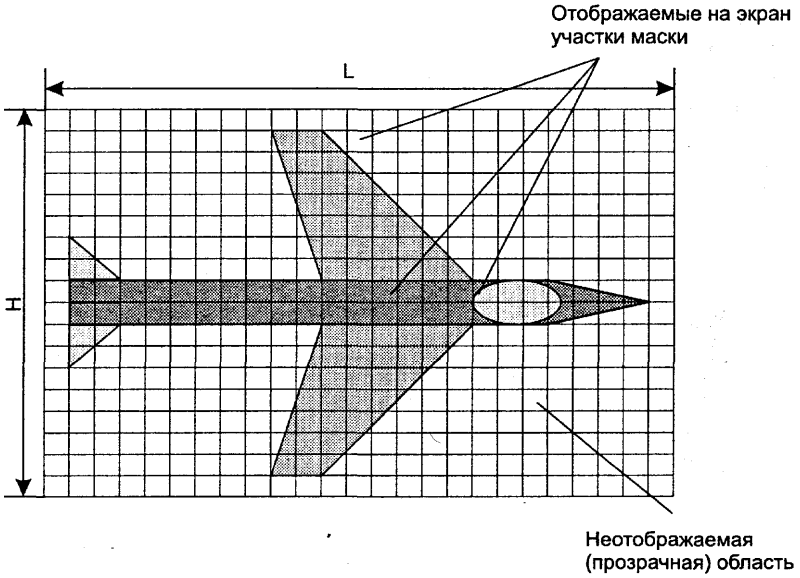


Рис. 2.31. Маска объекта «самолет»

Дело в том, что изображение объекта обычно хранится в виде матрицы, то есть имеет прямоугольную форму, но форма самого объекта редко является прямоугольной. Шаблон прозрачности показывает, какие части растрового изображения относятся к объекту и должны быть отображены на экране, а какие — не относятся и не отображаются. При совмещении шаблона прозрачности с изображением объекта в качестве признака прозрачности выбирают один из цветовых кодов, чаще всего — ноль.

Вывод изображения объекта по методу маски в простейшем случае заключается в копировании маски из оперативной памяти в соответствующий этому изображению участок видеопамяти. Те точки маски, значение кода которых соответствует признаку прозрачности, не копируются, а пропускаются — на их месте в видеопамяти будут сохраняться точки фона.

При выводе текста (или небольших статических изображений) часто бывает целесообразно применять

разметку экрана на знакоместа (как в текстовом режиме), так как это позволяет упростить алгоритмы вывода на экран и ускорить выполнение соответствующий операций. Полный размер знакоместа может быть больше размера маски символа, если между символами вставить пустые строки и колонки — читабельность текста в результате несколько улучшается, но снижается емкость экрана в знакоместах (рис. 2.32). Однако обычно символы выводятся вплотную друг к другу, без дополнительных промежутков.

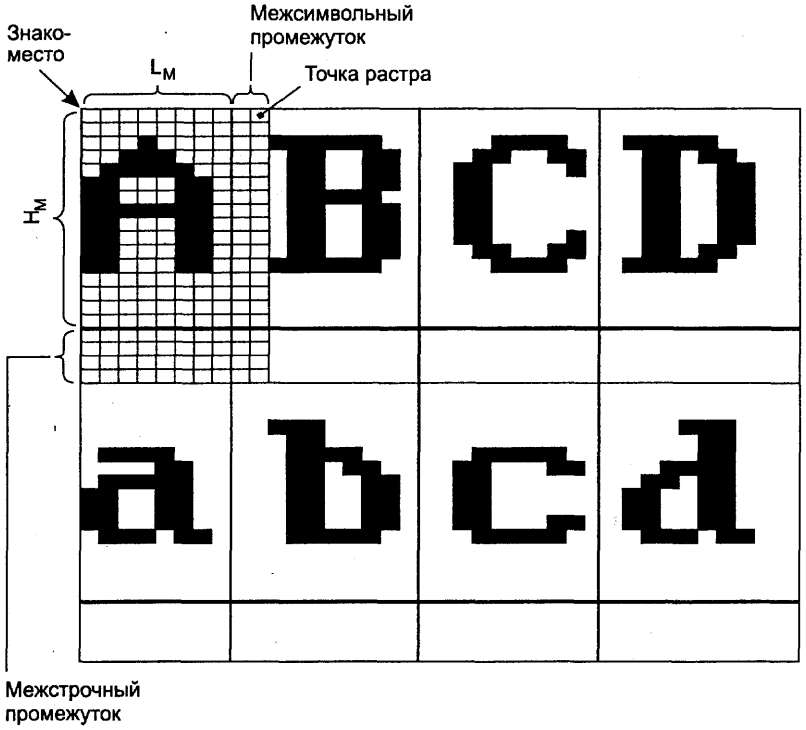


Рис. 2.32. Разметка экрана на знакоместа при выводе текста в графическом режиме (L_M, — ширина маски символа, H_M, — высота маски)

Если необходимо отображать одинаковые по форме объекты различного цвета, например, выводить текстовые символы различными цветами, то маска применяется для кодирования не цвета, а яркости точек изображения. В этом случае процессор не копирует маску прямо в видеопамять, а выполняет вначале сложение базового кода цвета со значением яркости (в 256-цветных режимах) или распределение кода яркости по заданным цветовым компонентам RGB (в режимах DirectDraw).

Рассмотрим теперь примеры выполнения простых операций в различных графических режимах. В листинге 2.8 собраны универсальные процедуры, предназначенные для установки графических VESA-режимов, вывода текстовой и числовой информации в видеопамять.

- Процедура GrabRusFont считывает текущий шрифт 8×16 прямо из памяти видеоконтроллера и записывает его в массив Font8×l6. Перед запуском подпрограммы должен быть установлен текстовый режим, а сразу после ее выполнения желательно выполнить переустановку видеорежима.
- Процедура SetVESAVideoMode проверяет наличие режима VESA с номером, заданным константой GraphicsMode, запоминает в глобальной переменной LinearVideoBuffer начальный абсолютный (линейный) адрес памяти видеоконтроллера, устанавливает видеорежим, а затем увеличивает логическую длину строки экрана до 1024 пикселов. В случае если режим не поддерживается, процедура завершается немедленным (аварийным) выходом из программы.
- Процедура SetTrueCo1or32 осуществляет поиск видеорежима TrueColor32 с разрешением, заданным константами ScreenLength и ScreenHeigth, устанавливает его (если он поддерживается видеоконтроллером) и задает логическую длину строки экрана 1024 пиксела. В случае если подходящий режим не обнаружен, процедура завершается немедленным выходом из программы.
- Процедура WaitVSync предназначена для синхронизации вывода динамических изображений — она выполняет операцию ожидания начала обратного хода луча по кадру.
- Процедура GShowString выводит строку текста в заданную область экрана, используя цвет символов и фона, заданный по умолчанию.
- Процедура GShowByteHexCode отображает в заданную позицию экрана содержимое регистра AL (байт данных) в шестнадцатеричном коде.
- Процедура GShowHexWord отображает в заданную позицию экрана содержимое регистра AX (слово данных) в шестнадцатеричном коде.
- Процедура GShowHexDWord отображает в заданную позицию экрана содержимое регистра EAX (двойное слово) в шестнадцатеричном коде.

- Процедура GShowByteBinCode отображает в заданную позицию экрана содержимое регистра AL (байт данных) в двоичном коде.
- Процедура GShowBinDWord отображает в заданную позицию экрана содержимое регистра EAX (двойное слово) в двоичном коде.
- Процедура ShowColorString выводит текстовую строку заданного цвета в заданную область экрана.

Листинг 2.8. Процедуры общего назначения, предназначенные для установки графических режимов и работы в них (list4_02.inc)

```
DATASEG
; Место для хранения информации VESA BIOS
VESA_BIOS  DB 512 DUP(?)
; Область для приема информации о параметрах видеорежима
VESA_info  DB 256 DUP(?)
; Физический адрес линейного видеобуфера
LinearVideoBuffer DD ?
; Буфер для сохранения шрифта (16x256 байт)
Font8x16   DB 4096 DUP(?)
; Позиция отображаемого символа
FontString DW ? ;номер строки шрифта
FontColumn DW ? ;номер колонки шрифта
ErrMod DB 12,17,"Заданный режим не "
       DB "поддерживаются контроллером",0
ErrPrm DB 12,17
       DB "Некорректный параметр у функции GShowString",0
ENDS

CODESEG
;*****************************************************
;*      СЧИТЫВАНИЕ "РУССКОГО" ШРИФТА ИЗ ПАМЯТИ       *
;*                 ВИДЕОКОНТРОЛЛЕРА                  *
;*          (процедура параметров не имеет)          *
;* Операция захвата шрифта требует перепрограммиро-  *
;* вания регистров видеоконтроллера и ее следует     *
;* выполнять непосредственно перед переходом из      *
;* текстового режима в графический, чтобы не восста- *
;* навливать регистры (при смене режима они все      *
;* равно будут перезаписаны).                        *
;*****************************************************
PROC GrabRusFont near
        pushad
; Перепрограммировать синхронизатор
        cli
        mov     DX,3C4h
        ; Установить последовательную адресацию
        ; ячеек видеопамяти
        mov     AX,0704h
        out     DX,AX
        sti
; Перепрограммировать графический контроллер
        mov     DX,3CEh
        ; Выбрать для считывания плоскость 2
        mov     AX,0204h
        out     DX,AX
        ; Запретить четную-нечетную адресацию
        mov     AX,0005h
        out     DX,AX
        ; Установить окно доступа по адресу A0000h
        mov     AX,0006h
        out     DX,AX
; Скопировать шрифт в буфер Font8x16
        mov     AX,0A000h
        mov     ES,AX
        mov     SI,0
        mov     BX,offset Font8x16
        mov     DX,256
@@M0:   mov     CX,16
@@M1:   mov     AL,[ES:SI]
        mov     [BX],AL
        inc     BX
        inc     SI
        loop    @@M1
        add     SI,16
        dec     DX
        jnz     @@M0
        popad
        ret
ENDP GrabRusFont
;*****************************************
;* УСТАНОВИТЬ ВИДЕОРЕЖИМ ЧЕРЕЗ ФУНКЦИЮ VESA *
;*     (процедура параметров не имеет)     *
;*****************************************
PROC SetVESAVideoMode near
        pushad
        push    ES
        mov     AX,DS
        mov     ES,AX
; Получить информацию о наличии VESA BIOS
        mov     AX,4F00h
        mov     DI,offset VESA_BIOS
        int     10h
; Получить результат выполнения функции
        cmp     AX,4Fh
        jne     @@Err   ;функция не выполнена
        cmp     [dword ptr VESA_BIOS],'ASEV'
        jne     @@Err   ;нет сигнатуры "VESA"
        cmp     [word ptr VESA_BIOS+4],200h
        jb      @@Err   ;версия "VESA" слишком старая
; Получить параметры видеорежима, номер
; которого задан переменной GraphicsMode
        mov     AX,4F01h
        mov     CX,GraphicsMode
        mov     DI,offset VESA_info
        int     10h
; Выделить адрес линейной области видеопамяти
        mov     EAX,[offset VESA_info+28h]
        mov     [LinearVideoBuffer],EAX
; Установить заданный видеорежим
        mov     BX,GraphicsMode
        mov     AX,4F02h
        int     10h
; Устанавливаем заданную логическую ширину строки
        xor     BX,BX
        mov     CX,LogicalStringLength
        mov     AX,4F06h
        int     10h
        pop     ES
        popad
        ret
; Аварийный выход - нет поддержки VESA
@@Err:  MFatalError ErrMod
ENDP SetVESAVideoMode

;*************************************
;* УСТАНОВИТЬ ВИДЕОРЕЖИМ TRUECOLOR32 *
;*  (процедура параметров не имеет)  *
;*************************************
PROC SetTrueColor32 near
        pushad
        push    ES
; Получить информацию о наличии VESA BIOS
        mov     AX,DS
        mov     ES,AX
        mov     AX,4F00h
        mov     DI,offset VESA_BIOS
        int     10h
; Проверить результат выполнения функции
```

Листинг 2.8 (продолжение)

```
        cmp     AX,4Fh
        jne     @@Err   ;функция не выполнена
        cmp     [dword ptr VESA_BIOS],'ASEV'
        jne     @@Err   ;нет сигнатуры "VESA"
        cmp     [word ptr VESA_BIOS+4],200h
        jb      @@Err   ;версия "VESA" слишком старая
; Найти видеорежим с параметрами TrueColor32 640x480
        ; Загрузить адрес массива номеров
        mov     BX,[word ptr VESA_BIOS+0Eh]
        mov     AX,[word ptr VESA_BIOS+10h]
        mov     ES,AX
@@NextMode:
        ; Получить параметры очередного видеорежима
        mov     CX,[ES:BX] ;загрузить номер режима
        cmp     CX,0       ;ошибка?
        je      @@Err
        cmp     CX,0FFFFh  ;конец списка?
        je      @@Err
        push    ES
        mov     AX,DS
        mov     ES,AX
        mov     AX,4F01h
        mov     DI,offset VESA_info
        int     10h
        pop     ES
; Проверить значения полей
        ; Горизонтальное разрешение
        cmp     [word ptr VESA_info+12h],ScreenLength
        jne     @@NotTrueColor32
        ; Вертикальное разрешение
        cmp     [word ptr VESA_info+14h],ScreenHeigth
        jne     @@NotTrueColor32
        ; Число битов на пиксел
        cmp     [byte ptr VESA_info+19h],32
        jne     @@NotTrueColor32
        ; Линейный адрес буфера кадра
        cmp     [dword ptr VESA_info+28h],0
        jne     @@SetMode
@@NotTrueColor32:
        inc     BX
        jmp     @@NextMode

@@SetMode:
; Выделить адрес линейной области видеопамяти
        mov     EAX,[offset VESA_info+28h]
        mov     [LinearVideoBuffer],EAX
; Установить найденный режим
        mov     AX,4F02h
        ; Загрузить номер режима
        mov     BX,[ES:BX]
        ; Использовать линейную адресацию
        or      BX,4000h
        int     10h
; Установить логическую ширину строки
        mov     AX,4F06h
        xor     BX,BX
        mov     CX,LogicalStringLength
        int     10h
        pop     ES
        popad
        ret
; Аварийный выход - нет поддержки заданного режима
@@Err:  MFatalError ErrMod
ENDP SetTrueColor32
;****************************************
;* ОЖИДАНИЕ ОБРАТНОГО ХОДА ЛУЧА ПО КАДРУ *
;****************************************
PROC WaitVSync NEAR
        push  AX
        push  DX
        mov   DX,03DAh ;регистр статуса VGA
@@WaitNotVSyncLoop:
        in    AL,DX
        and   AL,08h ;выделяем бит вертикальной синхронизации
        jnz   @@WaitNotVSyncLoop
@@WaitVSyncLoop:
        in    AL,DX
        and   AL,08h ;выделяем бит вертикальной синхронизации
        jz    @@WaitVSyncLoop
        pop   DX
        pop   AX
        ret
ENDP WaitVSync

;*****************************************************
;*          ВЫВОД ТЕКСТОВОЙ СТРОКИ НА ЭКРАН          *
;* Все параметры передаются через одну структуру:    *
;* первый байт - номер начальной строки (0-47);      *
;* второй байт - номер начальной колонки  (0-127);   *
;* далее идет строка, ограниченная нулем.            *
;* Адрес структуры передается через регистры DS:SI.  *
;*****************************************************
PROC GShowString near
        push  AX
        push  DX
        cld
; Вычисляем адрес для строки в видеопамяти
        ;загрузить номер строки экрана в DH
        lodsb
        cmp     AL,47
        ja      @@Err ;выход за нижнюю границу экрана
        mov     DH,AL
        ;загрузить номер строки экрана в DL
        lodsb
        cmp     AL,127
        ja      @@Err ;выход за правую границу экрана
        mov     DL,AL
@@L1:   ; Загрузить очередной символ строки в AL
        lodsb
        ; Проверка на 0 (на конец строки)
        and     AL,AL
        jz      @@L2
        ; Вывести символ на экран
        call    PutGraChar
        jmp     @@L1
; Нормальное завершение
@@L2:   pop     DX
        pop     AX
        ret
; Немедленный выход в DOS при ошибке
@@Err:  MFatalError ErrPrm
ENDP GShowString

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

;*******************************************************
```

Листинг 2.8 (продолжение)

```
;*    ВЫВОД БАЙТА НА ЭКРАН В ШЕСТНАДЦАТЕРИЧНОМ 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 GShowHexByte near
        pusha
        mov     DH,[byte ptr ScreenString]
        mov     DL,[byte ptr ScreenColumn]
; 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:   call    PutGraChar
        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:   call    PutGraChar

; 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]
; Конец подпрограммы
@@End:  popa
        ret
ENDP GShowHexByte

;*****************************************************
;*        ВЫВОД 16-РАЗРЯДНОГО СЛОВА НА ЭКРАН         *
;*              В ШЕСТНАДЦАТЕРИЧНОМ КОДЕ             *
;* Параметры:                                         *
;* AX - число, которое будет выведено на экран.       *
;* Номер строки передается через глобальную           *
;* переменную ScreenString, номер столбца - через     *
;* переменную ScreenColumn, цвет текста определяется  *
;* переменными DefaultColor и DefaultBackground.      *
;*****************************************************
PROC GShowHexWord NEAR

        xchg    AL,AH
        call    GShowHexByte
        xchg    AL,AH
        call    GShowHexByte
        ret
ENDP GShowHexWord

;*****************************************************
;*        ВЫВОД 32-РАЗРЯДНОГО СЛОВА НА ЭКРАН         *
;*              В ШЕСТНАДЦАТЕРИЧНОМ КОДЕ             *
;* Параметры:                                         *
;* EAX - число, которое будет выведено на экран.      *
;* Номер строки передается через глобальную           *
;* переменную ScreenString, номер столбца - через     *
;* переменную ScreenColumn, цвет текста определяется  *
;* переменными DefaultColor и DefaultBackground.      *
;*****************************************************
PROC GShowHexDWord NEAR
        rol     EAX,8
        call    GShowHexByte
        rol     EAX,8
        call    GShowHexByte
        rol     EAX,8
        call    GShowHexByte
        rol     EAX,8
        call    GShowHexByte
        ret
ENDP GShowHexDWord
;*****************************************************
;*        ВЫВОД БАЙТА НА ЭКРАН В ДВОИЧНОМ КОДЕ       *
;* Подпрограмма выводит содержимое регистра AL        *
;* в двоичном 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 GShowBinByte near
        pusha
        mov     DH,[byte ptr ScreenString]
        mov     DL,[byte ptr ScreenColumn]
        ; Копируем отображаемый байт в BL
        mov     BL,AL
; Отобразить разряды числа (начиная со старшего)
        mov     CX,8   ;счетчик разрядов
@@L0:   mov     AL,'0'
        ; Выделить очередной разряд числа
        rol     BL,1

        jnc     @@L1
        mov     AL,'1'
        ; Вывести разряд числа на экран
@@L1:   call    PutGraChar
        loop    @@L0
; Подготовка для вывода следующих байтов
        ; Перевести текущую позицию на 8 символов влево
        add     [ScreenColumn],8
        ; Проверить пересечение правой границы экрана
        cmp     [ScreenColumn],80
        jb      @@End
        ; Если достигнута правая граница экрана -
        ; перейти на следующую строку
        sub     [ScreenColumn],80
        inc     [ScreenString]
; Конец подпрограммы
@@End:  popa
        ret
ENDP GShowBinByte
;*****************************************************
;* ВЫВОД 32-РАЗРЯДНОГО СЛОВА НА ЭКРАН В ДВОИЧНОМ КОДЕ *
;* Параметры:                                         *
;* EAX - число, которое будет выведено на экран.      *
;* Номер строки передается через глобальную           *
;* переменную ScreenString, номер столбца - через     *
;* переменную ScreenColumn, цвет текста определяется  *
;* переменными DefaultColor и DefaultBackground.      *
;*****************************************************
PROC GShowBinDWord NEAR
        rol     EAX,8
        call    GShowBinByte
```

Листинг 2.8 (продолжение)

```
        inc     [ScreenColumn]
        rol     EAX,8
        call    GShowBinByte
        inc     [ScreenColumn]
        rol     EAX,8
        call    GShowBinByte
        inc     [ScreenColumn]
        rol     EAX,8
        call    GShowBinByte
        ret
ENDP GShowBinDWord
ENDS
```

Универсальная процедура вывода текстовой строки опирается на процедуру вывода символа, специфическую для каждого типа видеорежима. В листинге 2.9 показан вариант реализации процедуры вывода символа PutGraChar и процедуры очистки экрана GClearScreen для 256-цветных режимов с линейным видеобуфером.

ВНИМАНИЕ ——————————————————————————————————

Стандартные растровые шрифты MS-DOS с размером маски символа 8×16 целесообразно использовать только при низком разрешении экрана (до 800×600 включительно). При разрешении 1024×768 и выше программист вынужден создавать (при помощи типовой или самодельной программы — генератора шрифтов) свой собственный шрифт, увеличенный в 1,5 - 2 раза (12×24 или 16×32).

Листинг 2.9. Процедуры вывода символа и очистки экрана для 256-цветных режимов с линейной адресацией видеобуфера (list4_03.inc)

```
DATASEG
; Цвет текста в графическом режиме по умолчанию
DefaultColor      DB WHITE  ;белый
; Цвет фона в графическом режиме по умолчанию
DefaultBackground DB BLACK  ;черный
ENDS
CODESEG
;*****************************************************
;* ВЫВОД СИМВОЛА 8x16 НА ЭКРАН В ГРАФИЧЕСКОМ РЕЖИМЕ *
;*            (для 256-цветных режимов)              *
;* Все параметры передаются через регистры:          *
;* AL - ASCII-код символа;                           *
;* DH - номер текстовой строки экрана;               *
;* DL - номер текстовой колонки экрана;              *
;* Используются цвет символов и цвет фона,           *
;* заданные по умолчанию.                            *
;*****************************************************
PROC PutGraChar NEAR
        push    DS
        pushad
        mov     CX,[CS:MainDataSeg]
        mov     DS,CX
        cld
        ; Смещение символа от начала шрифта
        mov     SI,offset Font8x16
        xor     AH,AH
        shl     AX,4
        add     SI,AX
        ; Вычислить левый верхний угол символа
        xor     EBX,EBX
        mov     BL,DH
        shl     EBX,14 ;умножить номер строки на 16*1024
        xor     DH,DH
        shl     DX,3   ;умножить номер столбца на 8
        or      BX,DX
        mov     EDI,EBX
        add     EDI,[LinearVideoBuffer]
        mov     BL,[DefaultColor]
        mov     DL,[DefaultBackground]

        mov     AH,16 ;счетчик строк маски буквы
@@M0:   lodsb
        mov     CX,8  ;счетчик точек в строке маски
@@M1:   rol     AL,1
        jc      @@M2
        mov     [GS:EDI],DL
        jmp     @@M3
@@M2:   mov     [GS:EDI],BL
@@M3:   inc     EDI
        loop    @@M1
        add     EDI,LogicalStringLength-8
        dec     AH
        jnz     @@M0
; Завершение процедуры
@@EndPutGraChar:
        popad
        pop     DS
        inc     DL
        ret
ENDP PutGraChar
;*************************************
;* ОЧИСТКА ЭКРАНА В ГРАФИЧЕСКОМ РЕЖИМЕ *
;*   (процедура параметров не имеет)   *
;*************************************
PROC GClearScreen NEAR
        pushad
; Умножить высоту экрана ScreenHeigth на логическую
; ширину строки 1024 пиксела
        mov     ECX,ScreenHeigth
        shl     ECX,10
; Загрузить в индексный регистр линейный
; адрес видеопамяти
        mov     EDI,[LinearVideoBuffer]
; Заполнить видеопамять нулями
        mov     AL,0     ;черный цвет
@@NextPixels:
        mov     [GS:EDI],AL
        inc     EDI
        dec     ECX
        jnz     @@NextPixels
        popad
        ret
ENDP GClearScreen
ENDS
```

Программа Test256Mode, приведенная в листинге 2.10, демонстрирует вывод текста и графики (пучка прямых линий) в 256-цветном режиме. В нем используются процедуры из листингов 2.8 и 2.9, а также универсальные модули из глав «Работа с клавиатурой» и «Недокументированные возможности процессоров Intel 80x86». При рисовании линий применяются упрощенные алгоритмы, каждый из которых позволяет чертить линию только под строго определенным углом.

В приведенном примере выбрано разрешение 640×480 точек, при котором текст, выводимый шрифтом 8×16, выглядит почти так же, как в текстовом режиме. Вы можете изменить разрешение, выбрав одно из трех предложенных в примере значений константы GraphicsMode. При увеличении разрешения становится менее заметным «лестничный эффект», то есть линии кажутся более «гладкими», но снижается яркость изображения (из-за того, что точки стали меньше, а линии — тоньше) и хуже читается текст (вследствие уменьшения размера символов). Изменяя номер режима, не забудьте присвоить соответствующие значения константам, описывающим физическое разрешение экрана (ScreenLength и ScreenHeigth).

Листинг 2.10. Вывод на экран текста и линий в режиме 256 цветов с разрешением 640×480

```
    IDEAL
    P386
    LOCALS
    MODEL MEDIUM
    ; Коды 256-цветных видеорежимов с линейной
    ; адресацией видеобуфера:
    ;    4101h - режим с разрешением 640x480
    ;    4103h - режим с разрешением 800x600
    ;    4105h - режим с разрешением 1024x768
    GraphicsMode  equ  4101h
    ; Логическая ширина строки в пикселах
    LogicalStringLength equ 1024
    ; Ширина экрана в пикселах
    ScreenLength equ 640
    ; Высота экрана, строк
    ScreenHeigth equ 480

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

    DATASEG
    ; Абсолютный адрес начальной точки для "пучка"
; цветных линий
StartPixelAddress DD ?
; Текстовые сообщения
Txt1 DB 0,19
     DB "ТЕСТ 256-ЦВЕТНОГО РЕЖИМА ВИДЕОКОНТРОЛЛЕРА",0
Txt2 DB 2,22,"Вывод на экран текста, вертикальных,",0
     DB 3,18
     DB "горизонтальных и диагональных цветных линий",0
AnyK DB 29,29,"Нажмите любую клавишу",0
ENDS
SEGMENT sseg para stack 'STACK'
DB 400h DUP(?)
ENDS
CODESEG
;****************************
;* Основной модуль программы *
;****************************
PROC Test256Mode
        mov     AX,DGROUP
        mov     DS,AX
        mov     [CS:MainDataSeg],AX
; Установить текстовый режим
        mov     AX,3
        int     10h
; Установить режим прямой адресации памяти
        call    Initialization
; "Захватить" текстовый шрифт
        call    GrabRusFont
; Установить видеорежим
        call    SetVESAVideoMode
; Отобразить текстовые сообщения
        ; Установить черный цвет фона
        mov     [DefaultBackground],BLACK
        ; Установить желтый цвет текста
        mov     [DefaultColor],YELLOW
        MGShowString Txt1
        ; Установить зеленый цвет текста
        mov     [DefaultColor],LIGHTGREEN
        MGShowText 2,Txt2
        ; Установить желтый цвет текста
        mov     [DefaultColor],YELLOW
        MGShowString AnyK
; Вычислить адрес начальной точки для пучка линий.
; Координаты точки (220,100)
        ; Умножить длину строки на номер строки (Y)
        mov     EAX,LogicalStringLength
        mov     EDX,100
            mul     EDX
            ; Прибавить номер колонки (X)
            add     EAX,220
            ; Прибавить адрес видеобуфера
            add     EAX,[LinearVideoBuffer]
            ; Запомнить адрес начальной точки
            mov     [StartPixelAddress],EAX
    ; Начертить на экране горизонтальную белую
    ; линию с координатами (220,100)-(420,100)
            ; Установить цвет линии
            mov     AL,WHITE
            ; Записать адрес начальной
            ; точки в индексный регистр
            mov     EDI,[StartPixelAddress]
            ; Задать длину линии в пикселах
            mov     CX,200
    @@L1:   mov     [GS:EDI],AL ;нарисовать пиксел
            inc     EDI         ;перейти в следующую позицию
            loop    @@L1
    ; Начертить вертикальную белую линию
    ; с координатами (220,100)-(220,300)
            mov     AL,WHITE
            mov     EDI,[StartPixelAddress]
            mov     CX,200
    @@L2:   mov     [GS:EDI],AL
            add     EDI,LogicalStringLength
            loop    @@L2
    ; Начертить красную линию с координатами
    ; (220,100)-(420,300) (под углом 45 градусов
    ; к горизонтали)
            mov     AL,LIGHTRED
            mov     EDI,[StartPixelAddress]
            mov     CX,200
```

Листинг 2.10 (продолжение)

```
@@L3:   mov     [GS:EDI],AL
        add     EDI,LogicalStringLength+1
        loop    @@L3
; Начертить зеленую линию с координатами
; (220,100)-(420,200) (под углом 27 градусов
; к горизонтали)
        mov     AL,LIGHTGREEN
        mov     EDI,[StartPixelAddress]
        mov     CX,200/2
@@L4:   mov     [GS:EDI],AL
        inc     EDI
        mov     [GS:EDI],AL
        add     EDI,LogicalStringLength+1
        loop    @@L4
; Начертить синюю линию с координатами
; (220,100)-(320,300) (под углом 63 градуса
; к горизонтали)
        mov     AL,LIGHTBLUE
        mov     EDI,[StartPixelAddress]
        mov     CX,200/2
@@L5:   mov     [GS:EDI],AL
        add     EDI,LogicalStringLength
        mov     [GS:EDI],AL
        add     EDI,LogicalStringLength+1
        loop    @@L5
; Ожидать нажатия любой клавиши
        call    GetChar
; ВЫХОД ИЗ ПРОГРАММЫ
@@End:  ; Установить текстовый режим
        mov     AX,3
        int     10h
        ; Выход в DOS
        mov     AH,4Ch
        int     21h
ENDP Test256Mode
ENDS
; Подключить процедуры ввода данных и вывода на экран
; в текстовом режиме
include "list1_02.inc"
; Подключить подпрограмму, переводящую сегментный
; регистр GS в режим линейной адресации
include "list2_01.inc"
; Подключить набор процедур общего назначения,
; предназначенных для установки графических
; видеорежимов и работы в них
include "list4_02.inc"
; Подключить набор процедур вывода текста,
; предназначенных для 256-цветных режимов
include "list4_03.inc"

END
```

При выводе текста и чертежей режимы HiColor и TrueColor не дают никаких преимуществ по сравнению с 256-цветными режимами. В то же время 256-цветные режимы экономно используют видеопамять (достаточно 1-2 Мбайт даже при высоком разрешении) и существенно превосходят все остальные типы режимов по скорости вывода информации (правда, только при использовании специальных алгоритмов, позволяющих выводить по четыре пиксела за одну операцию). Листинги программ, реализующих вывод текста и чертежей в режимах HiColor и TrueColor, приведены в [5].

Пример использования алгоритма Брезенхема для рисования линии в 256-цветных режимах приведен в листингах 2.11 и 2.12. Процедура рисования линии EVGALine, приведенная в листинге 2.11, получена путем прямого перевода примера из книги Майкла Абраша [1]с языка С на язык Ассемблер x86. Процедура использует две вспомогательные подпрограммы Octant0 и Octant1 для рисования линий в различных октантах. Процедура специально приведена в неоптимизированном варианте (переменные хранятся в памяти, а не в регистрах), так как после оптимизации программа утрачивает свойство наглядности. Вызывающая (основная) программа TestLines256 из листинга 2.12 предназначена для тестирования процедуры рисования линии: она выводит четыре пучка линий различных цветов — белого, красного, синего и зеленого. Листинги рисования линий по алгоритму Брезенхема для режима TrueColor32 приведены в [5].

Листинг 2.11. Подпрограмма рисования линии по алгоритму Брезенхема для 256-цветных режимов (list4_09.inc)

```
DATASEG
EVEN        ;выравнивание смещения данных на 2
; Координаты начала линии
X0          DD ?
Y0          DD ?
; Координаты конца линии
X1          DD ?
Y1          DD ?
; Длина линии по X и по Y
DeltaX      DD ?
DeltaY      DD ?
; Направление рисования по X (1 - линия
; прорисовывается слева направо, -1 - справа налево)
XDirection DD ?
; Цвет линии
Color       DB ?
EVEN        ;выравнивание смещения данных на 2
; Внутренние переменные процедур рисования
DeltaYx2                DD ?
DeltaYx2MinusDeltaXx2   DD ?
DeltaXx2                DD ?
DeltaXx2MinusDeltaYx2   DD ?
ErrorTerm               DD ? ;ошибка накопления
PixelOffset             DD ?
ENDS
CODESEG
;*********************************************
;* ПРОЦЕДУРА РИСОВАНИЯ ЛИНИИ В ОКТАНТАХ 0 И 3 *
;*            (|DeltaX| >= DeltaY)            *
;*********************************************
PROC Octant0 NEAR
        pushad
; Установить начальную ошибку накопления и значения,
; используемые во внутреннем цикле
        ; (DeltaYx2 = 2*DeltaY)
        mov     EAX,[DeltaY]
        shl     EAX,1
        mov     [DeltaYx2],EAX
```

Листинг 2.11 (продолжение)

```
        ; (ErrorTerm = DeltaYx2 - DeltaX)
        sub     EAX,[DeltaX]
        mov     [ErrorTerm],EAX
        ; (DeltaYx2MinusDeltaXx2 = DeltaYx2 - 2*DeltaX)
        sub     EAX,[DeltaX]
        mov     [DeltaYx2MinusDeltaXx2],EAX
; Рисуем линию
        ; (PixelOffset=Y0*LogicalStringLength+X0)
        mov     EAX,[Y0]
        mov     EDX,LogicalStringLength
        mul     EDX
        add     EAX,[X0]
        mov     [PixelOffset],EAX
; Рисуем первую точку линии
        mov     EBX,[PixelOffset]
        add     EBX,[LinearVideoBuffer]
        mov     AL,[Color]
        mov     [GS:EBX],AL
; Цикл, пока DeltaX>=0
@@NextDot:
        ; Проверить, не пора ли перейти на точку
        ; по оси Y
        cmp     [ErrorTerm],0
        jl      @@AddError
        ; Сделать шаг по Y
        add     [PixelOffset],LogicalStringLength
        ; Увеличить ошибку накопления
        mov     EAX,[DeltaYx2MinusDeltaXx2]
        add     [ErrorTerm],EAX
        jmp     @@PutPixel
@@AddError:
        ; Увеличить ошибку накопления
        mov     EAX,[DeltaYx2]
        add     [ErrorTerm],EAX
@@PutPixel:
        ; Сделать шаг по X
        mov     EAX,[XDirection]
        add     [PixelOffset],EAX
        ; Вывести очередную точку линии на экран
        mov     EBX,[PixelOffset]
        add     EBX,[LinearVideoBuffer]
        mov     AL,[Color]
        mov     [GS:EBX],AL
        dec     [DeltaX]
        jnz     @@NextDot
      popad
      ret
ENDP Octant0
;*********************************************
;* ПРОЦЕДУРА РИСОВАНИЯ ЛИНИИ В ОКТАНТАХ 1 И 2 *
;*          (|DeltaX| < DeltaY)             *
;*********************************************
PROC Octant1 NEAR
      pushad
; Установить начальную ошибку накопления и значения,
; используемые во внутреннем цикле
      ; (DeltaXx2 = 2*DeltaX)
      mov     EAX,[DeltaX]
      shl     EAX,1
      mov     [DeltaXx2],EAX
      ; (ErrorTerm = DeltaXx2 - DeltaY)
      sub     EAX,[DeltaY]
      mov     [ErrorTerm],EAX
      ; (DeltaXx2MinusDeltaYx2 = DeltaXx2 - 2*DeltaY)
      sub     EAX,[DeltaY]
      mov     [DeltaXx2MinusDeltaYx2],EAX
; Рисуем линию
      ; (PixelOffset=Y0*LogicalStringLength+X0);
      mov     EAX,[Y0]
      mov     EDX,LogicalStringLength
      mul     EDX
      add     EAX,[X0]
      mov     [PixelOffset],EAX
; Рисуем первый пиксел
      mov     EBX,[PixelOffset]
      add     EBX,[LinearVideoBuffer]
      mov     AL,[Color]
      mov     [GS:EBX],AL
; Цикл, пока DeltaY>=0
@@NextDot:
      ; Проверить, не пора ли перейти на точку
      ; по оси X
      cmp     [ErrorTerm],0
      jl      @@AddError
      ; Сделать шаг по X
      mov     EAX,[XDirection]
      add     [PixelOffset],EAX
      ; Увеличить ошибку накопления
      mov     EAX,[DeltaXx2MinusDeltaYx2]
      add     [ErrorTerm],EAX
      jmp     @@PutPixel
      ; Увеличить ошибку накопления
@@AddError:
      mov     EAX,[DeltaXx2]
          add     [ErrorTerm],EAX
    @@PutPixel:
          ; Сделать шаг по Y
          add     [PixelOffset],LogicalStringLength
          ; Вывести очередную точку линии на экран
          mov     EBX,[PixelOffset]
          add     EBX,[LinearVideoBuffer]
          mov     AL,[Color]
          mov     [GS:EBX],AL
          dec     [DeltaY]
          jnz     @@NextDot
          popad
          ret
    ENDP Octant1

    ;*****************************************************
    ;* ПРОЦЕДУРА РИСОВАНИЯ ЛИНИИ ПО АЛГОРИТМУ БРЕЗЕНХЕМА *
    ;* Передача параметров выполняется через             *
    ;* глобальные переменные:                            *
    ;* X0, Y0 - координаты начальной точки;              *
    ;* X1, Y1 - координаты конечной точки;               *
    ;* Color - цвет линии.                               *
    ;*****************************************************
    PROC EVGALine NEAR
          pushad
    ; Запомнить координаты линии в стеке
          push    [X0]
          push    [Y0]
          push    [X1]
          push    [Y1]
    ; Если Y0 > Y1, поменять местами начальную
    ; и конечную точки линии.
          mov     EAX,[Y0]
          cmp     EAX,[Y1]
          jbe     @@L0
          xchg    EAX,[Y1]
          xchg    EAX,[Y0]
          mov     EAX,[X0]
          xchg    EAX,[X1]
          xchg    EAX,[X0]
    @@L0:
    ; Вычислить DeltaX
          mov     EAX,[X1]
          sub     EAX,[X0]
          mov     [DeltaX],EAX
    ; Вычислить DeltaY
        mov     EAX,[Y1]
        sub     EAX,[Y0]
        mov     [DeltaY],EAX
    ; Выбрать номер октанта и направление движения
          mov     [XDirection],1
          mov     EAX,[DeltaX]
          cmp     EAX,0
          jge     @@L1
```

Листинг 2.11 (продолжение)

```
        neg     EAX
        mov     [DeltaX],EAX
        mov     [XDirection],-1
@@L1:   cmp     EAX,[DeltaY]
        jle     @@L2
        call    Octant0
        jmp     @@End
@@L2:   call    Octant1
; Восстановить координаты линии
@@End:  pop     [Y1]
        pop     [X1]
        pop     [Y0]
        pop     [X0]
        popad
        ret
ENDP EVGALine
ENDS
```

Листинг 2.12. Рисование линий по алгоритму Брезенхема в режиме 256 цветов с разрешением 640×480

```
IDEAL
P386
LOCALS
MODEL MEDIUM
; Коды 256-цветных видеорежимов с линейной
; адресацией видеобуфера:
;    4101h - режим с разрешением 640x480
;    4103h - режим с разрешением 800x600
;    4105h - режим с разрешением 1024x768
GraphicsMode  equ  4101h
; Логическая ширина строки в пикселах
LogicalStringLength equ 1024
; Ширина экрана в пикселах
ScreenLength equ 640
; Высота экрана, строк
ScreenHeigth equ 480
; Подключить файл мнемонических обозначений
; кодов управляющих клавиш и цветовых кодов
include "list1_03.inc"
; Подключить файл макросов
include "list1_04.inc"
DATASEG
; Текстовые сообщения
Txt1 DB 0,22,"РИСОВАНИЕ ЛИНИЙ В 256-ЦВЕТНОМ РЕЖИМЕ",0
AnyK DB 29,29,"Нажмите любую клавишу",0
ENDS
SEGMENT sseg para stack 'STACK'
DB 400h DUP(?)
ENDS
CODESEG
;*****************************
;* Основной модуль программы *
;*****************************
PROC TestLines256
        mov     AX,DGROUP
        mov     DS,AX
        mov     [CS:MainDataSeg],AX
; Установить текстовый режим
        mov     AX,3
        int     10h
; Установить режим прямой адресации памяти
        call    Initialization
; "Захватить" текстовый шрифт
        call    GrabRusFont
; Установить видеорежим
        call    SetVESAVideoMode
; Отобразить текстовые сообщения
        ; Установить черный цвет фона
        mov     [DefaultBackground],BLACK
        ; Установить зеленый цвет текста
        mov     [DefaultColor],LIGHTCYAN
        MGShowString Txt1
        ; Установить желтый цвет текста
        mov     [DefaultColor],YELLOW
        MGShowString AnyK
; Занести координаты начальной точки для пучка линий
        mov     [dword ptr X0],320
        mov     [dword ptr Y0],240
; Нарисовать пучок белых линий
        mov     EAX,[dword ptr X0]
        add     EAX,100
        mov     [dword ptr X1],EAX
        mov     EAX,[dword ptr Y0]
        sub     EAX,100
        mov     [dword ptr Y1],EAX
        mov     [Color],WHITE
        mov     CX,50
@@NextWhiteLine:
        call    EVGALine
        add     [dword ptr Y1],4
        loop    @@NextWhiteLine
; Нарисовать пучок красных линий
        mov     EAX,[dword ptr X0]
        add     EAX,100
        mov     [dword ptr X1],EAX
        mov     EAX,[dword ptr Y0]
        add     EAX,100
        mov     [dword ptr Y1],EAX
        mov     [Color],LIGHTRED
        mov     CX,50
@@NextRedLine:
        call    EVGALine
        sub     [dword ptr X1],4
        loop    @@NextRedLine
; Нарисовать пучок зеленых линий
        mov     EAX,[dword ptr X0]
        sub     EAX,100
        mov     [dword ptr X1],EAX
        mov     EAX,[dword ptr Y0]
        add     EAX,100
        mov     [dword ptr Y1],EAX
        mov     [Color],LIGHTGREEN
        mov     CX,50
@@NextGreenLine:
        call    EVGALine
        sub     [dword ptr Y1],4
        loop    @@NextGreenLine
; Нарисовать пучок синих линий
        mov     EAX,[dword ptr X0]
        sub     EAX,100
        mov     [dword ptr X1],EAX
        mov     EAX,[dword ptr Y0]
        sub     EAX,100
        mov     [dword ptr Y1],EAX
        mov     [Color],LIGHTBLUE
        mov     CX,50
@@NextBlueLine:
        call    EVGALine
        add     [dword ptr X1],4
        loop    @@NextBlueLine

; Ожидать нажатия любой клавиши
        call    GetChar
; Установить текстовый режим
        mov     AX,3
        int     10h
; Выход в DOS
        mov     AH,4Ch
```

Листинг 2.12 (продолжение)

```
        int     21h
ENDP TestLines256
ENDS
; Подключить процедуры ввода данных и вывода на экран
; в текстовом режиме
include "list1_02.inc"
; Подключить подпрограмму, переводящую сегментный
; регистр GS в режим линейной адресации
include "list2_01.inc"
; Подключить набор процедур общего назначения,
; предназначенных для установки графических
; видеорежимов и работы в них
include "list4_02.inc"
; Подключить набор процедур вывода текста,
; предназначенных для 256-цветных режимов
include "list4_03.inc"
; Подключить подпрограмму рисования линии по
; алгоритму Брезенхема для 256-цветных режимов
include "list4_09.inc"

END
```

2.2.6.2.6. Масштабирование изображений

При работе с различными изображениями на компьютере довольно часто возникает необходимость в их увеличении или уменьшении, то есть в масштабировании. Алгоритмы для получения изображения в произвольном масштабе относительно сложные, но увеличить или уменьшить изображение в целое число раз нетрудно.

Чтобы увеличить изображение без искажений, достаточно просто увеличить в N раз каждую его точку, то есть представить ее в виде квадрата N×N точек. Недостаток такого способа заключается в том, что сильно проявляет себя «лестничный эффект» — становится очень заметной ступенчатость изображения. Пример двукратного увеличения изображения текстового символа показан на рис. 2.33.

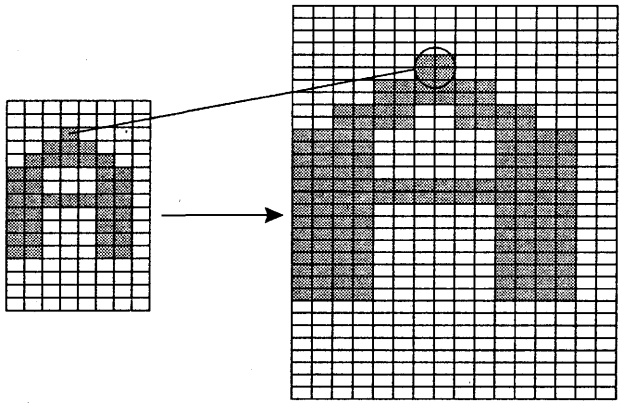


Рис. 2.33. Увеличение размера изображения символа в два раза

Столь же просто можно деформировать изображение, растягивая его по вертикали и горизонтали в различное число раз. При этом точка превращается уже не в квадрат, а в прямоугольник, содержащий N×M точек.

Чтобы сжать изображение в целое число раз, нужно вычислить среднюю яркость квадрата из N×N точек и округлить ее до ближайшего доступного значения яркости. Однако такое округление приводит к сильным искажениям при выводе черно-белого изображения (рис. 2.34, слева).

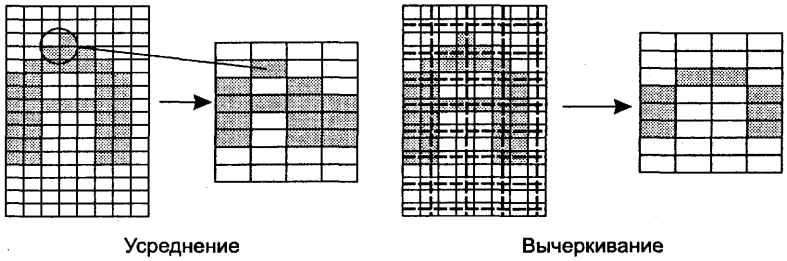


Рис. 2.34. Сжатие изображения едва раза различными способами

Другой простой способ сжатия — вычеркивание. В этом случае сохраняются лишь одна из N последовательно расположенных строк и одна из N колонок, а остальные зачеркиваются (рис. 2.34, справа). Основной недостаток данного способа состоит в том, что тонкие вертикальные и горизонтальные линии могут быть полностью потеряны: в приведенном примере таким образом стерта перекладина у буквы А.

Программа ShowFont, предназначенная для просмотра в увеличенном масштабе символов шрифта, извлеченного из знакогенератора видеоконтроллера, приведена в листинге 2.13. Программа использует две вспомогательные процедуры:

- процедура ShowRusFont отображает на экран шрифт в виде таблицы в масштабе 1:1 — каждому

биту маски соответствует на экране один пиксел;

- процедура ShowLargeChar осуществляет вывод символа в верхней части экрана (по центру) в увеличенном масштабе — каждому биту маски соответствует на экране квадрат размером 8×8 пикселов.

Листинг 2.13. Просмотр символов шрифта 8×16 в графическом режиме в увеличенном масштабе

```
IDEAL
P386
LOCALS
MODEL MEDIUM
; Код видеорежима 640x480, 256 цветов, с линейной
; адресацией видеобуфера
GraphicsMode equ 4101h
; Логическая ширина строки в пикселах
LogicalStringLength equ 1024
; Ширина экрана в пикселах
ScreenLength equ 640
; Высота экрана, строк
ScreenHeigth equ 480
; Подключить файл мнемонических обозначений
; кодов управляющих клавиш и цветовых кодов
include "list1_03.inc"
; Подключить файл макросов
include "list1_04.inc"
DATASEG
; Номер строки "активного" символа
ActiveCharString DW ?
; Номер колонки "активного" символа
ActiveCharColumn DW ?
; Текстовые сообщения
Text DB 10,13,"Выберите символ для просмотра "
     DB "в увеличенном масштабе:",0
     DB 25,14,"Для выбора символа в таблице "
     DB "используйте управляющие",0
     DB 26,14,"клавиши: '",18h,"','",19h
     DB "','",1Ah,"','",1Bh,"'.",0
PEsc DB 29,20,"Для выхода из программы нажмите ESC",0
ENDS

SEGMENT sseg para stack 'STACK'
DB 400h DUP(?)
ENDS
CODESEG
;****************************
;* Основной модуль программы *
;****************************
PROC ShowFont
        mov     AX,DGROUP
        mov     DS,AX
        mov     [CS:MainDataSeg],AX
; Установить текстовый режим
        mov     AX,3
        int     10h
; Установить режим прямой адресации памяти
        call    Initialization
; "Захватить" текстовый шрифт
        call    GrabRusFont
; Установить видеорежим
        call    SetVESAVideoMode
; Отобразить текстовые сообщения
        ; Установить черный цвет фона
        mov     [DefaultBackground],BLACK
        ; Установить зеленый цвет текста
        mov     [DefaultColor],LIGHTGREEN
        MGShowText 3,Text
        ; Установить желтый цвет текста
        mov     [DefaultColor],YELLOW
        MGShowString PEsc
; Инициализировать переменные
        mov     [ActiveCharString],0
        mov     [ActiveCharColumn],0

; ОСНОВНОЙ ЦИКЛ
@@GetCommand:
; Отобразить шрифт
        call    ShowRusFont
; Отобразить укрупненно "активный" символ
        call    ShowLargeChar
; Ввести команду
        call    GetChar
        cmp     AL,0        ;введена команда?
        jne     @@Error
; Проанализировать код команды и выполнить
; соответствующую операцию
        cmp     AH,B_Esc    ;"Выход"
        je      @@End
@@TestBS:
        cmp     AH,B_BS     ;"Стрелка влево"
        jne     @@TestFWD
        ; Проверить значение номера колонки
        ; (минимально допустимое значение - 0)
        cmp     [ActiveCharColumn],0
        je      @@Error
        ; Уменьшить номер колонки
        dec     [ActiveCharColumn]
        jmp short @@GetCommand
@@TestFWD:
        cmp     AH,B_FWD    ;"Стрелка вправо"
        jne     @@TestUp
        ; Проверить значение номера колонки
        ; (максимально допустимое значение - 31)
        cmp     [ActiveCharColumn],31
        jae     @@Error
        ; Увеличить номер колонки
        inc     [ActiveCharColumn]
        jmp short @@GetCommand
@@TestUp:
        cmp     AH,B_UP     ;"Стрелка вверх"
        jne     @@TestDn
            ; Проверить значение номера строки
            ; (минимально допустимое значение - 0)
            cmp     [ActiveCharString],0
            je      @@Error
            ; Уменьшить номер строки
            dec     [ActiveCharString]
            jmp short @@GetCommand
    @@TestDn:
            cmp     AH,B_DN     ;"Стрелка вниз"
            jne     @@Error
            ; Проверить значение номера строки
            ; (максимально допустимое значение - 7)
            cmp     [ActiveCharString],7
            jae     @@Error
            ; Увеличить номер строки
            inc     [ActiveCharString]
            jmp short @@GetCommand
    @@Error:
            ; Нврная команда - подать звуковой сигнал
            call    Beep
            jmp     @@GetCommand

    ; ВЫХОД ИЗ ПРОГРАММЫ
    @@End:  ; Установить текстовый режим
            mov     AX,3
            int     10h
            ; Выход в DOS
            mov     AH,4Ch
            int     21h
    ENDP ShowFont
```

Листинг 2.13 (продолжение)

```
;*****************************************
;*    ОТОБРАЗИТЬ ШРИФТ В ВИДЕ ТАБЛИЦЫ    *
;* Параметры передаются через переменные *
;* ActiveCharString и ActiveCharColumn   *
;*****************************************
PROC ShowRusFont near
        pushad
        mov     SI,offset Font8x16
        mov     EDI,[LinearVideoBuffer]
        add     EDI,LogicalStringLength*200+64+4
        mov     [FontString],0
@@m0:   ; Отобразить очередную строку символов
        mov     [FontColumn],0
@@m1:   ; Отобразить очередной символ
        mov     AH,16   ;число строк (байт) в маске символа
        mov     DL,0    ;цвет фона символа
        mov     DH,7    ;цвет символа
      mov     CX,[ActiveCharString]
      cmp     [FontString],CX
      jne     @@m2
      mov     CX,[ActiveCharColumn]
      cmp     [FontColumn],CX
      jne     @@m2
      mov     DL,1    ;цвет фона "активного" символа
      mov     DH,15   ;цвет "активного" символа
@@m2: ; Отобразить строку изображения символа
      mov     AL,[SI] ;загрузить очередной байт маски символа
      mov     CX,8
@@m3: ; Вывести на экран очередную точку изображения символа
      rol     AL,1
      jc      @@m4
      mov     [byte ptr GS:EDI],DL
      jmp short @@m5
@@m4: mov     [byte ptr GS:EDI],DH
@@m5: inc     EDI
      loop    @@m3
      inc     SI
      add     EDI,LogicalStringLength-8
      dec     AH
      jnz     @@m2
      sub     EDI,LogicalStringLength*16-16
      inc     [FontColumn]
      cmp     [FontColumn],32
      jb      @@m1
      add     EDI,LogicalStringLength*24-32*16
      inc     [FontString]
      cmp     [FontString],8
      jb      @@m0
      popad
      ret
ENDP ShowRusFont
;*************************************************
;* ОТОБРАЗИТЬ СИМВОЛ В УВЕЛИЧЕННОМ МАСШТАБЕ (8:1) *
;* Номер отображаемого символа определяется      *
;* переменными ActiveCharString и ActiveCharColumn *
;*************************************************
PROC ShowLargeChar near
      pushad
; Отобразить символ сверху, по центру экрана
      mov     EDI,[LinearVideoBuffer]
      add     EDI,(640-64)/2
; Вычислить положение маски символа в массиве шрифта
      ; Загрузить указатель на шрифт
      mov     SI,offset Font8x16
      ; Умножить номер строки на 32
      mov     AX,[ActiveCharString]
        shl     AX,5
        ; Прибавить номер колонки
        add     AX,[ActiveCharColumn]
        ; Умножить на размер символа в байтах (на 16)
        shl     AX,4
        add     SI,AX

; Отобразить символ в масштабе 8:1 (размер точки
; символа - 8x8 точек экрана)
        mov     DX,16     ;высота маски символа в точках
@@m1:   mov     AH,8      ;строку символа повторить 8 раз
@@m2:   ;Отображаем строку символа
        mov     AL,[SI]   ;прочитать байт маски
        mov     CX,8      ;ширина маски символа в точках
@@m3:   rol     AL,1
        jc      @@m4
        ;Отобразить 8 точек синего цвета
        mov     [dword ptr GS:EDI],01010101h
        add     EDI,4
        mov     [dword ptr GS:EDI],01010101h
        jmp short @@m5
@@m4:   ;Отобразить 8 точек желтого цвета
        mov     [dword ptr GS:EDI],0E0E0E0Eh
        add     EDI,4
        mov     [dword ptr GS:EDI],0E0E0E0Eh
@@m5:   add     EDI,4
        loop    @@m3
        ; Перейти на следующую строку изображения символа
        add     EDI,LogicalStringLength-64
        dec     AH
        jnz     @@m2
        ; Перейти на следующую строку маски символа
        inc     SI
        dec     DX
        jnz     @@m1
        popad
        ret
ENDP ShowLargeChar
ENDS

; Подключить процедуры ввода данных и вывода на экран
; в текстовом режиме
include "list1_02.inc"
; Подключить подпрограмму, переводящую сегментный
; регистр GS в режим линейной адресации
include "list2_01.inc"
; Подключить набор процедур общего назначения,
; предназначенных для установки графических
; видеорежимов и работы в них
include "list4_02.inc"
; Подключить набор процедур вывода текста,
; предназначенных для 256-цветных режимов
include "list4_03.inc"

END
```

2.2.6.2.7. Анимация двухмерных изображений

Анимация в двухмерном режиме, то есть создание на экране иллюзии движения плоских изображений (спрайтов), также реализуется при помощи масок [13]. Однако если для неподвижного объекта (например, буквы шрифта) достаточно одной маски, то движущийся объект требует отдельную маску для каждой фазы движения в каждом из возможных направлений. Чтобы получить приличную иллюзию движения, необходимо отобразить от 8 до 12 фаз. Человеческий глаз способен различать углы менее одного градуса, поэтому, чтобы создать иллюзию вращения крупного объекта или движения его по произвольным направлениям, нужно иметь по маске на каждый из возможных трехсот шестидесяти градусов поворота.

Сложные объекты способны совершать разнообразные виды движений, для каждого из которых нужен

собственный комплект масок. Объект типа «человек», например, может идти, ползти, прыгать, приседать, взбираться по лестнице, плыть и т. п. Механические объекты обычно проще, чем живые, — требуют меньшего количества фаз и разновидностей движения (поэтому их так любят использовать разработчики игр).

Современные компьютерные игры (например, стратегические) предусматривают одновременный вывод на экран нескольких сотен или даже тысяч объектов (подвижных и неподвижных). Можно сформировать группы, считая однотипными те объекты, для отображения которых пригодны одинаковые комплекты масок, однако количество групп все равно измеряется десятками. Одновременно на экран выводится до десяти типов танков, несколько типов самолетов, несколько типов солдат и т. д. В результате общее количество масок, которое необходимо хранить в памяти компьютера, является произведением количества групп объектов, умноженного на количество возможных видов движений объекта, на количество фаз движений и на количество направлений движения.

Рассмотрим следующий пример: пусть имеется 10 типов объектов, способных совершать по два вида движений; при отображении объектов используется 10 фаз, объекты могут двигаться по 50 направлениям. Тогда в общей сложности нужно десять тысяч масок! Каждую маску необходимо предварительно нарисовать, для чего применяются специальные анимационные программные пакеты (проводить такую работу ручным способом слишком дорого и долго). Кроме того, одна маска сравнительно небольшого объекта размером 32×32 пиксела занимает от 1 до 4 Кбайт памяти, а десять тысяч масок требуют соответственно 10-40 Мбайт!

Как следует из приведенного примера, при разработке компьютерных игр и других программ, использующих анимацию, приходится искусственно ограничивать количество типов объектов, видов и направлений движения.

В общем случае цикл вывода движущегося объекта включает следующую последовательность операций.

1. Вычисление координат области отображения объекта.
2. Сохранение фона области отображения.
3. Рисование текущей фазы движения объекта.
4. Ожидание начала обратного хода луча по кадру.
5. Стирание изображения объекта путем восстановления фона.

Для сохранения фона нужно выделить область памяти такого же размера, что и для маски отображаемой фазы движения. Если маска объекта имеет размер M×N пикселов, а цвет пиксела кодируется K байтами, то область сохранения фона занимает M×N×K байт. При работе с группой движущихся объектов необходимо запоминать последовательность их рисования на экране, поскольку стирание должно проводиться в порядке, обратном порядку вывода.

Вывод изображения на экран обязательно должен быть синхронизирован с началом хода луча по кадру — в противном случае изображение движущегося объекта будет искажаться всякий раз, когда совпадают момент перерисовки и вывода объекта на экран. Если в режиме анимации используется только одна видеостраница, то обязательно необходимо учитывать скорость вывода изображения в видеопамять — даже при наличии синхронизации с ходом луча по кадру и высокой скорости вывода данных в верхней части картинки образуется зона искажений. В зависимости от алгоритма вывода изображений возможны два основных типа искажений — разрезание картинки или непрорисовка некоторых ее участков (рис. 2.35).

Ширина зоны искажений определяется количеством и размером динамических (движущихся или изменяющихся) объектов, их размерами и скоростью вывода изображения на экран. Зона искажений заканчивается в том месте экрана, где при любом возможном количестве и порядке вывода объектов центральный процессор заведомо опережает луч электронно-лучевой трубки (ЭЛТ). Обычно изготовители компьютерных игр идут на маленькую хитрость — занимают верхнюю часть экрана широким статическим изображением (например, графическим меню).

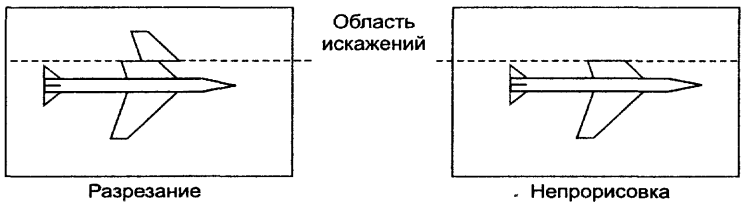


Рис. 2.35. Основные виды искажений при выводе движущихся объектов

При использовании одной видеостраницы движение объектов по экрану будет выглядеть плавным только в том случае, если процессор успевает полностью перерисовывать изображение в течение одного кадра ЭЛТ. Поскольку частота кадров у современных мониторов обычно составляет 65 - 100 Гц, процессор должен успеть перерисовать кадр за 10 - 15 миллисекунд, причем для каждого объекта нужно (по изложенному выше алгоритму) успеть:

- восстановить фон в предыдущей позиции;
- сохранить фон в новой позиции;
- нарисовать изображение.

Кроме того, допустимая высота зоны искажений обычно составляет не более 1/5 высоты экрана: пока луч находится в этой зоне, процессор должен успеть перерисовать все подвижные объекты в остальной части экрана.

Проведем оценочные расчеты при условии, что скорость вывода данных составляет 10 млн пикселов в секунду. За время одного кадра (15 мс) процессор успеет перерисовать 150 тыс. пикселов, но это число нужно поделить на 3 (для каждого объекта — 3 операции) и на 5 (под зону искажений выделена 1/5 часть экрана). В результате получается, что сумма размеров (в байтах) всех массивов-масок не должна превышать 10 Кбайт. Например, если размер маски равен 1 Кбайт (небольшой объект 32×32 в 256-цветном режиме), то можно вывести 10 движущихся объектов, а при размере маски 4 Кбайт (32×32, режим TrueColor) — всего 2 объекта!

Чтобы ускорить вывод данных, разработчикам программ приходится применять множество разнообразных хитростей.

Первая хитрость основана на том, что современные компьютеры имеют большой объем оперативной памяти, и фон можно в ней хранить целиком, а не кусочками. Ускорение работы при этом получается даже больше, чем на треть, поскольку операция чтения из видеопамяти выполняется медленнее, чем запись в нее (видеоконтроллеры всегда оптимизированы по записи).

Второй трюк заключается в том, чтобы избавиться от операций контроля пересечения границ экрана при выводе маски объекта. Если видеопамять организована обычным образом, а контроля пересечения границ нет, то возможно появление ложных изображений вдоль левой и правой границ экрана (рис. 2.36, сверху) или зависание программы при выходе за пределы видеопамяти. Проблема легко решается, если объем видеопамяти позволяет установить логическую длину строки больше физической — при выходе за боковые границы маска попадает в неотображаемую область памяти (рис. 2.36, снизу). Кроме того, выравнивание логической длины строки на 2^N пикселов позволяет (хотя и весьма незначительно) упростить расчет координат точек изображения.

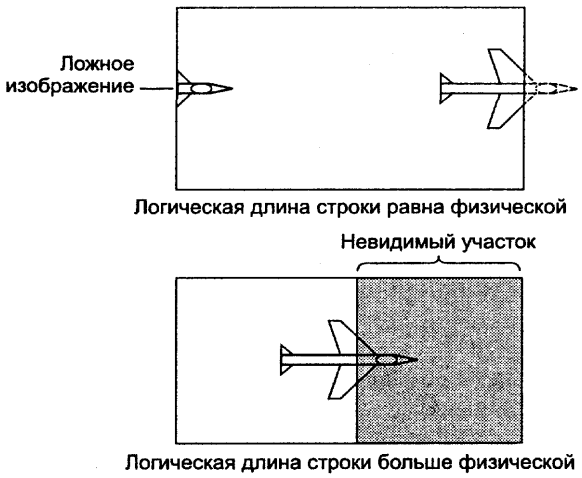


Рис. 2.36. Использование невидимого участка видеопамяти для подавления ложных изображений

Рисунок 2.36 создает впечатление, что ложное изображение подавляется только при выходе за правую границу. На самом деле подавление происходит и в случае выхода за левый край экрана: хорошей моделью развертки линейной памяти контроллера на экран является спираль (пружина), показанная на рис. 2.37. При выходе за любую боковую границу маска попадает на невидимый участок.

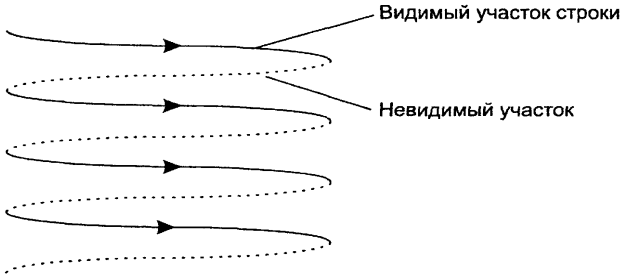


Рис. 2.37. Модель развертки видеопамяти на экран монитора

Можно избавиться от необходимости контролировать пересечение не только боковых, но также верхней

и нижней границ. С этой целью создаются горизонтальные неотображаемые области, точнее — горизонтальная область над страницей, поскольку невидимая область под страницей присутствует изначально (см. 2.38). Ширина вертикальной невидимой области должна быть больше или равна максимальной возможной ширине маски L_{max} Высота горизонтальной области должна быть равна максимальной возможной высоте маски H_{max}

К сожалению, трюки с защитными полосами применимы не всегда — очень часто по крайней мере одну из сторон изображения занимает полоса меню управления. Проще всего контролировать нижнюю границу спрайта, поэтому (если имеется возможность выбора) лучше всего размещать полосу меню снизу.

Самый эффективный способ устранения искажений анимированного изображения состоит в использовании упомянутого выше механизма переключения страниц: пока видеоконтроллер отображает на экран одну страницу, процессор перерисовывает другую. Однако при этом расход видеопамяти увеличивается вдвое — на старых видеоконтроллерах ее может оказаться недостаточно для создания второй страницы. Кроме того, невидимые области также съедают заметный объем памяти. Организация видеопамяти в режиме переключения страниц при наличии невидимых областей показана на рис. 2.38.

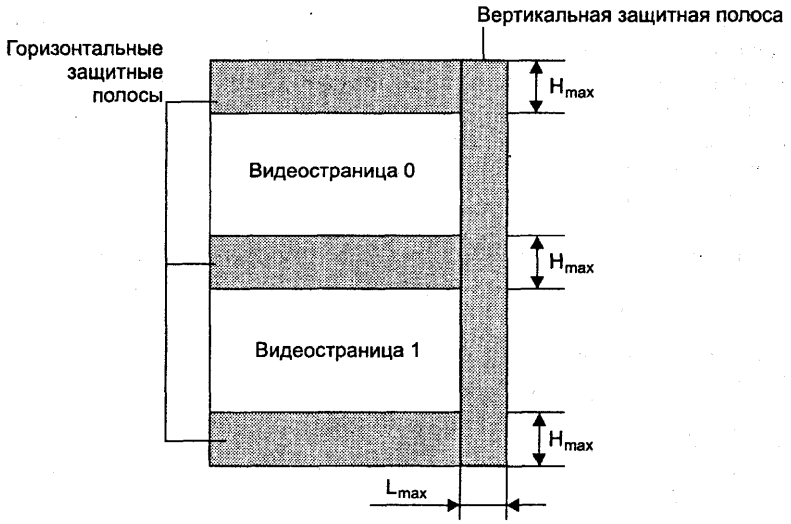


Рис. 2.38. Организация видеопамяти в режиме переключения страниц при наличии защитных неотображаемых полос

При переключении страниц требования к скорости перерисовки кадра значительно ниже, поскольку процессор может заниматься выводом данных в течение всего времени хода луча по кадру. Кроме того, уже не обязательно перерисовывать кадры со скоростью их вывода на экран монитора, то есть переключение страниц можно выполнять не в каждом кадре, а через один или через два (но обязательно — через одинаковое число кадров). Для создания приемлемого изображения достаточно генерировать 25 - 30 кадров в секунду, то есть на перерисовку каждого кадра можно тратить 30 - 40 миллисекунд. В 256-цветном режиме с разрешением 640×480 точек за это время можно полностью перерисовать весь экран!

Программа, реализующая анимацию в одностраничном режиме, показана в листингах 2.14 - 2.20. Листинг 2.14 — это усовершенствованная версия программы из листинга 2.1, реализующая доступ к видеопамяти и дополнительной оперативной памяти через раздельные сегменты большого размера с контролем границ. Пределы сегментов установлены в расчете на минимальную конфигурацию современных ПК — 4 Мбайт ОЗУ и 2 Мбайт видеопамяти, но в примере используется только часть (меньше половины) выделенного пространства. В принципе, размеры указанных сегментов можно задавать динамически (в соответствии с конфигурацией конкретного компьютера), однако недавно возникла проблема определения объема ОЗУ, если он больше 64 Мбайт (предел, на который были когда-то рассчитаны стандартные функции BIOS). Процедура GInitialization использует для загрузки теневых регистров вспомогательную подпрограмму SetSegAddrModeForFSGS, а затем разблокирует линию A20 при помощи подпрограмм Enable_A20 и устанавливает обработчик прерывания по нарушению границ сегментов MemoryProtectionInterrupt при помощи процедуры SetProtectionInterrupt.

Листинг 2.14. Подпрограмма, выполняющая настройку регистра FS на видеопамять, регистра GS — на дополнительную память (list4_14.inc)

```
; Порт, управляющий запретом немаскируемых прерываний
CMOS_ADDR      equ  0070h
CMOS_DATA      equ  0071h
; Селекторы сегментов
SYS_PROT_CS    equ  0008h
SYS_REAL_SEG   equ  0010h
SYS_VIDEO_SEG  equ  0018h
SYS_LINEAR_SEG equ  0020h
```

Листинг 2.14 (продолжение)

```
DATASEG
; Текстовые сообщения
ProtErr DB LIGHTRED,12,0
        DB "Ошибка: нарушение границ сегмента памяти",0
        DB YELLOW,24,29,"Нажмите любую клавишу",0
; Область сохранения старого вектора прерывания
; по IRQ5 и защите памяти
OldProtInterruptOffset  DW ?
OldProtInterruptSegment DW ?
ENDS
CODESEG
;******************************
;* Подготовка системы к работе *
;******************************
PROC GInitialization NEAR
        pushad
; Занести линейный адрес видеопамяти (для FS)
        mov     EAX,[LinearVideoBuffer]
        mov     [word ptr CS:GDT+26],AX
        ror     EAX,16
        mov     [byte ptr CS:GDT+28],AL
        mov     [byte ptr CS:GDT+31],AH
; Установить предел видеопамяти 2 Мб
        mov     EAX,2*100000h
        shr     EAX,12 ;поделить на гранулярность (4 Кб)
        dec     EAX    ;уменьшить на 1
        mov     [word ptr CS:GDT+24],AX
        ror     EAX,16
        or      [byte ptr CS:GDT+30],AL
; Занести линейный адрес расширенной памяти (для GS)
        mov     EAX,110000h       ;1 Мб + 64 Кб
        mov     [word ptr CS:GDT+34],AX
        ror     EAX,16
        mov     [byte ptr CS:GDT+36],AL
        mov     [byte ptr CS:GDT+39],AH
; Установить предел расширенной памяти
; (4 Мб минус 1,064 Мб)
        mov     EAX,4*100000h-110000h
        shr     EAX,12 ;поделить на гранулярность (4 Кб)
        dec     EAX    ;уменьшить на 1
        mov     [word ptr CS:GDT+32],AX
        ror     EAX,16
        or      [byte ptr CS:GDT+38],AL
; Сохранить значения сегментных регистров в
; реальном режиме (кроме GS)
        mov     [CS:Save_SP],SP
        mov     AX,SS
        mov     [CS:Save_SS],AX
        mov     AX,DS
        mov     [CS:Save_DS],AX
; (работаем теперь только с кодовым сегментом)
        mov     AX,CS
        mov     [word ptr CS:Self_Mod_CS],AX
        mov     DS,AX
        cli
        mov     SS,AX
        mov     SP,offset Local_Stk_Top
        sti
; Установить режим линейной адресации
        call    SetSegAddrModeForFSGS

; Восстановить значения сегментных регистров
        cli
        mov     SP,[CS:Save_SP]
        mov     AX,[CS:Save_SS]
        mov     SS,AX
        mov     AX,[CS:Save_DS]
        mov     DS,AX
        sti
; Разрешить работу линии A20
        call    Enable_A20
; Замаскировать прерывание IRQ5 и установить
; обработчик прерывания по нарушению границ сегментов
        call    SetProtectionInterrupt
        popad
        ret
ENDP GInitialization

; Область сохранения значений сегментных регистров
Save_SP DW ?
Save_SS DW ?
Save_DS DW ?
; Указатель на GDT
GDTPtr  DQ ?
; Таблица дескрипторов сегментов для
; входа в защищенный режим
GDT DW 00000h,00000h,00000h,00000h ;не используется
    DW 0FFFFh,00000h,09A00h,00000h ;сегмент кода CS
    DW 0FFFFh,00000h,09200h,00000h ;сегмент данных DS
    DW 00000h,00000h,09200h,000F0h ;сегмент FS
    DW 00000h,00000h,09200h,000F0h ;сегмент GS
; Локальный стек для защищенного режима
; (организован внутри кодового сегмента)
label GDTEnd word
        DB 255 DUP(0FFh)
Local_Stk_Top DB (0FFh)
;*****************************************
;* Процедура, изменяющая содержимое теневых *
;* регистров FS и GS                        *
;*****************************************
PROC SetSegAddrModeForFSGS near
; Вычислить линейный адрес кодового сегмента
        mov     AX,CS
        movzx   EAX,AX
        shl     EAX,4   ;умножить номер параграфа на 16
        mov     EBX,EAX ;сохранить линейный адрес в EBX
; Занести младшее слово линейного адреса в дескрипторы
; сегментов кода и данных
        mov     [word ptr CS:GDT+10],AX
        mov     [word ptr CS:GDT+18],AX
        ; Переставить местами старшее и младшее слова
        ror     EAX,16
; Занести биты 16-23 линейного адреса в дескрипторы
; сегментов кода и данных
        mov     [byte ptr CS:GDT+12],AL
        mov     [byte ptr CS:GDT+20],AL

; Установить предел (Limit) и базу (Base) для GDTR
        add     EBX, offset GDT
        mov [word ptr CS:GDTPtr],(offset GDTEnd-GDT-1)
        mov     [dword ptr CS:GDTPtr+2],EBX
; Сохранить регистр флагов
        pushf
; Запретить прерывания, так как таблица прерываний IDT
; не сформирована для защищенного режима
        cli
; Запретить немаскируемые прерывания NMI
        in      AL,CMOS_ADDR
        mov     AH,AL
        or      AL,080h      ;установить старший разряд
        out     CMOS_ADDR,AL ;не затрагивая остальные
        and     AH,080h
        ; Запомнить старое состояние маски NMI
        mov     CH,AH
; Перейти в защищенный режим
        lgdt    [fword ptr CS:GDTPtr]
        mov     BX,CS    ;запомнить сегмент кода
        mov     EAX,CR0
        or      AL,01b   ;установить бит PE
        mov     CR0,EAX  ;защита разрешена
        ; Безусловный дальний переход на метку SetPMode
        ; (очистить очередь команд и перезагрузить CS)
                DB      0EAh
                DW      (offset SetPMode)
                DW      SYS_PROT_CS
SetPMode:
```

Листинг 2.14 (продолжение)

```
        ; Подготовить границы сегментов
        mov     AX,SYS_REAL_SEG
        mov     SS,AX
        mov     DS,AX
        mov     ES,AX
        ; Настроить сегмент FS на видеопамять
        mov     AX,SYS_VIDEO_SEG
        mov     FS,AX
        ; Настроить сегмент GS на расширенную память
        mov     AX,SYS_LINEAR_SEG
        mov     GS,AX
; Вернуться в реальный режим
        mov     EAX,CR0
        and     AL,11111110b ;сбросить бит PE
        mov     CR0,EAX      ;защита отключена
        ; Безусловный дальний переход на метку SetRMode
        ; (очистить очередь команд и перезагрузить CS)
            DB 0EAh
            DW (offset SetRMode)
Self_Mod_CS DW ?
SetRMode:
        ; Регистры стека и данных
        ; настроить на сегмент кода
        mov     SS,BX
        mov     DS,BX
        ; Обнулить ES
    xor     AX,AX
    mov     ES,AX
    ; Возврат в реальный режим,
    ; прерывания снова разрешены
    in      AL,CMOS_ADDR
    and     AL,07Fh
    or      AL,CH
    out     CMOS_ADDR,AL
    popf
    ret
ENDP SetSegAddrModeForFSGS

;****************************************
;* Разрешить работу с памятью выше 1 Мб *
;****************************************
PROC Enable_A20 near
    call    Wait8042BufferEmpty
    mov     AL,0D1h ;команда управления линий A20
    out     64h,AL
    call    Wait8042BufferEmpty
    mov     AL,0DFh ;разрешить работу линии
    out     60h,AL
    call    Wait8042BufferEmpty
    ret
ENDP Enable_A20

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

;**********************************
;* УСТАНОВИТЬ ВЕКТОР ПРЕРЫВАНИЯ ПО *
;*   СРАБАТЫВАНИЮ ЗАЩИТЫ ПАМЯТИ    *
;**********************************
PROC SetProtectionInterrupt NEAR
    pusha
    push    ES
        mov     AX,[CS:MainDataSeg]
        mov     DS,AX
; Замаскировать прерывание IRQ5
        cli
        in      AL,21h
        or      AL,100000b
        out     21h,AL
        sti
; Установить вектор прерывания
        ; Настроить ES на область векторов
        mov     AX,0
        mov     ES,AX
        ; Сохранить старый вектор
        mov     AX,[ES:13*4]
        mov     [OldProtInterruptOffset],AX
        mov     AX,[ES:13*4+2]
        mov     [OldProtInterruptSegment],AX
        ; Установить новый вектор
        cli
        mov     AX,offset MemoryProtectionInterrupt
        mov     [ES:13*4],AX
        mov     AX,CS
        mov     [ES:13*4+2],AX
        sti
        pop     ES
        popa
        ret
ENDP SetProtectionInterrupt

;*************************************
;* ВОССТАНОВИТЬ СТАРЫЙ ВЕКТОР ЗАЩИТЫ *
;*************************************
PROC RestoreOldProtectionInterrupt NEAR
        pusha
        push    ES
        ; Настроить ES на область векторов
        mov     AX,0
        mov     ES,AX
        ; Восстановить старый вектор
        cli
        mov     AX,[OldProtInterruptOffset]
        mov     [ES:13*4],AX
        mov     AX,[OldProtInterruptSegment]
        mov     [ES:13*4+2],AX
        sti
        pop     ES
        popa
        ret
ENDP RestoreOldProtectionInterrupt

;*************************************
;*   НОВЫЙ ОБРАБОТЧИК ПРЕРЫВАНИЯ ПО  *
;* НАРУШЕНИЮ ГРАНИЦ СЕГМЕНТОВ ПАМЯТИ *
;*************************************
PROC MemoryProtectionInterrupt NEAR
        mov     AX,[CS:MainDataSeg]
        mov     DS,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
```

Листинг 2.14 (продолжение)

```
        MShowColorText 2,ProtErr
; Ожидать нажатия клавиши
        call    GetChar
; Восстановить старый обработчик прерывания
        call    RestoreOldProtectionInterrupt
; Выход в DOS
        mov     AH,4Ch
        int     21h
ENDP MemoryProtectionInterrupt
ENDS
```

После выполнения процедуры GInitialization к видеопамяти можно обращаться через сегментный регистр FS, а к дополнительной оперативной памяти — через GS. В случае нарушения установленных границ сегментов происходит прерывание общей защиты памяти, и процедура MemoryProtectionInterrupt выдает сообщение об ошибке, а затем осуществляет аварийное завершение работы программы с немедленным выходом в DOS.

ВНИМАНИЕ ——————————————————————————————

Система защиты памяти и параллельный порт LPT2 настроены на один и тот же вектор, поэтому сигнал от LPT2 обязательно нужно маскировать при установке обработчика сигнала защиты памяти.

Листинг 2.15 является видоизмененным для сегментной адресации вариантом листинга 2.4. Он включает две подпрограммы: процедуру вывода символа PutGraChar и процедуру очистки экрана GClearScreen для 256-цветных режимов с линейным видеобуфером.

Листинг 2.15. Процедуры вывода символа и очистки экрана для 256-цветных режимов с линейной адресацией видеобуфера (list4_15.inc)

```
DATASEG
; Цвет текста в графическом режиме по умолчанию
DefaultColor      DB WHITE  ;белый
; Цвет фона в графическом режиме по умолчанию
DefaultBackground DB BLACK  ;черный
ENDS
CODESEG
;*****************************************************
;* ВЫВОД СИМВОЛА 8x16 НА ЭКРАН В ГРАФИЧЕСКОМ РЕЖИМЕ *
;*             (для 256-цветных режимов)             *
;* Все параметры передаются через регистры:          *
;* AL - ASCII-код символа;                           *
;* DH - номер текстовой строки экрана;               *
;* DL - номер текстовой колонки экрана;              *
;* Используются цвет символов и цвет фона,           *
;* заданные по умолчанию.                            *
;*****************************************************
PROC PutGraChar NEAR
        push    DS
        pushad
        mov     CX,[CS:MainDataSeg]
        mov     DS,CX
        cld
        ; Смещение символа от начала шрифта
        mov     SI,offset Font8x16
        xor     AH,AH
        shl     AX,4
        add     SI,AX
        ; Вычислить левый верхний угол символа
        xor     EBX,EBX
        mov     BL,DH
        shl     EBX,14 ;умножить номер строки на 16*1024
        xor     DH,DH
        shl     DX,3   ;умножить номер столбца на 8
        or      BX,DX
        mov     EDI,EBX
        mov     BL,[DefaultColor]
        mov     DL,[DefaultBackground]
        mov     AH,16 ;счетчик строк маски буквы
@@M0:   lodsb
        mov     CX,8  ;счетчик точек в строке маски
@@M1:   rol     AL,1
        jc      @@M2
        mov     [FS:EDI],DL
        jmp     @@M3
@@M2:   mov     [FS:EDI],BL
@@M3:   inc     EDI
        loop    @@M1
        add     EDI,LogicalStringLength-8
        dec     AH
        jnz     @@M0
;Завершение процедуры
@@EndPutGraChar:
        popad
        pop     DS
        inc     DL
        ret
ENDP PutGraChar
;*************************************
;* ОЧИСТКА ЭКРАНА В ГРАФИЧЕСКОМ РЕЖИМЕ *
;*   (процедура параметров не имеет)   *
;*************************************
PROC GClearScreen NEAR
        pushad
; Умножить высоту экрана ScreenHeigth на логическую
; ширину строки (1024 пиксела)
        mov     ECX,ScreenHeigth
        shl     ECX,10
        mov     EDI,0
; Заполнить видеопамять нулями
        mov     AL,0     ;черный цвет
@@NextPixels:
        mov     [FS:EDI],AL
        inc     EDI
        dec     ECX
        jnz     @@NextPixels
        popad
        ret
ENDP GClearScreen
ENDS
```

В листинге 2.16 приведено описание констант, глобальных переменных и макрокоманд, используемых в листингах 2.18 - 2.20. Макрокоманды в данном случае делают текст значительно компактнее и практически не маскируют особенности работы программы. Макрос DrawMImage осуществляет подготовку параметров и вызов функции рисования движущегося изображения DrawMovingImage, макрос DeleteMImage — подготовку параметров и вызов функции стирания движущегося изображения DeleteImage, а макрос DrawSImage — подготовку параметров и вызов функции рисования статического изображения DrawStaticImage.

Объем данных в приведенном примере невелик, и потому все массивы размещены в основной области памяти DOS. В программах с большим размером структур данных можно свободно использовать для хранения переменных и массивов расширенную память, однако следует учитывать, что вся ответственность за распределение расширенной памяти при этом возлагается на программиста. Обычно количество простых переменных не превышает нескольких тысяч, и для их хранения лучше все-таки применять обычную память, автоматически распределяемую программой-компилятором. В расширенной памяти удобно хранить большие объекты (например, статический фон для экранного изображения) и группы объектов с повторяющейся структурой (например, маски различных фаз движения в программах с двумерной анимацией). При необходимости также можно из специальных файлов загружать прямо в расширенную память крупные текстовые сообщения (используемые обычно в режиме подсказки).

Листинг 2.16. Описание констант, глобальных переменных и макрокоманд (list4_16.inc)

```
   ; КОНСТАНТЫ
   ; Режим с разрешением 640x480, 256 цветов
   GraphicsMode equ 4101h
   ; Логическая ширина строки в пикселах
   LogicalStringLength equ 1024
   ; Ширина экрана в пикселах
   ScreenLength equ 640
   ; Высота экрана, строк
   ScreenHeigth equ 480
   ; Количество "облаков" на небе (облака накладываются
   ; друг на друга)
   MaxCloudNum equ 10
   ; Число кадров взлета (разгона) ракеты
   MaxRktStartFrame equ 56
   ; Число кадров взрыва
   MaxExplFrameNumber equ 10
   ; Число осколков при взрыве
   SplinterMaxNumber equ 40
   ; Радиус "разлета осколков" при взрыве
   ExpR equ 32
   ; Квадрат расстояния "подрыва" (от центра ракеты
   ; до цента цели)
   TargetDistanceSQ equ 576  ;(расстояние подрыва 24)
   ; Максимально допустимое количество
   ; пропущенных самолетов
   MaxSavedPlanes equ 3
   ; Максимальное число эпизодов игры
MaxEpNumber equ 100

; МАКРОСЫ
; Вызов функции рисования движущегося объекта
MACRO DrawMImage ObjS,ObjC,ObjL,ObjH,Obj,ObjA,ObjF
        ; Передать параметры подпрограмме рисования
        mov     AL,[ObjF]
        mov     [ImageF],AL
        mov     EAX,[ObjS]
        mov     [ImageS],EAX
        mov     EAX,[ObjC]
        mov     [ImageC],EAX
        mov     [dword ptr ImageL],ObjL
        mov     [dword ptr ImageH],ObjH
        mov     [ImageMaskOffset], offset Obj
        ; Нарисовать объект
        call    DrawMovingImage
        ; Запомнить адрес изображения в видеопамяти
        mov     EAX,[ImageA]
        mov     [ObjA],EAX
ENDM
; Вызов функции восстановления фона
; (стирание изображения объекта)
MACRO DeleteMImage ObjA,ObjL,ObjH,ObjF
        mov     EAX,[ObjA]
        mov     [ImageA],EAX
        mov     [dword ptr ImageL],ObjL
        mov     [dword ptr ImageH],ObjH
        mov     AL,[ObjF]
        mov     [ImageF],AL
        call    DeleteImage
ENDM
; Вызов функции рисования неподвижного объекта
MACRO DrawSImage ObjS,ObjC,ObjL,ObjH,Obj
        ; Передать параметры подпрограмме рисования
        mov     EAX,[ObjS]
        mov     [ImageS],EAX
        mov     EAX,[ObjC]
        mov     [ImageC],EAX
        mov     [dword ptr ImageL],ObjL
        mov     [dword ptr ImageH],ObjH
        mov     [ImageMaskOffset], offset Obj
        ; Нарисовать объект
        call    DrawStaticImage
ENDM

DATASEG
; ОСНОВНЫЕ ПЕРЕМЕННЫЕ СОСТОЯНИЯ ИГРЫ
; Счетчик игровых эпизодов
   EpisodeNumber DW ?
   ; Счетчик игрового времени (число видеокадров
   ; от начала игрового эпизода)
   GameTimeCounter DW ?
   ; Флаг изменения состояния игры
   GameStateGhange DB ?
   ; Состояние самолета (0 - ожидание, 1 - полет,
   ; 2 - взрыв)
   PlnState     DB ?
   ; Состояние ракеты (0 - ожидание запуска, 1 - запуск,
   ; 2 - полет, 3 - взрыв, 4 - разлет осколков,
   ; 5 - ракеты нет на экране)
   RktState     DB ?
   ; Счетчик самолетов
   PlnCounter   DW ?
   ; Счетчик сбитых самолетов
   DestroedPlns DW ?
   ; Счетчик пропущенных самолетов
   EscPlns      DW ?
   ; Счетчик потраченных ракет
   RktCounter   DW ?
```

Листинг 2.16 (продолжение)

```
; ИНФОРМАЦИЯ ОБ ОТОБРАЖАЕМЫХ ОБЪЕКТАХ
; Позиция маски самолета на экране
PlnS   DD ?  ;строка
PlnC   DD ?  ;колонка
PlnA   DD ?  ;текущий линейный адрес
PlnF1  DB ?  ;флаг наличия объекта в текущей странице
; Признак попадания ракеты
HitFlag DW ?
; Скорость движения самолета
PlaneSpeed DD ?
; Направление движения самолета
PlaneDirection DB ?
; Задержка пуска самолета
PlaneDeltaT DW ?
; Позиция маски ракеты на экране
RktS   DD ?  ;строка
RktC   DD ?  ;колонка
RktA   DD ?  ;текущий линейный адрес
RktF1  DB ?  ;флаг наличия объекта в текущей странице
RktStartFrameNumber DW ? ;номер кадра старта ракеты
; Позиция маски пламени на экране
FlmS   DD ?  ;строка
FlmA   DD ?  ;текущий линейный адрес
FlmF1  DB ?  ;флаг наличия объекта в текущей странице
; Параметры взрыва
ExpS   DD ?  ;строка
ExpC   DD ?  ;колонка
ExpA   DD ?  ;текущий линейный адрес
ExpF1  DB ?  ;флаг наличия объекта в текущей странице
ExpFrameNumber DW ? ;номер кадра взрыва
; Параметры дерева
TreeS    DD ?  ;строка
TreeC    DD ?  ;колонка
; Параметры облака
CloudS   DD ?  ;строка
CloudC   DD ?  ;колонка
; Позиция пусковой установки на экране
PRS      DD ?  ;строка
PRC      DD ?  ;колонка
; ТЕКСТОВЫЕ СООБЩЕНИЯ
EndTxt DB 0,32,"РЕЗУЛЬТАТЫ ИГРЫ",0
       DB 12,0,"Сыграно эпизодов:",0
       DB 14,0,"Выпущено ракет:",0
       DB 16,0,"Сбито самолетов:",0
       DB 18,0,"Пропущено самолетов:",0
       DB 29,29,"Нажмите любую клавишу",0
; ДИНАМИЧЕСКИЕ МАСКИ
PlnMask DB 1024 DUP(?)   ;текущая маска самолета
ExpMask DB 4096 DUP(?)   ;динамическая маска взрыва
ENDS
```

Листинг 2.17 включает в себя маски статических (дерево, облако, пусковая установка) и динамических (самолет, ракета, пламя из сопла ракеты) объектов, а также массивы координат статических объектов и массивы случайных параметров для динамических объектов. Каждая строка данных в масках широких объектов (самолета и облака) поделены на две части, в результате чего утрачена наглядность, однако все остальные объекты просматриваются достаточно четко.

Листинг 2.17. Набор масок объектов, массивов координат и случайных чисел для игры «Самолет и ракета» (list4_17.inc)

```
DATASEG
; Маска самолета
PlnML equ 32  ; ширина маски самолета
PlnMH equ 32  ; высота маски самолета
Plane DB 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
      
```

Листинг 2.17 (продолжение)

```
; Маска ракеты
RktML equ 16 ; ширина маски ракеты
RktMH equ 32 ; высота маски ракеты
Rkt DB  0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 0, 7,15, 7, 0, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 0, 7,15, 7, 0, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 0, 7,15, 7, 0, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 7,15,15,15, 7, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 7,15,15,15, 7, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 7,15,15,15, 7, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 6,10, 2, 6, 2, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 2, 2, 6, 2, 2, 2, 2, 0, 0, 0, 0, 0
    DB  0, 0, 0, 2, 2, 2,10, 2,14, 2, 2, 2, 0, 0, 0, 0
    DB  0, 0, 2, 6, 2, 2, 2, 2, 2, 2,14, 2, 2, 0, 0, 0
    DB  0, 2, 6,14, 6, 2, 6, 2, 6, 2, 6,10, 6, 2, 0, 0
    DB  2, 2, 2, 2, 2, 2, 2, 2,10, 2, 2, 2, 2, 2, 2, 0
    DB  0, 0, 0, 0, 0, 2, 6, 2,14, 2, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 2, 6,10, 6, 2, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 2,14, 6, 2, 2, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 2, 6, 2,10, 2, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 2,10,14, 6, 2, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 2, 6, 2, 6, 2, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 2,14, 6, 2, 2, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 2, 6, 2,14, 2, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 2,10, 6,10, 2, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 0, 2,14, 2, 2, 2, 0, 0, 0, 0, 0, 0
    DB  0, 0, 0, 0, 2, 2,10, 2, 6, 2, 2, 0, 0, 0, 0, 0
    DB  0, 0, 0, 2, 6, 2, 2, 2,14, 2,14, 2, 0, 0, 0, 0
    DB  0, 0, 2,14, 2, 2, 6, 2,10, 2, 6, 2, 2, 0, 0, 0
    DB  0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0

; Маска пламени из сопла ракеты
FlmML equ 16  ; ширина маски пламени
FlmMH equ 16   ; высота маски пламени
    Flm DB  0, 0, 0, 0, 0,12,15,15,15,12, 0, 0, 0, 0, 0, 0
        DB  0, 0, 0, 0, 0,12,15,15,15,12, 0, 0, 0, 0, 0, 0
        DB  0, 0, 0, 0, 0,12,14,15,14,12, 0, 0, 0, 0, 0, 0
        DB  0, 0, 0, 0, 0,12,14,14,14,12, 0, 0, 0, 0, 0, 0
        DB  0, 0, 0, 0, 0,12,14,14,14,12, 0, 0, 0, 0, 0, 0
        DB  0, 0, 0, 0, 0,12,14,14,14,12, 0, 0, 0, 0, 0, 0
        DB  0, 0, 0, 0, 0, 4,12,14,12, 4, 0, 0, 0, 0, 0, 0
        DB  0, 0, 0, 0, 0, 4,12,14,12, 4, 0, 0, 0, 0, 0, 0
        DB  0, 0, 0, 0, 0, 4,12,14,12, 4, 0, 0, 0, 0, 0, 0
        DB  0, 0, 0, 0, 0, 4,12,14,12, 4, 0, 0, 0, 0, 0, 0
        DB  0, 0, 0, 0, 0, 0, 4,12, 4, 0, 0, 0, 0, 0, 0, 0
        DB  0, 0, 0, 0, 0, 0, 4,12, 4, 0, 0, 0, 0, 0, 0, 0
        DB  0, 0, 0, 0, 0, 0, 0, 4,12, 4, 0, 0, 0, 0, 0, 0
        DB  0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0
        DB  0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0
        DB  0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0

    ; Маска дерева
    TreeML equ 16   ; ширина маски дерева
    TreeMH equ 32   ; высота маски дерева
    Tree DB  0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0, 0,10,10,10, 0, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0,10,10,10,10,10, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0, 0,10,10,10, 0, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0, 0,10,10,10,10,10,10,10, 0, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0, 0, 6, 6, 6, 0, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0, 0, 6, 6, 6, 0, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0,10,10,10,10,10, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0,10,10,10,10,10,10,10,10,10, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0, 0, 6, 6, 6, 0, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0, 0, 6, 6, 6, 0, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0,10,10,10,10,10,10,10,10,10, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0, 0, 6, 6, 6, 0, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0, 0, 6, 6, 6, 0, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0, 0, 6, 6, 6, 0, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0, 0, 6, 6, 6, 0, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0,10,10,10,10,10, 0, 0, 0, 0, 0, 0
         DB  0, 0, 0,10,10,10,10,10,10,10,10,10, 0, 0, 0, 0
         DB  0, 0, 0, 0, 0, 0, 6, 6, 6, 0, 0, 0
```

Листинг 2.17 (продолжение)

```
        DB 29,23,30, 0,26,27,24, 0,28, 0,25,30, 0,24, 0, 0
        DB  0,25,29,24, 0,30,26,24,28,27,25, 0,26, 0,26,24
        DB 28,24,25,27,28, 0,25,23,26,27,23,27,24, 0,25,22
        DB 24,23, 0,25,22,27,24, 0,25,23,22,28,24,26,30,23
        DB 25, 0,24,22, 0,25,23,22, 0,25,25, 0,26,24,27, 0
        DB 25, 0,31,23,26, 0,29,24,23, 0,26,24,27,26,23, 0
        DB 22,25,26, 0,29,26,24, 0,26, 0,24,26, 0,25, 0, 0
        DB  0,25,20, 0,23,25,22,25, 0,23, 0,30, 0,24,25,27
        DB 24, 0,27,19,28,26, 0,20,24,22,21,25,23,25,22, 0
        DB 22, 0,23,24,20, 0,24,23,20,21,22,24,25,23, 0,24
        DB 21,22,19, 0,24,23,24,22,25, 0,21, 0,23,20, 0, 0
        DB  0,19,20,24, 0,20,23, 0,18,19,17, 0,21,20,19,20
        DB 18,21,22,20,23, 0,20,23,22,20,23,22,18, 0, 0,19
        DB  0, 0,18, 0,20, 0,17,21,20, 0,19,17,16,19, 0,18
        DB  0,24,18,19, 0,17,20,22,18,17, 0,20, 0,23, 0, 0
        DB  0, 0, 0, 0,20,23, 0,24,20,18,21,23, 0,20,19, 0
        DB 22,21, 0,22,18,19,24, 0,20, 0,19,22,20, 0, 0, 0
        DB  0, 0, 0,18, 0,17,20,19,20,19,21, 0,20,17,19,21
        DB 20,18,17,20,20,19,20,17,20, 0,19,22, 0, 0, 0, 0
        DB  0,22, 0, 0,19, 0, 0,16,17, 0,21,23,25, 0,19, 0
        DB  0,16, 0,17,18, 0, 0, 0,20, 0,19, 0, 0,18, 0, 0

; Массив координат облаков: первое значение в паре -
; номер строки (Y), второе - номер колонки (X)
CloudPos DD 100,200, 200,300, 200,310, 200,320, 160,500
         DD 160,515, 165,505, 300,110, 295,105, 293,115

; Массив пар приращений координат для ракеты и пламени
; из ее сопла: первый элемент пары - приращение
; координаты ракеты, второй - смещение маски
; пламени относительно начала маски ракеты
; (изменяется от 16 до 32).
RktStartStep DD 0,16, 0,17, 0,18, 0,19
             DD 0,20, 0,21, 0,22, 0,23
             DD 0,24, 0,25, 0,26, 0,27
             DD 1,28, 1,29, 2,30, 2,31
             DD 3,32, 3,32, 4,32, 4,32
             DD 5,32, 5,32, 6,32, 6,32
             DD 7,32, 7,32, 8,32, 8,32
             DD 9,32, 10,32, 11,32, 12,32
             DD 13,32, 14,32, 15,32, 16,32
             DD 17,32, 18,32, 19,32, 20,32
             DD 21,32, 22,32, 23,32, 24,32
          DD 26,32, 28,32, 30,32, 32,32
          DD 34,32, 36,32, 38,32, 40,32
          DD 42,32, 44,32, 46,32, 48,32

; Массив шагов (смещений) осколков из
; 2*SplinterMaxNumber элементов: первый элемент
; пары - шаг по Y, второй - шаг по X
SplStep DB 0,1, 1,0, 0,-1, -1,0, 1,1, 1,-1, -1,1, -1,-1
        DB 0,2, 2,0, 0,-2, -2,0, 2,2, 2,-2, -2,2, -2,-2
        DB 1,2, 1,-2, -1,2, -1,-2, 2,1, 2,-1, -2,1,-2,-1
        DB 1,3, 1,-3, -1,3, -1,-3, 3,1, 3,-1, -3,1,-3,-1
        DB 2,3, 2,-3, -2,3, -2,-3, 3,2, 3,-2, -3,2,-3,-2

; Массив цветов осколков
ExpColors DB WHITE,WHITE,YELLOW,YELLOW,LIGHTRED
          DB LIGHTRED,RED,RED,DARKGREY,DARKGREY

; МАССИВЫ СЛУЧАЙНЫХ ЧИСЕЛ
; Массив смещений самолета по высоте
DeltaS DD 182,3,52,230,162,6,142,227,122,189
       DD 72,91,128,214,209,76,45,169,31,3
       DD 125,153,139,168,27,220,66,176,174,233
       DD 205,83,137,66,184,51,134,192,81,24
       DD 71,185,173,164,69,8,90,233,16,153
       DD 25,69,170,233,79,73,110,15,147,21
       DD 225,81,13,145,161,125,166,167,80,220
       DD 105,228,129,181,162,118,127,207,30,16
       DD 144,84,9,125,146,135,80,1,90,229
       DD 189,235,97,47,193,107,162,30,234,82
; Массив начальных задержек запуска самолета
DeltaT DW 148,83,65,149,54,18,131,2,55,166
       DW 87,97,128,51,117,188,96,163,61,177
       DW 115,197,78,177,56,73,152,144,99,48
       DW 5,35,163,97,96,117,79,70,55,192
       DW 143,11,42,160,141,135,192,83,98,63
       DW 4,183,41,189,63,118,9,2,146,181
       DW 26,31,91,174,58,23,94,182,68,34
       DW 199,14,189,44,63,131,135,172,150,28
       DW 76,102,25,94,76,191,127,104,134,82
       DW 82,189,136,88,121,18,139,162,82,43
; Массив направлений полета (0 - слева направо,
; 1 - справа налево)
Direction DB 0,1,1,1,1,0,0,0,1,1,0,0,0,1,1,0,1,0,1,0
          DB 0,0,1,1,0,1,1,0,1,0,1,1,0,0,0,1,1,1,0,0
          DB 0,0,0,1,0,1,0,1,1,1,0,1,1,0,1,0,0,1,1,0
          DB 1,1,1,0,1,1,0,1,0,0,1,0,1,1,1,1,0,0,1,0
          DB 1,1,1,1,1,0,0,1,1,1,1,0,0,1,1,1,0,0,0,1
; Массив скоростей полета самолета
PlnSpeed DD 2,1,2,1,1,1,2,1,2,1,2,2,1,2,2,2,2,2,1,2
         DD 2,1,1,2,2,1,2,2,2,1,1,2,1,2,1,2,2,2,2,1
            DD 2,1,1,2,1,2,1,1,2,2,2,2,2,1,1,1,1,1,2,2
            DD 1,2,2,2,1,1,1,2,2,1,1,1,1,2,2,1,1,2,2,2
            DD 2,1,1,1,1,1,1,2,1,2,2,2,1,2,2,1,1,2,1,2
   ENDS
```

ПРИМЕЧАНИЕ

При создании масок я не использовал никаких специальных редакторов, а непосредственно набирал числовые коды цветов из стандартной палитры VGA. Первые 16 кодов стандартной палитры соответствуют кодам 16-цветных режимов, а следующие 16 кодов — оттенкам серого цвета (от черного до белого). Используя анимационные пакеты или самодельные редакторы изображений, можно создавать гораздо более красивые маски, если задействовать все 256 доступных оттенков и перепрограммировать регистры ЦАП при помощи функции 10h прерывания lnt10h (подфункция 12h).

Листинг 2.18 содержит группу подпрограмм, специфических для одностраничного режима (для режима с переключением страниц в них нужно вносить дополнения). Процедура DrawMovingImage обеспечивает вывод динамических (движущихся) изображений в видеопамять, а Deleteimage — их стирание путем восстановления фона. Для рисования статических объектов фона используется подпрограмма DrawStaticImage. Процедура ShowBackground полностью копирует фон в видеопамять в начале каждого нового эпизода. Процедура ShowEscapedPlanes служит для вывода в левый верхний угол экрана количества пропущенных самолетов.

Листинг 2.18. Набор подпрограмм для вывода фона и динамических объектов в одностраничном режиме (list4_18.inc)

```
; Номер начальной строки видеостраницы
VPage0StartString equ 0
; Начальный адрес видеостраницы
VPage0Address     equ 0

DATASEG
; ПАРАМЕТРЫ ПОДПРОГРАММЫ РИСОВАНИЯ СПРАЙТА
; Указатель на маску объекта
ImageMaskOffset DW ?
; Размеры маски изображения
ImageL DD ? ;ширина маски
ImageH DD ? ;высота маски
; Позиция маски изображения на экране
ImageS DD ? ;строка
ImageC DD ? ;колонка
ImageA DD ? ;линейный адрес
ImageF DB ? ;флаг присутствия объекта на экране
ENDS

CODESEG
;*****************************************************
;* ПЕРЕПИСАТЬ ФОН ИЗ ОПЕРАТИВНОЙ ПАМЯТИ В ВИДЕОПАМЯТЬ *
;*          (процедура параметров не имеет)          *
;*****************************************************
PROC ShowBackground near
        pushad
        cld
        mov     EDI,0
        ; Загрузить в счетчик размер области фона
        mov     ECX,LogicalStringLength*ScreenHeigth
        ; Скопировать фон
@@N:    mov     AL,[GS:EDI]
        mov     [FS:EDI],AL
        inc     EDI
        dec     ECX
        jnz     @@N
        popad
        ret
ENDP ShowBackground

;*****************************************************
;*          НАРИСОВАТЬ ДИНАМИЧЕСКИЙ ОБЪЕКТ           *
;* Для передачи параметров используются глобальные *
;* переменные ImageMaskOffset, ImageL, ImageH,      *
;* ImageS, ImageC, ImageA, ImageF.                  *
;*****************************************************
PROC DrawMovingImage near
        pushad
        cmp     [ImageF],0
        je      @@End
        cld
; Вычислить адрес начальной точки для вывода маски
        ; Умножить длину строки на номер строки (Y)
        mov     EAX,LogicalStringLength
        mov     EDX,[ImageS]
        mul     EDX
        ; Прибавить номер колонки (X)
        add     EAX,[ImageC]
        mov     [ImageA],EAX ;запомнить смещение
        mov     EDI,EAX ;результат - в индексный регистр
        ; Записать адрес маски в индексный регистр
        mov     SI,[ImageMaskOffset]
; Вывести изображение
        mov     DX,[word ptr ImageH] ;высота маски
@@M0:   ; Вывести очередную строку маски
        mov     CX,[word ptr ImageL] ;ширина маски
    @@M1:   ; Проверить точку маски
            lodsb
            and     AL,AL           ;код цвета равен нулю?
            jz      @@M2            ;пропустить точку
            mov     [FS:EDI],AL     ;вывести точку
    @@M2:   ; Перейти к следующей точке
            inc     EDI
            loop    @@M1
            ; Перейти на следующую строку
            add     EDI,LogicalStringLength
            sub     EDI,[ImageL]
            dec     DX
            jnz     @@M0
    @@End:  popad
            ret
    ENDP DrawMovingImage
    ;*****************************************************
    ;*            НАРИСОВАТЬ ЭЛЕМЕНТ ФОНА                *
    ;* Для передачи параметров используются глобальные *
    ;* переменные ImageMaskOffset, ImageL, ImageH,      *
    ;* ImageS, ImageC.                                  *
    ;*****************************************************
    PROC DrawStaticImage near
            pushad
            cld
    ; Загрузить начальное смещение
            mov     EDI,VPage0Address
    ; Вычислить адрес начальной точки для вывода маски
            ; Умножить длину строки на номер строки (Y)
            mov     EAX,LogicalStringLength
            mov     EDX,[ImageS]
            mul     EDX
            ; Прибавить номер колонки (X)
            add     EAX,[ImageC]
            ; Прибавить результат к начальному смещению
            add     EDI,EAX
            ; Записать адрес маски в индексный регистр
            mov     SI,[ImageMaskOffset]
    ; Вывести изображение
            mov     DX,[word ptr ImageH] ;высота маски
    @@M0:   ; Вывести очередную строку маски
            mov     CX,[word ptr ImageL] ;ширина маски
    @@M1:   ; Проверить точку маски
            lodsb
            and     AL,AL           ;код цвета равен нулю?
            jz      @@M2            ;пропустить точку
            mov     [GS:EDI],AL     ;вывести точку
    @@M2:   ; Перейти к следующей точке
            inc     EDI
            loop    @@M1
            ; Перейти на следующую строку
            add     EDI,LogicalStringLength
            sub     EDI,[ImageL]
            dec     DX
            jnz     @@M0
            popad
            ret
ENDP DrawStaticImage
;*****************************************************
;*      СТЕРЕТЬ ИЗОБРАЖЕНИЕ (ВОССТАНОВИТЬ ФОН)       *
;* Для передачи параметров используются глобальные *
;* переменные ImageL, ImageH, ImageA, ImageF.       *
;*****************************************************
PROC DeleteImage near
        pushad
        cmp     [ImageF],0
        je      @@End
; Записать в индексные регистры адрес изображения
        mov     EDI,[ImageA]
; Вывести исходное изображение
        mov     DX,[word ptr ImageH] ;высота маски
@@M0:   mov     CX,[word ptr ImageL] ;ширина маски
@@M1:   mov     AL,[GS:EDI]
        mov     [FS:EDI],AL
        inc     EDI
        loop    @@M1
```

Листинг 2.18 (продолжение)

```
        add     EDI,LogicalStringLength
        sub     EDI,[ImageL]
        dec     DX
        jnz     @@M0
@@End:  popad
        ret
ENDP DeleteImage

;*****************************************
;* ОТОБРАЗИТЬ ЧИСЛО ПРОПУЩЕННЫХ САМОЛЕТОВ *
;*     (в левом верхнем углу экрана)     *
;*****************************************
PROC ShowEscapedPlanes NEAR
        pushad
        mov     SI,offset Font8x16
        mov     AX,[EscPlns]
        add     AX,'0'
        shl     AX,4
        add     SI,AX
        mov     EDI,4*LogicalStringLength+8
; Определить цвет цифры
            mov     BL,LIGHTGREEN
            cmp     [EscPlns],0
            je      @@Bkgr
            mov     BL,YELLOW
            cmp     [EscPlns],1
            je      @@Bkgr
            mov     BL,LIGHTRED
    @@Bkgr: mov     DL,BLUE
    ; Вывести цифру
            mov     AH,16 ;счетчик строк маски буквы
    @@M0:   lodsb
            mov     CX,8  ;счетчик точек в строке маски
    @@M1:   rol     AL,1
            jc      @@M2
            mov     [FS:EDI],DL
            mov     [FS:EDI+LogicalStringLength],DL
            inc     EDI
            mov     [FS:EDI],DL
            mov     [FS:EDI+LogicalStringLength],DL
            jmp     @@M3
    @@M2:   mov     [FS:EDI],BL
            mov     [FS:EDI+LogicalStringLength],BL
            inc     EDI
            mov     [FS:EDI],BL
            mov     [FS:EDI+LogicalStringLength],BL
    @@M3:   inc     EDI
            loop    @@M1
            add     EDI,2*LogicalStringLength-16
            dec     AH
            jnz     @@M0
            popad
            ret
ENDP ShowEscapedPlanes
ENDS
```

В листинге 2.19 собраны универсальные подпрограммы, которые могут использоваться как в одностраничном режиме, так и в режиме переключения страниц:

- процедура InitEpisode инициализирует переменные состояния динамических объектов в начале каждого эпизода — это просто линейный участок кода, вырезанный из основного модуля и оформленный в виде подпрограммы для большей наглядности;
- процедуры GShowDecByte, GShowDecWord и GShowDecDWord служат для вывода на экран в графическом режиме байта, слова и двойного слова данных в десятичном коде;
- подпрограмма ShowGameResults после завершения игры выводит на экран ее результаты, используя процедуры вывода десятичных чисел;
- процедуры CopyPlaneMask и MirrorPlaneMask предназначены для создания копии маски самолета, но CopyPlaneMask просто дублирует маску, а MirrorPlaneMask отражает ее слева направо (в начале каждого эпизода вызывается только одна из этих процедур — в зависимости от направления движения самолета; полученная таким образом копия маски используется в дальнейшем при выводе изображения самолета в видеопамять);
- процедура DrawMainBackground создает в дополнительной памяти компьютера фон изображения: верхняя область (небо) закрашивается в синий цвет, нижняя (земля) — в черный и зеленый (в клеточку); после этого поверх земли рисуются деревья в четыре ряда и пусковая установка в центре, а на небо наносятся облака (путем многократного наложения одной и той же маски);
- подпрограмма ExpMaskClear предназначена для очистки (обнуления) маски взрыва, а процедура ExplosionFrame — для генерации очередных кадров изображения взрыва (маска взрыва генерируется динамически путем наложения друг на друга цветных точек — осколков, разлетающихся в разные стороны с разными скоростями).

Листинг 2.19. Универсальные подпрограммы, пригодные для одностраничного и многостраничного режимов (list4_19.inc)

```
CODESEG
;*************************************************
;*     ВЫВОД БАЙТА НА ЭКРАН В ДЕСЯТИЧНОМ КОДЕ    *
;* Подпрограмма выводит содержимое регистра AL   *
;* в десятичном коде в указанную позицию экрана. *
;* Координаты позиции передаются через глобальные *
;* переменные ScreenString и ScreenColumn.       *
;*************************************************
PROC GShowDecByte NEAR
        push    EAX
        and     EAX,0FFh
        call    GShowDecDWord
        pop     EAX
```

Листинг 2.19 (продолжение)

```
        ret
ENDP GShowDecByte

;*****************************************************
;*        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 переменной TextColorAndBackground.     *
;*****************************************************
PROC GShowDecWord NEAR
        push    EAX
        and     EAX,0FFFFh
        call    GShowDecDWord
        pop     EAX
        ret
ENDP GShowDecWord

;*****************************************************
;*        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 переменной TextColorAndBackground.     *
;*****************************************************
PROC GShowDecDWord NEAR
        pushad
        push    DS
; Настроить регистр DS на глобальный сегмент данных
        mov     SI,[CS:MainDataSeg]
        mov     DS,SI
; Перевести число в десятичный код
        mov     [Data_Int32],EAX
        call    Int32_to_String
        mov     DH,[byte ptr ScreenString]
        mov     DL,[byte ptr ScreenColumn]
; 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
        call    PutGraChar
        loop    @@NextChar
@@EndOfString:
        pop     DS
        popad
        ret
ENDP GShowDecDWord

;*****************************************************
;*  УСТАНОВИТЬ НАЧАЛЬНОЕ СОСТОЯНИЕ ПЕРЕМЕННЫХ ЭПИЗОДА *
;* (вспомогательная процедура, регистры не сохраняет) *
;*****************************************************
PROC InitEpisode near
; Перерисовать фон
        call    ShowBackground
; Сбросить начальные адреса изображений объектов
        mov     [PlnA],0
        mov     [RktA],0
        mov     [FlmA],0
        mov     [ExpA],0
; Сбросить признаки наличия объектов
        mov     [PlnF1],0
        mov     [RktF1],0
        mov     [FlmF1],0
        mov     [ExpF1],0
; Сбросить счетчик времени
        mov     [GameTimeCounter],0
; Сбросить признак попадания ракеты в самолет
        mov     [HitFlag],0
; Установить начальное состояние самолета
        mov     [PlnState],0
        ; Прибавить случайное смещение по высоте
        mov     BX,[EpisodeNumber]
        shl     BX,2
        add     BX,offset DeltaS
        mov     EAX,[BX]
        add     EAX,80
        mov     [PlnS],EAX
        ; Запомнить направление движения самолета
        mov     BX,[EpisodeNumber]
        add     BX,offset Direction
        mov     AL,[BX]
        mov     [PlaneDirection],AL
        ; Вычислить скорость самолета
        mov     BX,[EpisodeNumber]
        shl     BX,2
        add     BX,offset PlnSpeed
        mov     EAX,[BX]
        mov     [PlaneSpeed],EAX
        ; Задержка пуска самолета
        mov     BX,[EpisodeNumber]
        shl     BX,2
        add     BX,offset DeltaT
        mov     AX,[BX]
        mov     [PlaneDeltaT],AX
        ; Скопировать маску самолета
        call    CopyPlaneMask
        ; Сместить самолет за левую границу экрана
        mov     [PlnC],-PlnML
        ; Направление движения?
        cmp     [PlaneDirection],0
        je      @@dir0
        ; Сместить самолет за правую границу экрана
        mov     [PlnC],ScreenLength
        ; "Отрицательная" скорость
        neg     [PlaneSpeed]
        ; Использовать отраженную маску
        call    MirrorPlaneMask
; Установить начальное состояние ракеты
@@dir0: mov     [RktState],0 ;состояние 0
        ; Отобразить ракету на стартовой позиции
        mov     [RktF1],1  ;отобразить ракету
        mov     EAX,[PRS]  ;координаты стартовой позиции
        mov     [RktS],EAX
        mov     EAX,[PRC]
        mov     [RktC],EAX
        ret
ENDP InitEpisode
;*************************
;* ВЫВСТИ РЕЗУЛЬТАТЫ ИГРЫ *
;*************************
PROC ShowGameResults near
        pushad
; Очистить экрана
        call    GClearScreen
; Отобразить результаты
        mov     SI,offset EndTxt
        mov     [DefaultColor],LIGHTCYAN
        call    GShowString
```

Листинг 2.19 (продолжение)

```
        mov     [DefaultColor],LIGHTGREEN
        call    GShowString
        call    GShowString
        call    GShowString
        call    GShowString
        mov     [DefaultColor],YELLOW
        call    GShowString
        mov     [DefaultColor],WHITE
        mov     [ScreenString],12
        mov     [ScreenColumn],18
        mov     AX,[EpisodeNumber]
        call    GShowDecWord
        mov     [ScreenString],14
        mov     [ScreenColumn],16
        mov     AX,[RktCounter]
        call    GShowDecWord
        mov     [ScreenString],16
        mov     [ScreenColumn],17
        mov     AX,[DestroedPlns]
        call    GShowDecWord
        mov     [ScreenString],18
        mov     [ScreenColumn],21
        mov     AX,[EscPlns]
        call    GShowDecWord
; Ожидать нажатия любой клавиши
        call    GetChar
        popad
        ret
ENDP ShowGameResults

;************************
;* СОЗДАТЬ ОСНОВНОЙ ФОН *
;************************
PROC DrawMainBackground near
        pushad
; Загрузить начальное смещение (адрес видеостраницы)
        mov     EDI,VPage0Address
; Умножить высоту "неба" (высота экрана минус 64 строки)
; на логическую ширину строки (1024 пиксела)
        mov     ECX,ScreenHeigth-64
        shl     ECX,10
; Покрасить небо в синий цвет
        mov     AL,BLUE
@@NextPixel1:
        mov     [GS:EDI],AL
        inc     EDI
        dec     ECX
        jnz     @@NextPixel1
; Покрасить землю в черный и зеленый цвета
        mov     AL,BLACK
        mov     AH,GREEN
; Высота "земли" 64 строки
        mov     DX,64
@@NextStr:
        mov     CX,LogicalStringLength/2
@@NextPixel2:
        mov     [GS:EDI],AX
        add     EDI,2
        loop    @@NextPixel2
        xchg    AL,AH
        dec     DX
        jnz     @@NextStr

; НАРИСОВАТЬ ЛЕС
        ; Начальная строка "леса"
        mov     [dword ptr TreeS],ScreenHeigth-80
            ; Число рядов деревьев
            mov     DX,4
    ; Цикл рисования леса
    @@Trees:; Нарисовать очередной ряд деревьев
            mov     [dword ptr TreeC],0
            test    DX,01b
            jz      @@EvenLine
            ; Сместить нечетные ряды
            add     [dword ptr TreeC],16
    @@EvenLine:
            mov     CX,ScreenLength/(TreeML*2)
    @@NextTree:
            cmp     CX,ScreenLength/(TreeML*4)
            je      @@NoTree ;"просека" в лесу
            cmp     CX,ScreenLength/(TreeML*4)+1
            je      @@NoTree ;"просека" в лесу
            ; Нарисовать очередное дерево
    DrawSImage TreeS,TreeC,TreeML,TreeMH,Tree
    @@NoTree:
            add     [dword ptr TreeC],TreeML*2
            loop    @@NextTree
            add     [dword ptr TreeS],TreeMH/2
            dec     DX
            jnz     @@Trees

    ; НАРИСОВАТЬ ОБЛАКА
            mov     BX,offset CloudPos
            mov     CX,MaxCloudNum ;счетчик облаков
    @@NextCloud:
            ; Нарисовать очередное облако
            mov     EAX,[BX]
            mov     [CloudS],EAX
            add     BX,4
            mov     EAX,[BX]
            mov     [CloudC],EAX
            add     BX,4
    DrawSImage CloudS,CloudC,CloML,CloMH,Cloud
            loop    @@NextCloud

    ; НАРИСОВАТЬ ПУСКОВУЮ УСТАНОВКУ
            mov     [PRS],ScreenHeigth-PRMH-16
            mov     [PRC],(ScreenLength-PRML)/2
    DrawSImage PRS,PRC,PRML,PRMH,Platf
            popad
            ret
    ENDP DrawMainBackground

    ;******************************
    ;* СКОПИРОВАТЬ МАСКУ САМОЛЕТА *
    ;******************************
PROC CopyPlaneMask near
        pusha
        push    ES
        mov     AX,[CS:MainDataSeg]
        mov     ES,AX
        mov     SI,offset Plane
        mov     DI,offset PlnMask
        ; Скопировать весь массив
        mov     CX,PlnMH*PlnML
        rep     movsb
        pop     ES
        popa
        ret
ENDP CopyPlaneMask

;************************************
;* ПОЛУЧИТЬ ОТРАЖЕНИЕ МАСКИ САМОЛЕТА *
;************************************
PROC MirrorPlaneMask near
        pusha
        mov     SI,offset Plane
        mov     DI,offset PlnMask
        mov     DX,PlnMH
; Цикл по строкам
@@L0:   mov     CX,PlnML
        add     DI,PlnML-1
; Цикл по пикселам
@@L1:   lodsb
        mov     [DI],AL
        dec     DI
```

Листинг 2.19 (продолжение)

```
        loop    @@L1
        add     DI,PlnML+1
        dec     DX
        jnz     @@L0
        popa
        ret
ENDP MirrorPlaneMask

;************************
;* СТЕРЕТЬ МАСКУ ВЗРЫВА *
;************************
PROC ExpMaskClear near
        pusha
        push    ES
        mov     AX,[CS:MainDataSeg]
        mov     ES,AX
        mov     DI,offset ExpMask
        mov     CX,4*ExpR*ExpR
        mov     AL,0
        rep     stosb
            pop     ES
            popa
            ret
    ENDP ExpMaskClear
    ;*****************************************
    ;* НАРИСОВАТЬ МАСКУ ОЧЕРЕДНОЙ ФАЗЫ ВЗРЫВА *
    ;*****************************************
    PROC ExplosionFrame near
            pusha
            cld
    ; Вывести изображение взрыва
            ; Записать адрес массива смещений осколков
            mov     SI,offset SplStep
            ; Записать число осколков
            mov     CX,SplinterMaxNumber
    @@NextSplinter:
    ; Загрузить в DI указатель на маску взрыва
            mov     DI,offset ExpMask
    ; Умножить шаг по Y на номер кадра (плюс 1)
            lodsb           ;записать шаг по Y в AL
            mov     AH,[byte ptr ExpFrameNumber]
            inc     AH      ;увеличить номер кадра на 1
            imul    AH      ;умножить шаг на номер кадра
            ; Прибавить номер строки центра маски
            add     AX,ExpR-1
            ; Умножить разультат на длину строки маски
            mov     DX,2*ExpR
            mul     DX
            ; Прибавить результат к смещению в маске
            add     DI,AX
    ; Умножить шаг по X на номер кадра (плюс 1)
            lodsb           ;записать шаг по X в AL
            mov     AH,[byte ptr ExpFrameNumber]
            inc     AH      ;увеличить номер кадра на 1
            imul    AH      ;умножить шаг на номер кадра
            ; Прибавить номер колонки центра маски
            add     AX,ExpR-1
            ; Прибавить результат к смещению в маске
            add     DI,AX
    ; Нанести точку-осколок на маску
            mov     BX,offset ExpColors
            add     BX,[ExpFrameNumber]
            mov     AL,[BX]
            mov     [DI],AL
            loop    @@NextSplinter
            popa
            ret
    ENDP ExplosionFrame
    ENDS
```

Листинг 2.20 содержит головной модуль одностраничного варианта игры «Самолет и ракета» PlaneAndRocket. Перед запуском игры на экран выдается текст с описанием задания (сбить все пролетающие самолеты, при трех пропущенных игра заканчивается) и управляющих клавиш (Пробел — запуск ракеты, Esc — срочный выход из игры). Самолет движется по экрану с постоянной скоростью (1 или 2 пиксела на кадр); скорость, высота и направление полета выбираются случайным образом (на каждый параметр — своя таблица случайных чисел). Ракета имеет нелинейный начальный участок разгона, а затем движется с постоянной скоростью 3 пиксела на кадр. В каждом эпизоде игры имеется только одна ракета — поспешность или задержка с запуском приводят к проигрышу в данном эпизоде. В случае промаха ракета самоликвидируется в верхних слоях атмосферы (чтобы она не попадала в зону искажения изображений).

ПРИМЕЧАНИЕ

Под мертвую зону (область искажений) в верхней части экрана выделена полоса высотой в 32 пиксела (1/15 высоты экрана), куда не должен попасть ни один динамический объект. На медленных видеоконтроллерах такой ширины мертвой зоны недостаточно, и можно наблюдать разнообразные эффекты.

Листинг 2.20. Игра «Самолет и ракета» — основной модуль программы для одностраничного режима

```
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"
; Подключить описание констант, глобальных переменных
; и макрокоманд
include "list4_16.inc"
; Подключить файл масок объектов
include "list4_17.inc"
DATASEG
; ТЕКСТОВЫЕ СООБЩЕНИЯ
Txt1 DB 0,24,"АНИМАЦИЯ В ОДНОСТРАНИЧНОМ РЕЖИМЕ",0
     DB 2,2,"Параметры видеорежима: разрешение "
     DB "640x480, 256 цветов, линейная адресация",0
Txt2 DB 10,28,"ИГРА ",'"',"САМОЛЕТ И РАКЕТА",'"',0
Txt3 DB 13,19
     DB "Задание: сбить все пролетающие самолеты.",0
     DB 15,22,"Игра завершается после 100 эпизодов",0
     DB 16,24,"или трех пропущенных самолетов.",0
Txt4 DB 20,0,"Управляющие клавиши:",0
Txt5 DB 22,0,"Пробел - запустить ракету;",0
     DB 24,0,"Esc    - срочный выход из программы.",0
Txt6 DB 29,29,"Нажмите любую клавишу",0
ENDS
```

Листинг 2.20 (продолжение)

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

CODESEG
;*****************************************
;*            ОСНОВНАЯ ПРОГРАММА          *
;*****************************************
PROC PlaneAndRocket
        mov     AX,DGROUP
        mov     DS,AX
        mov     [CS:MainDataSeg],AX
; Установить текстовый режим
        mov     AX,3
        int     10h
; "Захватить" текстовый шрифт
        call    GrabRusFont
; Установить видеорежим
        call    SetVESAVideoMode
; Установить режим прямой адресации памяти
        call    GInitialization

; ВВОДНЫЙ ТЕКСТ
; Отобразить текстовые сообщения
        mov     [DefaultBackground],BLACK ;черный фон
        mov     [DefaultColor],LIGHTGREEN ;зеленый текст
        MGShowText 2,Txt1
        mov     [DefaultColor],LIGHTCYAN  ;голубой текст
        MGShowString Txt2
        mov     [DefaultColor],LIGHTGREEN ;зеленый текст
        MGShowText 3,Txt3
        mov     [DefaultColor],YELLOW     ;желтый текст
        MGShowString Txt4
        mov     [DefaultColor],LIGHTGREEN ;зеленый текст
        MGShowText 2,Txt5
        mov     [DefaultColor],YELLOW     ;желтый текст
        MGShowString Txt6
        ; Ожидать нажатия любой клавиши
        call    GetChar
; ПОДГОТОВКА К ОСНОВНОМУ ЦИКЛУ
        call    DrawMainBackground ;рисование фона
; Сбросить счетчики
        mov     [PlnCounter],0
        mov     [DestroedPlns],0
        mov     [EscPlns],0
        mov     [RktCounter],0
        mov     [EpisodeNumber],0

;####################
;# ЦИКЛ ПО ЭПИЗОДАМ #
;####################
@@NextEpisode:
        call    InitEpisode

;#####################
;# ЦИКЛ ВЫВОДА КАДРА #
;#####################
@@FrameCycle:

; СТЕРЕТЬ ПОДВИЖНЫЕ ОБЪЕКТЫ
; Стереть самолет
DeleteMImage PlnA,PlnML,PlnMH,PlnF1
; Стереть ракету и пламя из ее сопла
DeleteMImage FlmA,FlmML,FlmMH,FlmF1
DeleteMImage RktA,RktML,RktMH,RktF1
; Стереть взрыв
DeleteMImage ExpA,ExpR*2,ExpR*2,ExpF1

; ОПРОС КЛАВИАТУРЫ
        call    WaitChar
        cmp     AX,0      ;клавиша была нажата?
        je      @@CommandNotInput ;нажатий не было
        cmp     AX,20h    ;нажат пробел?
        je      @@Fire    ;запустить ракету
        cmp     AX,1B00h  ;нажата клавиша ESC?
        je      @@Exit    ;выход
        jmp     @@CommandNotInput ;команды не было
; Пуск ракеты
@@Fire: cmp     [RktState],0
        jne     @@RocketNotReady ;ракета не готова
        mov     [RktStartFrameNumber],0
        mov     [RktState],1     ;запуск
        mov     [FlmF1],1        ;отобразить выхлоп
        inc     [RktCounter]
        mov     [GameStateGhange],1
@@RocketNotReady:
@@CommandNotInput:

; ВЫВОД ИЗОБРАЖЕНИЯ САМОЛЕТА
        cmp     [PlnState],0 ;ожидание
            je      @@Pl0
            cmp     [PlnState],1 ;полет
            je      @@Pl1
            cmp     [PlnState],2 ;взрыв
            je      @@Pl2
    ; Ожидание самолета
    @@Pl0:  ; Проверить истечение времени задержки самолета
            mov     AX,[PlaneDeltaT]
            cmp     [GameTimeCounter],AX
            jb      @@NoPl
            inc     [PlnCounter] ;увел. счетчик самолетов
            mov     [PlnState],1 ;запустить самолет
    ; Полет самолета
    @@Pl1:  mov     [PlnF1],1    ;отобразить самолет
            mov     EAX,[PlaneSpeed]
            add     [PlnC],EAX
            jmp     @@NoPl
    ; Взрыв самолета
    @@Pl2:  cmp     [ExpFrameNumber],MaxExplFrameNumber
            jae     @@EndOfEpisode
            ; Нарисовать маску очередной
            ; фазы взрыва самолета
            call    ExplosionFrame
            inc     [ExpFrameNumber]
    @@NoPl: ; (завершена обработка состояния самолета)

    ; ВЫВОД ИЗОБРАЖЕНИЯ РАКЕТЫ
            cmp     [RktState],1 ;запуск ракеты
            je      @@ro1
            cmp     [RktState],2 ;ракета в полете
            je      @@ro2
            cmp     [RktState],3 ;взрыв ракеты
            je      @@ro3
            cmp     [RktState],4 ;разлет осколков
            je      @@ro4
            jmp     @@EndRktDraw
    ; Разгон ракеты
    @@ro1:  mov     BX,[RktStartFrameNumber]
            cmp     BX,MaxRktStartFrame
            jb      @@NextRSFrame
            mov     [RktState],2
            jmp     @@ro2
    @@NextRSFrame:
            shl     BX,3       ;умножить номер кадра на 8
            add     BX,offset RktStartStep
            mov     EAX,[PRC]
            mov     [RktC],EAX
            mov     EAX,[PRS]  ;вычесть смещение ракеты
            sub     EAX,[BX]   ;из номера строки установки
        mov     [RktS],EAX
        add     EAX,[BX+4] ;прибавить смещение пламени
        mov     [FlmS],EAX ;к номеру строки ракеты
        inc     [RktStartFrameNumber]
        jmp     @@EndRktDraw
; Полет ракеты
```

Листинг 2.20 (продолжение)

```
@@ro2:  ; Проверка на выход ракеты за верхнюю
        ; границу "атмосферы"
        cmp     [RktS],48   ;граница зоны искажений?
        jl      @@ro3       ;самоуничтожение ракеты
        ; Вычислить квадрат расстояния
        ; между самолетом и ракетой
        mov     EAX,[RktS]
        sub     EAX,[PlnS]
        imul    EAX
        ; (в EAX - квадрат расстояния по Y)
        mov     EBX,EAX
        mov     EAX,[RktC]
        sub     EAX,[PlnC]
        sub     EAX,(PlnML-RktML)/2
        imul    EAX         ;квадрат расстояния по X
        ; (в EAX - квадрат расстояния по X)
        add     EAX,EBX
        ; (в EAX - квадрат расстояния между самолетом
        ; и ракетой)
        cmp     EAX,TargetDistanceSQ
        jae     @@DrawRocket
        ; Произошло попадание ракеты в самолет
        mov     [HitFlag],1 ;установить флаг попадания
        jmp     @@ro3       ;начать отображение взрыва
@@DrawRocket:
        sub     [RktS],3
        mov     EAX,[RktS]
        add     EAX,32
        mov     [FlmS],EAX
        jmp     @@EndRktDraw

; Начало взрыва ракеты
@@ro3:  ; Переключить состояние ракеты
        mov     [RktState],4
        ; Очистить маску взрыва
        call    ExpMaskClear
        ; Определить координаты взрыва
        mov     [ExpFrameNumber],0
        mov     EAX,[RktS]
        sub     EAX,ExpR-RktMH/2
        mov     [ExpS],EAX
        mov     EAX,[RktC]
        sub     EAX,ExpR-RktML/2
        mov     [ExpC],EAX
            mov     [RktF1],0    ;убрать ракету
            mov     [FlmF1],0    ;убрать выхлоп
            mov     [ExpF1],1    ;отобразить взрыв

    ; Разлет осколков ракеты
    @@ro4:  cmp     [ExpFrameNumber],MaxExplFrameNumber
            jae     @@SetRktState5
            ; Нарисовать маску очередной фазы взрыва ракеты
            call    ExplosionFrame
            inc     [ExpFrameNumber]
            jmp     @@EndRktDraw
    @@SetRktState5:
            mov     [RktState],5 ;перезарядка установки
            mov     [ExpF1],0    ;убрать взрыв
            ; Было попадание в самолет?
            cmp     [HitFlag],0
            je      @@EndRktDraw ;попадания не было
            mov     [PlnState],2 ;взрыв самолета
            ; Увеличить значение счетчика сбитых самолетов
            inc     [DestroedPlns]
            mov     [GameStateGhange],1
            ; Очистить маску взрыва
            call    ExpMaskClear
            ; Определить координаты взрыва
            mov     [ExpFrameNumber],0
            mov     EAX,[PlnS]
            sub     EAX,ExpR-PlnMH/2
            mov     [ExpS],EAX
            mov     EAX,[PlnC]
            sub     EAX,ExpR-PlnML/2
            mov     [ExpC],EAX
            mov     [ExpF1],1    ;отобразить взрыв самолета
    @@EndRktDraw:

    ; ОТОБРАЗИТЬ ДИНАМИЧЕСКИЕ ОБЪЕКТЫ
    ; Нарисовать самолет
    DrawMImage PlnS,PlnC,PlnML,PlnMH,PlnMask,PlnA,PlnF1
    ; Нарисовать пламя ракеты
    DrawMImage FlmS,RktC,FlmML,FlmMH,Flm,FlmA,FlmF1
    ; Нарисовать ракету
    DrawMImage RktS,RktC,RktML,RktMH,Rkt,RktA,RktF1
    ; Отобразить взрыв
    DrawMImage ExpS,ExpC,ExpR*2,ExpR*2,ExpMask,ExpA,ExpF1
    ; Счетчик пропущенных самолетов
            call    ShowEscapedPlanes

    ; ОЖИДАНИЕ НАЧАЛА СЛЕДУЮЩЕГО КАДРА
            call    WaitVSync
            ; Увеличить на 1 счетчик времени
        inc     [GameTimeCounter]

; КОНЕЦ ЦИКЛА ПО КАДРАМ
        ; Самолет покинул пределы экрана?
        cmp     [PlaneDirection],0
        je      @@LtoR      ;движение слева направо
        cmp     [PlnC],-PlnML
        jl      @@PlaneEscaped
        jmp     @@FrameCycle
@@LtoR: cmp     [PlnC],ScreenLength
        jl      @@FrameCycle
@@PlaneEscaped:
; Увеличить счетчик пропущенных самолетов
        inc     [EscPlns]
        mov     [GameStateGhange],1
; КОНЕЦ ЦИКЛА ПО ЭПИЗОДАМ
@@EndOfEpisode:
; Увеличить счетчик игровых эпизодов
        inc     [EpisodeNumber]
; Закончить игру после MaxEpNumber эпизодов
        cmp     [EpisodeNumber],MaxEpNumber
        jae     @@Exit
; Конец игры, если пропущено MaxSavedPlanes самолетов
        cmp     [EscPlns],MaxSavedPlanes
        jl      @@NextEpisode
@@Exit:
; Показать результаты
        call    ShowGameResults
; Установить текстовый режим
        mov     AX,3
        int     10h
; Выход в DOS
        mov     AH,4Ch
        int     21h
ENDP PlaneAndRocket
ENDS
; Подключить процедуры вывода данных на экран
; для текстовых режимов
include "list1_02.inc"
; Подключить процедуры перевода чисел
include "list2_05.inc"
; Подключить набор процедур общего назначения,
; предназначенных для установки графических
; видеорежимов и работы в них
include "list4_02.inc"
; Подключить подпрограмму, настраивающую FS на
; видеопамять, GS - на дополнительную память
include "list4_14.inc"
    ; Подключить набор процедур вывода текста,
    ; предназначенный для 256-цветных режимов с
    ; раздельным доступом с сегментам видеопамяти
    ; и дополнительной памяти
```

Листинг 2.20 (продолжение)

```
include "list4_15.inc"
; Подключить набор процедур для вывода
; изображений объектов в одностраничном режиме
include "list4_18.inc"
; Подключить набор универсальных подпрограмм, пригодных
; и для одностраничного, и для двухстраничного режима
include "list4_19.inc"

END
```

ПРИМЕЧАНИЕ

Листинги программ, реализующих игру "Самолет и ракета" в режиме переключения страниц, приведены в [5].

2.2.6.3. Простые форматы графических файлов

Задача вывода изображений на экран тесно связана с двумя другими задачами — сохранения изображений на диске в виде файлов и считывания изображений с диска. Существует множество различных форматов графических файлов [14], однако эти форматы отличаются обычно значительной сложностью. Далее мы рассмотрим два самых простых формата файлов — формат BMP (для несжатого RGB-изображения) и формат PCX для 256-цветных режимов.

2.2.6.3.1. Формат BMP для несжатого RGB-изображения

Формат файла BMP для несжатого RGB-изображения показан в табл. 2.21. Это самый простой *из всех форматов, он воспринима*ется большинством графических редакторов, но порождает файлы гигантских размеров, поскольку *одна точка изображения кодирует*ся тремя байтами данных, по байту на каждый *из цветовых компо*нентов (аналогично формату TrueColor24).

Таблица 2.21. Формат файла BMP для несжатого RGB-изображения

Смещение	Размер	Значение	Описание
00h	WORD	4D42h (BM)	Признак файла BMP
02h	DWORD	3×L×H + 54	Полный размер файла в байтах
06h	WORD	0	Не используется
08h	WORD	0	Не используется
0Ah	DWORD	54 (36h)	Смещение области данных изображения от начала файла
0Eh	DWORD	40 (28h)	Размер описателя изображения
12h	DWORD	L	Ширина изображения в пикселах
16h	DWORD	H	Высота изображения в пикселах
1Ah	WORD	1	Число битовых плоскостей
1Ch	WORD	24 (18h)	Число битов на пиксел
1Eh	DWORD	0	Метод сжатия
22h	DWORD	3×L×H	Размер изображения в байтах
26h	DWORD	0	Разрешение по горизонтали в пикселах на метр
2Ah	DWORD	0	Разрешение по вертикали в пикселах на метр
2Eh	DWORD	0	Число цветов в растровом изображении
32h	DWORD	0	Число важных цветов изображения
36h	3×L×H байт	-	Область данных изображения

Создание файла BMP начинается с заполнения заголовка. Как видно из таблицы, в формате RGB большая часть полей заголовка заполняется строго определенными значениями (константами), никак не зависящими от вида и размеров изображения. Переменными параметрами являются только ширина изображения L, высота изображения H, размер изображения в байтах (равен умноженному на три произведению ширины и высоты) и полный размер файла в байтах (является суммой размера изображения и размера заголовка).

Допустим, нужно сохранить изображение размером 1024×768 точек. Тогда ширина L=1024, высота H=768, размер изображения равен 2 359 296 байт, а размер файла — 2 359 350 байтов.

Другой неприятной особенностью формата BMP является перевернутый (по отношению к экрану монитора) порядок записи строк в файл — первой записывается самая нижняя строка изображения. Проявляет себя этот недостаток только в том случае, если изображение по высоте превышает размеры экрана, а его просмотр нужно начинать с левого верхнего угла — чтобы добраться до верхней строки, приходится прокручивать файл до самого конца. Использование метода Родена (см. главу 2 «Недокументированные возможности процессоров Intel 80x86») обеспечивает под управлением DOS доступ ко всей оперативной памяти, что позволяет нейтрализовать недостатки формата BMP (большой объем файлов и перевернутый порядок строк).

2.2.6.3.2. Формат PCX для 256-цветных изображений

Если необходимо экономить дисковое пространство, то для 256-цветных изображений лучше использовать формат PCX *со сжати*ем по алгоритму Run Length Encoding (RLE). Файл PCX состоит из трех частей: заголовка, сжатого изображения и таблицы палитры. Формат заголовка файла PCX показан в табл. 2.22.

Вообще говоря, данные в файлах PCX могут храниться в несжатом виде — в этом случае байт признака уплотнения данных в заголовке файла имеет значение 0. Неупакованный PCX *формат может ис*пользоваться, например, для хранения черно-белых фотографий в формате 256 оттенков серого.

Однако в целях сокращения объема файла обычно применяется сжатие по RLE. Способ кодирования RLE предполагает выделение последовательностей пикселов одинакового цвета и запоминание количества повторений в специальном байте-счетчике. Каждая строка изображения кодируется отдельно, причем правила кодирования следующие:

- отдельная точка со значением кода цвета менее C0h будет просто представлена своим кодом;
- отдельная точка со значением кода цвета, которое больше или равно C0h, записывается в виде пары кодов, первым из которых будет байт счетчика со значением C1h, а вторым — собственно байт кода цвета точки;
- горизонтальный отрезок из N (не более 63) одноцветных точек будет представлен парой кодов, первым из которых будет байт счетчика со значением N + C0h, а вторым — байт кода цвета.

Если в байте версии файла PCX установлено значение 5, то последние 769 байт файла хранят структуру, содержащую байт — признак формата регистров ЦАП и следующую за ним таблицу палитры (768 байт). Если в байте признака формата регистров записано значение 0Ah, то используется 6-разрядное кодирование основных цветовых компонентов, а если значение 0Ch — 8-разрядное кодирование. Таблицу нужно загрузить в регистры ЦАП видеоконтроллера при помощи подфункции 12h функции 10h прерывания Int 10h в соответствии со значением байта признака формата. Каждая из 256 строк таблицы состоит из трех байтов, содержащих значения компонентов соответствующего цветового оттенка в порядке, используемом функцией загрузки палитры: красный, зеленый, синий.

Таблица 2.22. Формат заголовка файла PCX

Смещение	Размер	Описание
00h	BYTE	Сигнатура 0Ah — признак файла PCX
01h	BYTE	Версия файла PCX: 0 — версия 2.5; 2 — версия 2.8 с описанием палитры; 3 — версия 2.8 без описания палитры; 5 — версия 3.0
02h	BYTE	Признак уплотнения данных: 0 — сжатие не производилось; 1 — сжатие выполнено методом RLE)
03h	BYTE	Количество цветовых слоев
04h	WORD	Координата Xmin левого верхнего угла изображения
06h	WORD	Координата Ymin левого верхнего угла изображения
08h	WORD	Координата Xmax правого нижнего угла изображения
0Ah	WORD	Координата Ymax правого нижнего угла изображения
0Ch	48 байт	Описание палитры для 16-цветного режима (16 записей, по 3 байта на каждый цвет)
40h	BYTE	Зарезервировано
41h	BYTE	Число битовых плоскостей
42h	WORD	Число байтов в одной строке изображения (всегда четное)
44h	WORD	Признак типа изображения: 1 — цветное или черно-белое; 2 — с градациями серого
46h	58 байт	Зарезервировано

Палитра PCX в 256-цветном режиме часто не совпадает с используемой по умолчанию палитрой VGA, что создает дополнительные сложности при просмотре изображений — если кроме картинки вы хотите вывести какой-то поясняющий текст, то он поменяет цвет при смене палитры. Поэтому, несмотря на то, что в файле кодирование 256-цветное, при его создании и последующем просмотре удобнее всего использовать графические режимы TrueColor.

При обмене информацией между программами при помощи PCX-файлов возможны ситуации трех основных видов:

- требуется передавать изображение только между своими программами, входящими в один пакет;

- требуется принять изображение, созданное чужой программой;
- нужно передать (транспортировать) изображение из своей программы в чужую.

При обмене файлами изображений внутри одного программного пакета никаких особых сложностей нет: можно использовать любую цветовую палитру. Вообще говоря, для транспортировки данных вы даже можете создать на основе принципа RLE свой собственный метод сжатия информации, нестандартный, но оптимальный в вашем конкретном случае.

При приеме файлов из чужих программ приходится либо устанавливать ту палитру, которая приходит вместе с файлом (что приводит к искажению цветов в вашей программе), либо производить перекодировку цветов изображения из чужой палитры в свою (что приводит к искажению изображения), либо работать в режиме TrueColor (что связано с проблемами совместимости видеоконтроллеров).

При транспортировке файлов из своего пакета программ в чужой следует учитывать, что многие редакторы изображений корректно работают только с восьмибитовым форматом цветовых компонентов, то есть значение байта признака формата регистров ЦАП должно быть равно 0Ch, а данные в таблице палитры должны соответствовать этому формату. Например, если нужно передать в MS Photo Editor файл, созданный в стандартном 256-цветном VGA-режиме DOS, то после считывания палитры из регистров ЦАП нужно перед записью таблицы в файл сдвинуть в ней каждый байт данных влево на два разряда.

Если требуется реализовать режим фотографии с 256 градациями серого цвета, то соответствующую таблицу палитры нужно вначале сформировать. Генерация таблицы выполняется по примитивному алгоритму: в первую строку записываются три байта со значением ноль, а в каждой следующей строке значения для всех компонент увеличиваются на 1 (получаем в результате в первой строке черный цвет, а в последней — белый).

Ситуация, когда приходится иметь дело с графическим редактором, работающим только с конкретной палитрой, возникает довольно редко. Однако если такая необходимость есть, то приходится заимствовать таблицу палитры вместе с байтом формата ЦАП. Для этого нужно создать файл (для любого произвольного изображения) в чужом пакете, «взять» из него последние 789 байт и либо записать в виде загружаемого двоичного файла, либо перекодировать в текстовый include-файл. Все изображения, которые вы предназначаете для работы с таким редактором, должны создаваться с использованием заимствованной палитры.

Контрольные вопросы

1. В каком режиме начинает работать видеоадаптер после включения питания?
2. Какие основные операции выполняются в текстовом режиме?
3. Как организована видеопамять в текстовом режиме?
4. Как организуется вывод текста на монитор в режиме 03h?
5. Как кодируется цвет в текстовом режиме?
6. Как производится перемещение курсора по экрану?
7. Как производится переключение видеостраниц?
8. Как могут использоваться дополнительные видеостраницы в текстовом режиме?
9. Как организована память в 256-цветном графическом режиме?
10. Что подразумевают под режимом DirectDraw?
11. Как кодируется пиксел в HiColor?
12. В чем разница между HiColor15 и HiColor16?
13. В чем разница между TrueColor24 и TrueColor32?
14. Как происходит отображение видеопамяти на экран монитора в режиме HrueColor32 с разрешением 1024×768?
15. Охарактеризуйте сегментный режим адресации видеопамяти.
16. Охарактеризуйте линейный режим адресации видеопамяти.
17. Каков основной недостатк метода линейной адресации?
18. Как задаются видеорежим и способ адресации видеопамяти?
19. Как организуется видеопамять при выводе статических и динамических видеоизображений?
20. Почему вывод линий при сегментной организации памяти замедляется?
21. Как выводятся линии при линейной организации видеопамяти?
22. Охарактеризуйте целочисленный алгоритм Брезенхема.
23. Как наиболее просто выводится статический рисунок и текст в графическом режиме?
24. Какие процедуры реализованы в программе, представленной в листинге 2.8?
25. Каковы проблемы вывода текста в графическом режиме при высоком разрешении?
26. Какие процедуры реализуются в листинге 2.9?
27. Охарактеризуйте программу, представленную в листинге 2.10?
28. Охарактеризуйте процедуры, представленные в листингах 2.11 и 2.12?
29. В чем заключаются проблемы масштабирования изображений в графическом режиме?
30. Для чего предназначена программа, представленная в листинге 2.13?

31. Охарактеризуйте проблемы реализации двумерной анимации.
32. Какова последовательность операций в цикле вывода движущихся объектов?
33. С чем синхронизируется вывод динамического изображения на экран монитора?
34. Чем определяется ширина зоны искажений?
35. Какие ограничения на вывод динамических объектов накладываются использованием одной видеостраницы?
36. Что применяется для ускорения вывода данных при 2D-анимации?
37. Для чего используются невидимые области экрана?
38. Как используется механизм переключения страниц при 2D-анимации?
39. Что реализует программа, представленная в листингах 2.14 - 2.20?
40. Что представлено в листинге 2.16?
41. Что представлено в листинге 2.17?
42. Что представлено в листинге 2.18?
43. Охарактеризуйте подпрограммы, представленные в листинге 2.19?
44. Что реализует программа, представленная в листинге 2.20?
45. Охарактеризуйте графический формат BMP.
46. Охарактеризуйте формат PCX для 256-цветных изображений.
47. Каковы особенности обмена между программами с помощью PCX-файлов?

В этом разделе использована информация из [5]. Листинги программ, реализующих вывод текста и чертежей в режимах HiColor и TrueColor, рисования линий по алгоритму Брезенхема для режима TrueColor32 и игру "Самолет и ракета" в режиме переключения страниц, приведены в [5].

2.3. Две видеосистемы в одном компьютере

Каким бы высококачественным ни был монитор, все равно, как правило, размер его экрана оказывается недостаточным. С такой ситуацией сталкиваются, например, биржевые работники — участники электронных торгов, которым необходимо одновременно отслеживать большие объемы разнородной информации: котировки валют, свежие новости, изменение различных индикаторов рынка и многое другое. Подобная ситуация возникает и при работе в графических редакторах и настольных издательских системах, когда множество открытых инструментальных окон сильно загромождают рабочее поле программы.

В этих случаях замена монитора на другой, с большим (17"—21") размером экрана, как правило, не решает проблемы. Кроме того, для работы за таким монитором требуется достаточно много места на рабочем столе.

Весьма эффективный и экономичный путь решения проблемы — использование двух мониторов стандартного размера (14"— 17"). При этом собственно для работы в приложении (например, графическом редакторе) можно использовать высококачественный монитор размером 15" или 17", а для размещения инструментальных панелей, окон диалога и т. п. — обычный монитор размером 14".

До недавнего времени для подключения к PC двух мониторов использовались только специализированные видеоадаптеры или профессиональные графические платы, оборудованные двумя VGA-разъемами (см. разд. 3.3.4).

Однако совместимость этих изделий с операционной системой и прикладным программным обеспечением оставляла желать лучшего.

2.3.1. Аппаратная совместимость видеоадаптеров

Ситуация радикально изменилась с появлением новой операционной системы Windows 98, в которую изначально заложена программная поддержка одновременно нескольких видеосистем. Как ни странно, но эта уникальная возможность, которую можно считать одним из наиболее ценных усовершенствований Windows 98 по сравнению с Windows предыдущих версий, неизвестна большинству пользователей PC.

Корректная работа двух видеоадаптеров, установленных в PC, возможна только в том случае, когда они имеют различные адреса портов ввода/вывода, кадрового буфера, Video BIOS и ряда других служебных областей памяти.

2.3.1.1. Видеоадаптеры с интерфейсом ISA

Классическим примером одновременного использования двух видеоадаптеров различных типов является комбинация видеоадаптеров MDA и CGA. Поскольку у видеоадаптеров с интерфейсом ISA адресное пространство жестко распределено, возможность их комбинирования в одной системе весьма ограничена и в настоящее время не представляет практического интереса (табл. 2.20). В частности, в PC нельзя одновременно установить два видеоадаптера VGA или два SVGA с интерфейсом ISA.

Таблица 2.20. Возможные комбинации видеоадаптеров с интерфейсом ISA

	MDA	CGA	EGA	MCGA	VGA, SVGA
MDA	○	●	●	●	●
CGA	●	○	●	○	○
EGA	●	●	○	●	○
MCGA	●	○	●	○	●
VGA, SVGA	●	○	○	●	○

Примечание. Здесь и далее: ● — да (есть); ○ — нет (отсутствует).

2.3.1.2. Видеоадаптеры с интерфейсами PCI и AGP

Иначе обстоит дело с видеоадаптерами SVGA с интерфейсом PCI (позднее — AGP). Как известно, устройства с интерфейсом PCI поддерживают протокол автоматического программного конфигурирования ресурсов Plug&Play. Адресное пространство видеоадаптеров с интерфейсом PCI задается не абсолютными адресами портов и других устройств адаптера, а лишь *смещением адресов относительно* некоторого базового адреса A_0h, который назначается видеоадаптеру операционной системой, исходя из конкретной конфигурации оборудования. Например, если в видеоадаптере VGA с интерфейсом ISA индексный регистр контроллера ЭЛТ всегда имеет адрес 3D4h (см. разд. 2.1.3), то в видеоадаптере с интерфейсом PCI этот адрес будет равен A_0h+3D4h. Таким образом, величина 3D4h выступает в роли не абсолютного адреса, а смещения.

Если базовым адресам двух видеоадаптеров присвоить такие значения, что адресные пространства каждого из них не будут перекрываться, это позволит избежать конфликтов между видеоадаптерами. Именно эту работу и выполняет Windows 98 совместно с программами BIOS. Таким образом, для одновременной установки в PC пригодны только видеоадаптеры с интерфейсом PCI или AGP (очевидно, что установить можно только один видеоадаптер с интерфейсом AGP, поскольку на материнской плате имеется единственный слот AGP).

2.3.2. Статус видеоадаптеров

Видеоадаптеры, установленные в систему, не будут равнозначными: один из них станет *первичным* (ведущим, *master),* а другой — *вторичным* (ведомым, *slave).* Различиями в статусе определяются разные возможности каждого из видеоадаптеров. В частности, аппаратное ускорение трехмерной графики и полноэкранный интерфейс, в том числе для программ, работающих под управлением DOS, доступно только для ведущего видеоадаптера.

Ведущим становится тот видеоадаптер, который первым будет обнаружен программой BIOS после включения компьютера. Это, в свою очередь, зависит от очередности опроса слотов шины ввода/вывода.

В слоты расширения материнских плат, рассчитанных на работу с CPU четвертого и пятого поколения (486, Pentium, Pentium MMX, AMD K5) и не поддерживающих спецификацию AGP, можно установить только видеоадаптеры с интерфейсом PCI. Порядок опроса слотов шины PCI обычно фиксирован и указан в технической документации на плату. В этом случае видеоадаптер, который необходимо сделать ведущим (обычно более мощный), следует установить в первый слот.

Если порядок опроса слотов не удалось определить по документации, это легко сделать опытным путем. Установите оба видеоадаптера в крайние слоты шины PCI и подключите к ним мониторы. Включите компьютер. Сообщения BIOS будут отображаться только на том мониторе, который подключен к ведущему видеоадаптеру

В слоты расширения материнских плат, предназначенных для работы с CPU шестого поколения (Pentium II, Celeron, Pentium III, AMD K6-II, K.6-III, K7 Athlon и т.д.), можно устанавливать видеоадаптеры как с интерфейсом PCI, так и с интерфейсом AGP. В первых моделях таких плат сначала опрашивались слоты шины PCI и лишь затем слот AGP, поэтому сделать ведущим более совершенный видеоадаптер с интерфейсом AGP вам вряд ли удастся. Проблему можно решить путем обновления версии BIOS, если на плате установлена микросхема Flash ROM BIOS.

В современных материнских платах очередность опроса слотов шин ввода/вывода определяется соответствующими настройками программы CMOS Setup. В частности, в CMOS Setup фирмы AWARD (AWARD BIOS) этот порядок определяется значением параметра Init Display First в разделе INTEGRATED PERIPHERALS: если выбрать значение AGP, то ведущим станет видеоадаптер, установленный в слот AGP (рекомендуемый вариант), если же выбрать значение PCI SLOT, то ведущим станет видеоадаптер с интерфейсом PCI.

2.3.3. Инсталляция двух видеосистем

2.3.3.1. Установка видеоадаптеров

Для упрощения процесса конфигурирования двух видеосистем в Windows 98 сначала следует установить только ведущую и тщательно ее настроить, поскольку вся информация о ходе дальнейшего конфигурирования системы будет отображаться при помощи этой ведущей видеосистемы. Опыт показывает, что в качестве ведущего может использоваться практически любой видеоадаптер с интерфейсом PCI или AGP. После его установки и настройки выключите компьютер и установите второй видеоадаптер.

Если после включения компьютера BIOS, а затем и Windows 98 корректно распознали установленный вторичный видеоадаптер, то в процессе загрузки Windows 98 на экране монитора, подключенного к этому адаптеру, последовательно появятся сообщения о версии Video BIOS и модели, а также сообщение об успешном опознавании видеоадаптера (рис. 2.19):

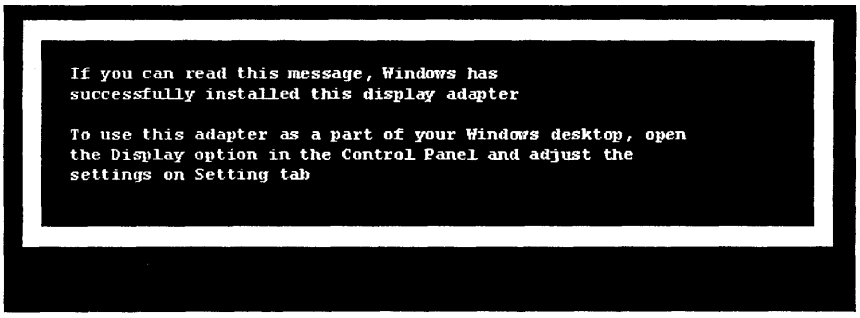


Рис. 2.19. Сообщение об успешном опознавании вторичного видеоадаптера

По окончании загрузки операционной системы наличие в системе второго видеоадаптера можно проконтролировать в диалоговом окне **Свойства: Экран** (рис. 2.20).

2.3.3.2. Программная поддержка Windows 98

К сожалению, в качестве вторичного Windows 98 поддерживает далеко не каждый видеоадаптер. Как правило, проблемы возникают с видеоадаптерами устаревших моделей (например, S3 Trio 32/64, CL5429 и т. п.) и особенно с изделиями типа Noname. Современные видеоадаптеры обычно работают без проблем.

Представителями компании Microsoft был опубликован перечень видеоадаптеров, которые могут использоваться в качестве вторичных под управлением Windows 98 (табл. 2.21).

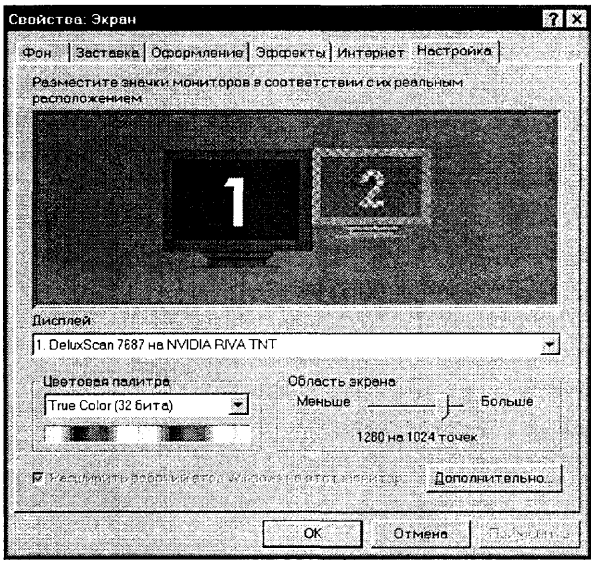


Рис. 2.20. Сообщение об успешном опознавании второго видеоадаптера в закладке Настройка диалогового окна **Свойства: Экран**

Однако этот перечень является далеко не полным; фактически он включает в себя те видеоадаптеры, драйверы с *мультимониторной поддержкой* которых фирма Microsoft включила в состав дистрибутива Windows 98. Если имеется подобный драйвер производителя видеоадаптера, он также может быть с успехом использован вместо встроенного в Windows 98 драйвера. В частности, в комплект поставки наиболее распространенных в настоящее время видеоадаптеров на основе Chipset Riva TNT/TNT2 входит драйвер вторичного видеоадаптера (файл NVMINI2.VXD).

Таблица 2.21. Перечень видеоадаптеров с мультимониторной поддержкой Windows 98

Модель видеоадаптера	Имя файла драйвера
ATI Mach 64 GX (GX, GXD, VT) ATI Graphics Pro Turbo PCI ATI Graphics Xpression ATI WinTurbo	ATIM64.drv
ATI Rage I, II, & II+ ATI All-In-Wonder ATI 3D Xpression+ PC2TV ATI 3D Xpression ATI 3D Xpression+	ATIM64.drv
ATI Rage Pro (AGP & PCI) ATI Xpert@Work, 4 & 8 MB ATI Xpert@p|ay, 4 & 8 MB ATI AII-ln-Wonder Pro	ATIR3.drv
S3 765 (Trio64V+) S3 Trio64V2 (DX/GX)	S3MM.drv
Diamond Stealth 64 Video 2001 STB PowerGraph 64V+ STB MVP 64 Miro TwinHead 22SD Hercules Terminator 64/Video Number Nine 9FX Reality 332 (S3 Virge) Number Nine 9FX Reality 334 (S3 Virge GX/2) Number Nine 9FX Reality 772 (S3 Virge VX) California Graphics V2/DX Videologic GraphicsStar 410 S3 AuroraCompaq Armada	S3MM.drv

Таблица 2.21. Перечень видеоадаптеров с мультимониторной поддержкой Windows 98 (продолжение).

Модель видеоадаптера	Имя файла драйвера
Cirrus 5436 Cirrus Alpine Cirrus 5446 STB Nitro 64V	CIRRUSMM.drv
S3 ViRGE S3 ViRGE (325) S3 ViRGE VX (988) S3 ViRGE DX (375/385) S3 ViRGE GX (375/385) Diamond Stealth 3D 2000 Diamond Stealth 3D 3000 Diamond Stealth 3D 2000 Pro Number Nine 9FX Reality 332 STB Nitro 3D STB Powergraph 3D STB Velocity 3D STB MVP/64 STB MVP/64 3D STB Workstation (2 & 4 output) Miro Crystal VR4000	S3V.drv
ET6000 Hercules Dynamite 128/Video STB Lightspeed 128	ET6000.drv
Trident 9685/9680/9682/9385/9382/9385 Jaton Video - 57P	TRIDPCI.DRV
InterGraphics Systems (IGS) CyberPro 2000A, 2MB	IGA2K.DRV
Permedia 2 Tl TVP4020, 8 meg PCI (Reference board) Tl TVP4020 8 meg AGP (Reference board) Diamond Fire GL Pro 1000 PCI Diamond Fire GL Pro 1000 AGP STB (Symmetric) Glyder MAX-2 PCI	GLINT. DRV

2.3.3.3. Программная настройка видеосистем

Теперь необходимо установить драйверы для вторичного видеоадаптера и монитора. Для этого щелкните мышью по изображению монитора с цифрой 2.

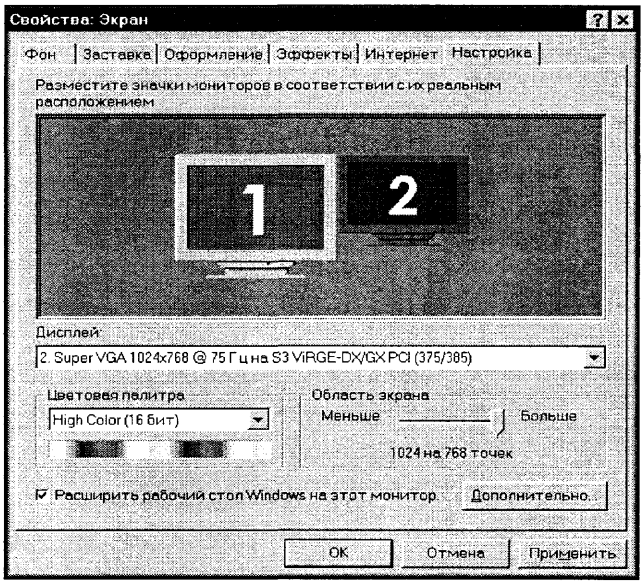


Рис. 2.21. Диалоговое окно Свойства: Экран после установки драйвера вторичного видеоадаптера

Если вторичный видеоадаптер или, по крайней мере, Chipset, на основе которого он выполнен, имеется в приведенном в табл. 2.18 перечне, то Windows 98 предложит установить для него наиболее подходящий драйвер. В противном случае установите драйвер производителя видеоадаптера. Как правило, именно на этом этапе и возникают проблемы с видеоадаптерами устаревших моделей. Для них драйверы, ориентированные на работу под управлением Windows 98, либо вообще отсутствуют, либо не поддерживают мультимониторную работу.

Для активизации вторичного видеоадаптера выберите опцию: Расширить рабочий стол Windows на этот монитор, при этом в строке Дисплей под номером 2 появятся наименования второго монитора и вторичного видеоадаптера (рис. 2.21).

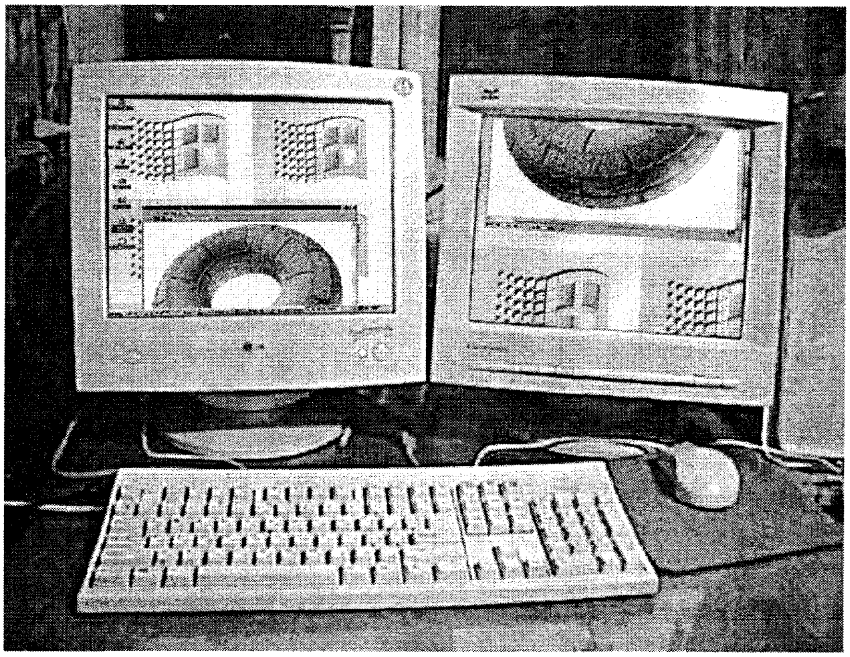


Рис. 2.22. Два монитора, подключенные к видеоадаптеру Matrox Millennium MGA G400

Отметим, что при этом все опции и настройки диалогового окна **Свойства: Экран** касаются только второй видеосистемы. Таким образом, щелчком мыши по изображению соответствующего монитора осуществляется переключение диалогового окна **Свойства: Экран** в режим настройки какой-либо одной из двух видеосистем.

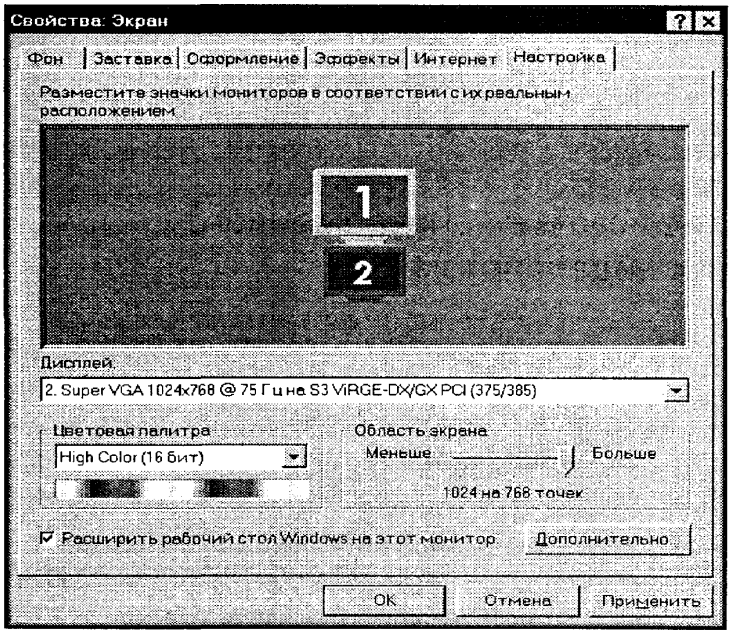


Рис. 4.21. Один из вариантов виртуального взаимного расположения двух экранов

Поскольку эти видеосистемы работают практически независимо друг от друга, для каждой из них можно выбрать любые разрешение, глубину цвета и частоту кадров (рис. 2.22). Можно также задать

виртуальное взаимное положение двух экранов, а точнее, направление, в котором объекты с экрана первого монитора можно перетаскивать на экран второго. Например, при таком расположении мониторов, как на рис. 2.20 и 2.21, перетаскивание объектов выполняется слева направо, а как на рис. 2.23, — сверху вниз.

В соответствии с расположением экранов, приведенным на рис. 2.23, рабочий стол будет выглядеть так, как на рис. 2.24.

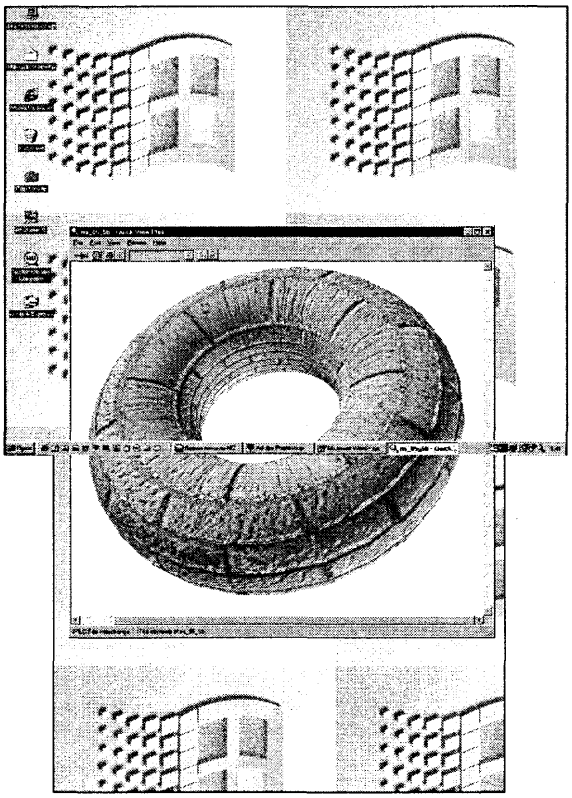


Рис. 2.24. Рабочий стол Windows 98 в случае виртуального расположения двух экранов друг над другом

Работа с двумя мониторами наиболее удобна и эффективна в том случае, когда используются одинаковые разрешение и глубина цвета. В этом случае не происходит искажения изображения объектов при перетаскивании их с одного экрана на другой. Что же касается частоты кадров, то ее, напротив, желательно установить различной; в некоторых случаях это поможет избежать взаимных помех на экранах мониторов.

2.3.4. Преимущества и недостатки одновременного использования двух видеосистем

Преимущества одновременной работы с двумя мониторами очевидны:

- значительно расширяется область рабочего стола Windows 98;
- различную по характеру информацию можно разместить на разных экранах, что значительно облегчит ее восприятие;
- появляется возможность просмотра больших изображений без использования прокрутки;
- приобретение второго монитора средних размеров (15"—17") и видеоадаптера для него (достаточно недорогого, на основе Chipset S3 Virge) обойдется значительно дешевле, нежели замена такого монитора на другой, размером 19"—21".

Существует, конечно, и ряд недостатков такого технического решения:

- второй монитор может использоваться не во всех режимах: при аппаратном ускорении 3D-графики, а также при работе в полноэкранном режиме, включая сеансы DOS в полноэкранном режиме, изображение будет выводиться только на первый, основной монитор;
- как правило, вторичный видеоадаптер работает заметно медленнее, чем первичный;
- не все модели видеоадаптеров могут использоваться в качестве вторичного (особенно устаревших моделей);
- некоторые программы неустойчиво работают при наличии в системе двух видеоадаптеров.

Однако в большинстве случаев эти недостатки либо несущественны, либо могут быть устранены путем более тщательной настройки системы. В целом же возможность одновременного использования двух мониторов является несомненным преимуществом Windows 98, по сравнению в Windows предыдущих

версий.

Контрольные вопросы.

1. В каком случае используют два монитора на одном компьютере?
2. Каковы варианты подключения двух мониторов?
3. Какая ОС имеет встроенные средства работы с несколькими видеосистемами?
4. Каковы условия корректной работы двух видеоадаптеров одновременно на одном ПК?
5. Каковы ограничения одновременной работы двух видеоадаптеров на шине ISA?
6. Как задается адресное пространство видеоадаптеров с интерфейсом PCI?
7. Каковы статусы двух видеоадаптеров в системе с PCI и их возможности?
8. Какой видеоадаптер назначается ведущим?
9. Как определить, какой слот PCI первым опрашивается BIOS?
10. В каких ПК можно установить одновременно видеоадаптеры с интерфейсами PCI и AGP?
11. Каким образом определяется прядок опроса слотов PCI и AGP?
12. Охарактеризуйте порядок установки двух видеоадаптеров?
13. Какие видеоадаптеры рекомендуется применять в качестве вторичных?
14. Охарактеризуйте программную настройку второй видеосистемы.
15. Охарактеризуйте преимущества одновременной работы с двумя мониторами.
16. Охарактеризуйте недостатки такого технического решения.

В данном разделе использованы материалы из [1]. Более подробную информацию об интерфейсах между видеоадаптером и материнской платой можно найти в [9, 10]. Информацию о настройке видеосистемы с двумя мониторами в среде Windows 98 можно найти в [16].

2.4. Мультимедийные функции видеоадаптера. TV-тюнер

Как уже отмечалось, современные видеоадаптеры, помимо своей основной задачи, могут выполнять ряд дополнительных, мультимедийных функций. До недавнего времени традиционным считался подход, заключающийся в том, что более высокое качество изображения может быть обеспечено лишь при выполнении этих функций отдельными, специализированными устройствами. Эти устройства выполнялись в виде отдельных карт расширения PC и соединялись с видеоадаптером через шину ввода/вывода, разъем Feature Connector или с помощью специальных высокочастотных кабелей. Очевидно, что в этом случае перечисленные мультимедийные функции прямого отношения к видеоадаптеру не имеют.

В настоящее время ситуация кардинально изменилась: все большее распространение получает другой подход. Современные Chipset для видеоадаптеров отличаются настолько высокой степенью интеграции, что большинство мультимедийных функций удобнее и дешевле реализовывать на одном мощном графическом процессоре. В результате на рынке появились мощные унифицированные видеоадаптеры, способные решать практически любые задачи: от воспроизведения цифрового видео до аппаратного ускорения 3D-графики. Примерами таких устройств могут служить изделия на основе Chipset S3 Savage 4, ATI Rage 128, Riva TNT(2) и др.

Итак, к мультимедийным функциям относятся следующие:

- прием телевизионных программ и просмотр их на экране монитора;
- прием радиопрограмм;
- ввод в PC отдельных кадров или целых видеофрагментов;
- формирование стандартного телевизионного сигнала (функция TV-out);
- воспроизведение Video-CD и DVD в полноэкранном режиме и др.

Реализация последних трех функций, связанных с обработкой видеосигналов (как аналоговых, так и цифровых), — предмет отдельного разговора, выходящего за рамки данной книги. Поэтому в настоящем разделе рассмотрим лишь первые две мультимедийные функции.

Итак, что же необходимо для приема радио- и просмотра телевизионных программ на экране монитора PC? Оказывается, совсем не так много, как может показаться на первый взгляд.

2.4.1. Устройство TV-тюнера

В самом деле, какие устройства составляют основу телевизора? Это кинескоп, приемник телевизионного радиосигнала, или телевизионный тюнер, блок обработки видеосигнала, или декодер (модуль цветности), и блок разверток. Видеосистема PC содержит практически те же элементы, за исключением телевизионного тюнера.

Монитор VGA (SVGA) работает по такому же принципу, как и обычный телевизор. Принципиальное различие между ними состоит в том, что в первом используется прогрессивная (построчная) развертка, а во втором — чересстрочная; различаются также частоты кадров и строк. Другим важным отличием является тип видеосигнала: если видеоадаптер изначально формирует компонентный видеосигнал (т. е. RGB-сигнал), то *полный цветовой телевизионный сигнал* является *композитным* (совмещенным), и для выделения из него сигнала RGB в телевизоре используется специальный блок — декодер.

Таким образом, для приема телевизионных программ достаточно снабдить компьютер приемником телевизионного радиосигнала, декодером (модулем цветности), а также устройством преобразования типа развертки. Устройство, включающее в себя перечисленные блоки, называют *TV-тюнером.*

На рис. 2.25 представлена обобщенная структурная схема TV-тюнера, выполняющего прием и обработку телевизионного радиосигнала.

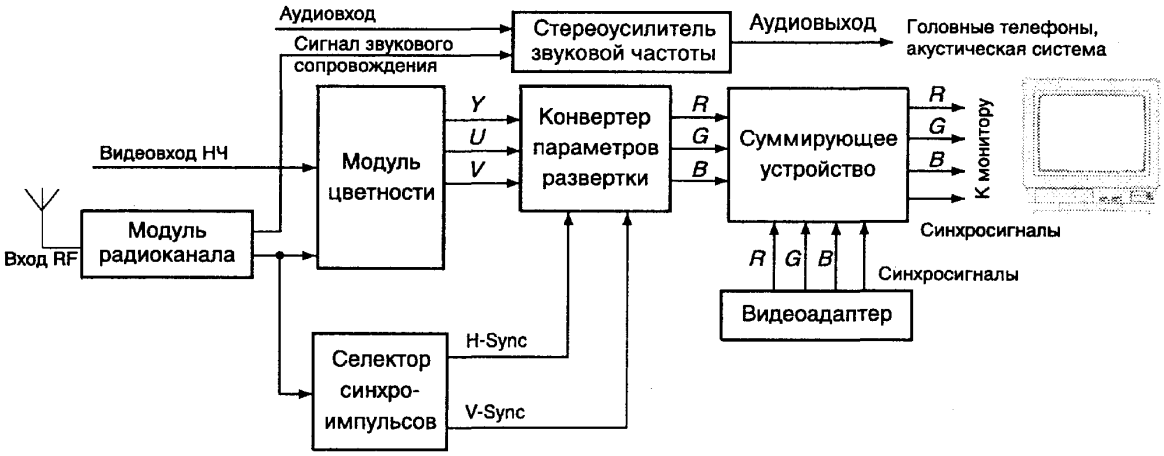


Рис. 2.25. Обобщенная структурная схема TV-тюнера

Модуль радиоканала осуществляет поиск высокочастотного радиосигнала, принятого антенной, его усиление, преобразование по частоте и детектирование (демодуляцию). На выходе радиоканала формируется телевизионный сигнал (ТВ-сигнал), содержащий сигнал яркости, сигналы цветности и сигналы синхронизации, а также частотно-модулированный сигнал звукового сопровождения. Селектор выделяет из видеосигнала импульсы кадровой и строчной синхронизации (*V-Sync* и *H-Sync),* декодер — сигнал яркости и два *цветоразностных сигнала.*

Устройство и алгоритм работы декодера зависят от того, с какими системами цветного телевидения (PAL, NTSC или SECAM) совместим TV-тюнер. В частности, при приеме ТВ-сигнала системы PAL на выходе модуля цветности формируются упоминавшиеся в разд. 2.2.1 сигналы *Y, U* и *V.* Наилучшим решением является использование универсального модуля цветности, поддерживающего все три системы.

Таким образом, на выходе декодера формируется компонентный видеосигнал, однако его формат отличен от используемого в видеоадаптерах формата RGB. Для преобразования цветового пространства (формата представления), а также типа развертки (из чересстрочной в построчную) используется специальный конвертер. В его состав входят специальные фильтры, позволяющие устранить неприятный эффект мерцания изображения и снизить негативное влияние внешних помех, имеющихся в радиоканале, а также собственных шумов модуля радиоканала.

Сформированные и отфильтрованные сигналы *R, G* и *B* подаются на суммирующее устройство, в котором они объединяются с выходным сигналом видеоадаптера и затем поступают на монитор. Способ объединения определяет форму представления телевизионного изображения на экране монитора: на полном экране или в окне.

Рассмотрим перечисленные выше этапы обработки TV-тюнером телевизионного радиосигнала более подробно.

2.4.2. Принцип работы TV-тюнера

2.4.2.1. Прием и преобразование телевизионного сигнала

В подавляющем большинстве моделей TV-тюнера используют радиоприемники, выполненные на интегральных микросхемах фирмы Philips. Такой радиоприемник представляет собой малогабаритный экранированный блок с антенным гнездом и низкочастотным входом. В нем реализованы все перечисленные этапы обработки радиосигнала, вплоть до формирования сигналов *R, G, B,* сигнала звукового сопровождения, а также сигналов синхронизации. Подаваемый на вход радиоприемника сигнал может быть как высокочастотным (RF — *Radio Frequency;* в TV-тюнерах используется единый для метрового и дециметрового диапазонов антенный вход с волновым сопротивлением 75 Ом), так и низкочастотным (видеосигнал), поэтому в качестве источника изображения могут использоваться:

- приемная антенна;
- линия кабельного телевидения;
- видеомагнитофон, подключенный к антенному входу либо к низкочастотным входам типа RCA (видеосигнал и звук отдельно);
- видеокамера.

При подаче на вход TV-тюнера низкочастотного сигнала первые этапы его обработки (в радиоканале) исключаются, благодаря чему качество изображения оказывается выше. Входной видеосигнал может быть композитным или компонентным, при этом для каждого из них должен быть предусмотрен соответствующий входной разъем. Сигнал звукового сопровождения подается на отдельный вход *Audio* (если в PC установлена звуковая карта, звуковой сигнал подается на ее вход Line In).

Композитный видеовход представляет собой коаксиальный разъем типа RCA («колокольчик», или «тюльпан»), по которому на вход тюнера подается полный цветной телевизионный сигнал. Он используется в бытовой видеоаппаратуре формата VHS.

Компонентный видеосигнал, как следует из его названия, предусматривает раздельную подачу сигнала яркости Y и сигналов цветности C и обозначается *Y/C* или S-Video. Раздельная подача сигналов Y и C позволяет повысить качество изображения за счет устранения типичных для композитного видеосигнала перекрестных искажений. Такой сигнал используется в полупрофессиональной видеозаписывающей аппаратуре формата S-VHS. В качестве соединителя используется 4-контактный разъем типа DIN.

Радиоприемник настраивается на требуемый телевизионный канал, усиливает, преобразует и производит демодуляцию телевизионного радиосигнала, выделяя из него низкочастотный видеосигнал. Настройка на телевизионные программы может осуществляться либо плавно, на любую частоту диапазона, либо дискретно, на одну из фиксированных частот. Первый вариант предпочтительнее, поскольку установленные производителем платы фиксированные частоты могут не совпадать с частотами местной телевизионной станции, особенно в дециметровом диапазоне.

Основной режим поиска программ и настройки — автоматический, под управлением программного обеспечения. Однако эти операции можно выполнить и вручную. В частности, можно самостоятельно выполнить точную настройку (она полезна при приеме слабых сигналов, когда система автоматической подстройки частоты малоэффективна).

После выделения видеосигнала декодер радиоприемника автоматически распознает систему цветного телевидения. Высококачественный TV-тюнер должен осуществлять прием цветного телевизионного сигнала трех основных систем: PAL, NTSC и SECAM, а также обеспечивать выбор требуемой звуковой поднесущей (в зависимости от региона и системы она может иметь два значения — 5,5 и 6,5 МГц). После декодирования сигнал яркости и цветоразностные сигналы преобразуются в сигналы основных цветов *R, G* и *B*.

Таким образом, на выходе радиоприемника формируются следующие сигналы:

- сигналы трех основных цветов *R, G* и *B;*
- кадровые и строчные синхроимпульсы;
- сигнал звукового сопровождения.

2.4.2.2. Преобразование параметров развертки

Сигналы *R, G, B* и синхросигналы поступают на вход преобразователя развертки, в котором телевизионные поля (полукадры) чересстрочной развертки преобразуются в полные кадры, необходимые для реализации построчной развертки на экране монитора. Кроме того, частоты кадров и строк преобразованного сигнала выбираются с учетом текущего режима работы видеосистемы PC. Следует отметить, что корректность выполнения этого преобразования очень сильно влияет на качество получаемого изображения. В дорогостоящих моделях TV-тюнеров на этом этапе используется так называемый *фильтр мерцаний (Flicker Filter),* позволяющий устранить характерные для телевизионного изображения дефекты, вызванные чересстрочной разверткой.

2.4.2.3. Смешение RGB- и ТВ-сигналов

Дальнейший алгоритм работы TV-тюнера зависит от способа смешения сигнала, соответствующего принятому телевизионному изображению, с выходным сигналом видеоадаптера. Это смешение может выполняться либо в аналоговом, либо в цифровом виде, с предварительным аналого-цифровым преобразованием выходного сигнала радиоприемника.

2.4.2.3.1. Аналоговое смешение

Аналоговое смешение конструктивно реализовать проще и дешевле, однако оно не обеспечивает изображения высокого качества, а также возможности просмотра телепрограмм в окне Windows. При цифровом смешении принятый телевизионный сигнал перед смешением должен быть оцифрован. Это усложняет аппаратную реализацию TV-тюнера, но позволяет осуществлять эффективное смешение двух сигналов и использовать многочисленные эффекты и удобные сервисные функции. Кроме того, появляется возможность осуществлять ввод в PC как отдельных кадров телевизионного изображения, так и коротких видео сюжетов.

Способ выполнения смешения предопределяет и способ соединения TV-тюнера с видеоадаптером.

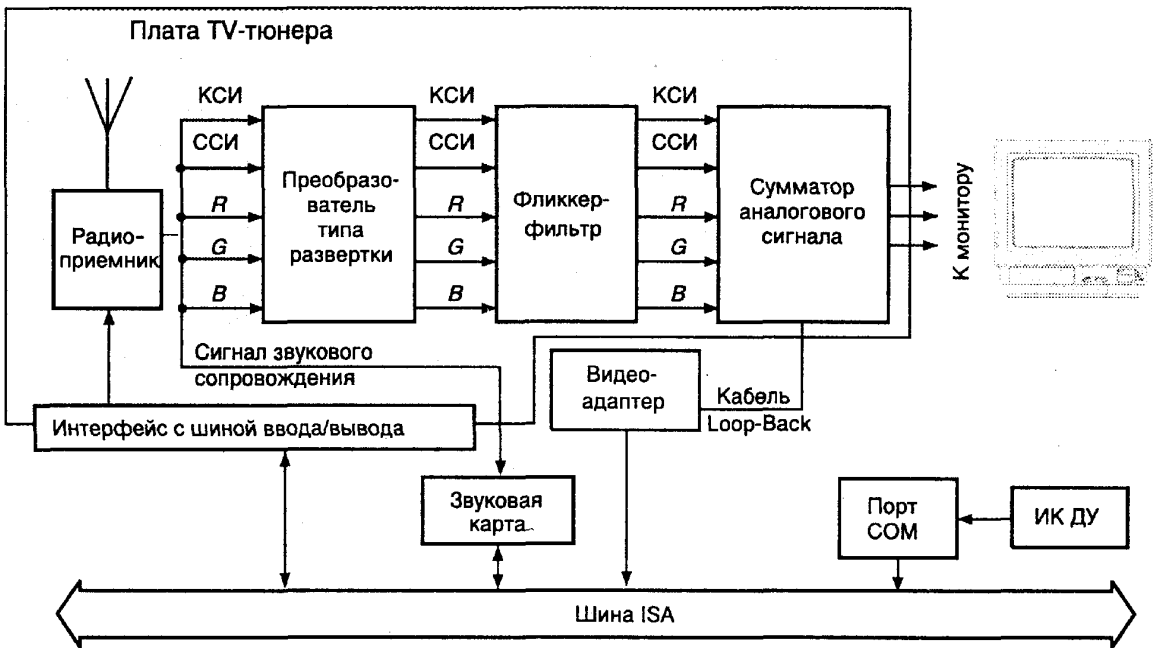


Рис. 2.26. Структурная схема TV-тюнера с аналоговым смешением

При аналоговом смешении TV-тюнер, выполненный на отдельной плате, соединяется с видеоадаптером специальным кабелем типа *LoopBack* (от англ. Loop Back — обратная петля). В этом случае видеосигнал с TV-тюнера как бы «врезается» в выходной сигнал видеоадаптера или вовсе заменяет его. Поскольку

«врезку» видеосигнала в выходной сигнал видеоадаптера выполнить довольно сложно, в недорогих моделях сигнал видеоадаптера просто заменяется видеосигналом с выхода TV-тюнера. Поэтому просмотр телепрограмм в окне с помощью такого TV-тюнера невозможен. При использовании совмещенной с видеоадаптером платы LoopBack-кабель не нужен.

TV-тюнеры с аналоговым смешением не предъявляют особых требований к быстродействию PC, так как через шину ввода/вывода передаются только команды управления тюнером, а основной поток информации в аналоговом виде поступает через LoopBack-кабель. Поэтому такие отдельные платы TV-тюнеров устанавливаются, как правило, в стандартный слот шины ISA. Структурная схема TV-тюнера с аналоговым смешением, выполненного в виде отдельной карты расширения, приведена на рис. 2.26.

2.4.2.3.2. Цифровое смешение

При цифровом смешении необходим специальный канал, по которому оцифрованный телевизионный сигнал будет подаваться на видеоадаптер (в том случае, если TV-тюнер и видеоадаптер выполнены как раздельные платы). В качестве такого канала могут использоваться:

- 26-жильный кабель, подключаемый к разъему Feature Connector видеоадаптера (таким разъемом снабжен практически каждый видеоадаптер SVGA);
- шина PCI.

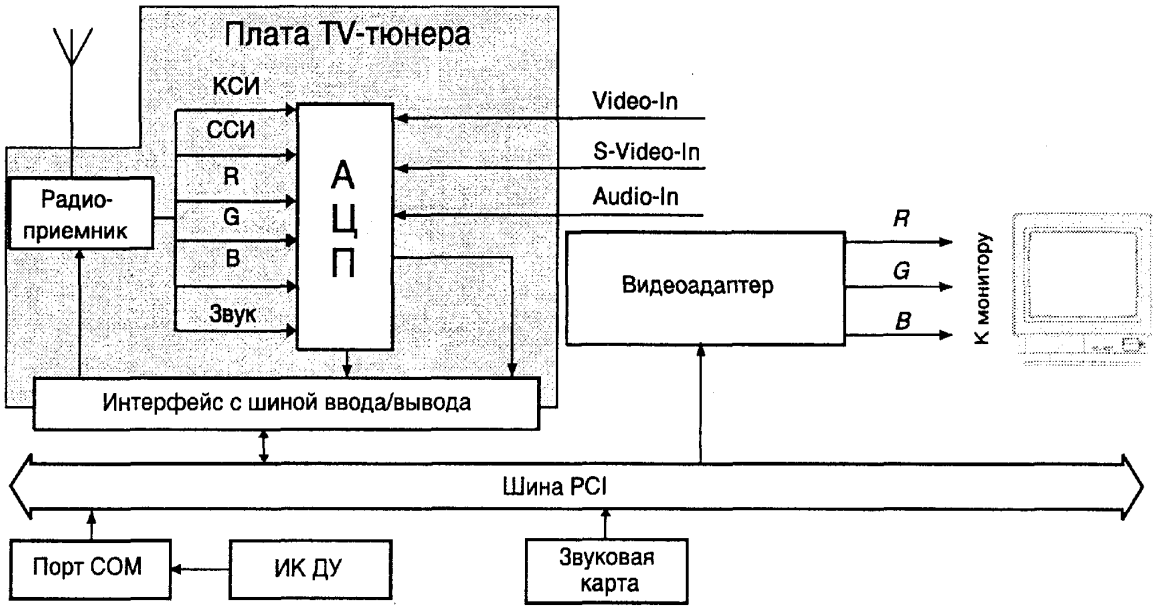


Рис. 2.27. Структурная схема TV-тюнера с цифровым смешением

В последнем случае через шину передается значительный информационный поток. Большинство операций по смешению, обработке изображения, созданию специальных эффектов и обеспечению сервисных функций (например, несколько окон предварительного просмотра разных ТВ-программ) выполняет центральный процессор PC. Поэтому компьютер должен обладать достаточной производительностью (обычно не ниже, чем Pentium-133).

Для управления процессом передачи информации через шину ввода/вывода необходимо соответствующее программное обеспечение. В частности, для таких TV-тюнеров в системе, как правило, должен быть установлен интерфейс прикладных программ DirectX. Более подробные сведения о назначении и структуре интерфейсов прикладного программирования будут приведены в разд. 3.1.3.

Очевидно, что наиболее широкие функциональные возможности и лучшее качество изображения в настоящее время могут обеспечить TV-тюнеры с цифровым смешением (рис. 2.27).

2.4.3. Дополнительные функции TV-тюнера

К числу дополнительных функций TV-тюнеров с цифровым смешением можно отнести следующие:

- возможность выбора размеров окна для просмотра телепрограмм (можно выбрать либо один из фиксированных размеров 192×144, 384×288, 768×576, кратных размеру телевизионного растра, либо произвольный, установив его вручную);
- захват отдельных кадров телевизионного изображения и сохранение их в одном из графических форматов (BMP, PCX, TIFF, JPG и др.);
- ввод в PC видеосигнала и сохранение видеопоследовательности на жестком диске в формате AVI;
- ввод в PC звуковых сигналов и сохранение их в виде файлов в формате WAV с возможностью регулировки частоты дискретизации и разрядности представления (при наличии на плате FM-

тюнера для приема радиовещательных стерео программ);

- возможность подключения портативной видеокамеры для проведения видеоконференций.

2.4.3.1. Захват отдельных кадров и ввод видеосигналов

Возможности TV-тюнеров по захвату отдельных кадров и вводу видео сюжетов подобны соответствующим функциям специализированных плат ввода/вывода видеосигналов. Однако нельзя сказать, что это равноценные устройства, особенно с точки зрения качества оцифровки видеоизображения. У плат ввода/вывода это основная функция; в процессе ввода они осуществляют сжатие видеоинформации (в частности, используется метод M-JPEG), а также выполняют ряд других функций, связанных с цифровой обработкой изображений. Они позволяют получить видео сюжет длительностью несколько десятков минут с высоким качеством изображения.

У TV-тюнеров, напротив, возможность ввода видеосигнала является дополнительной, поэтому в их структуре не предусмотрены средства аппаратного сжатия видеосигнала. Поскольку пропускная способность шины ввода/вывода ограничена, то весьма низкими оказываются и параметры оцифрованного видеоизображения (разрешение и частота кадров). В частности, TV-тюнеры осуществляют ввод видеосигналов в формате AVI *(Audio-Video Interleaved),* в котором сжатие видео- и аудиоинформации практически отсутствует. По этой причине максимальное разрешение вводимого сюжета при частоте 30 кадров/с не превышает 320×240 пикселов. Если же увеличить разрешение, система может не справиться с потоком данных, что неминуемо приведет к выпадению кадров из видеоряда.

2.4.3.2. Дистанционное управление

Непременным атрибутом телевизора является пульт инфракрасного дистанционного управления (ИКДУ). Снабжены таким устройством и большинство TV-тюнеров. Приемники ИКДУ обычно подключаются к PC через стандартный последовательный порт; в некоторых моделях предусмотрена возможность подключения к специальному разъему на плате TV-тюнера.

2.4.3.3. Прием радиопрограмм

Многие модели TV-тюнеров позволяют принимать не только телевизионные, но и радиовещательные стерео программы в диапазоне частот 87—108 МГц FM *(Frequency Modulation — с* частотной модуляцией). Поскольку для кодирования звукового сигнала в телевидении также используется частотная модуляция, реализация этой функции не требует каких-либо значительных доработок радиоприемника. Если в TV-тюнере используется цифровое смешение сигнала, то, как правило, предусматривается возможность и оцифровки звуковых сигналов и их записи в файлы формата WAV. При этом возможен выбор частоты дискретизации (11, 22 или 44 КГц) и разрядности представления данных (8 или 16 бит).

Все TV-тюнеры оборудуются аналоговым аудио выходом. Сигнал с него может быть подан либо на звуковую карту, либо на внешний усилитель.

Контрольные вопросы.

1. Каков был традиционный подход реализации мультимедийных функций, и каким он стал в настоящее время?
2. Перечислите мультимедийные функции видеосистемы.
3. Какие из этих функций требуют обработки видеосигналов (цифровых и аналоговых)?
4. Какие элементы содержит TV-тюнер?
5. Охарактеризуйте обобщенную структурную схему TV-тюнера.
6. Что содержит TV-сигнал, формируемый на выходе радиоканала?
7. Каковы функции селектора?
8. Чем определяется алгоритм работы декодера, и что формируется на его выходе?
9. Для чего служит конвертор, и какие элементы входят в его состав?
10. В чем назначение суммирующего устройства?
11. На какой элементной базе обычно выполняются радиоприемники TV-тюнера, и какие функции в нем реализуются?
12. Какие источники изображения могут подключаться ко входу радиоприемника?
13. Что представляет собой композитный и компонентный видеосигналы?
14. Как может настраиваться радиоприемник на частоту канала?
15. Какие системы цветного телевидения должен автоматически различать декодер радиоприемника?
16. Какие сигналы формируются на выходе радиоприемника?
17. Какую обработку сигналов RGB и синхронизации выполняет преобразователь развертки (конвертор)?
18. Какие алгоритмы смешения TV-сигнала с выходным сигналом видеоадаптера используются в TV-тюнерах?
19. Каковы сравнительные характеристики аналогового и цифрового смешения?
20. Как соединяется TV-тюнер и видеоадаптер при аналоговом смешении сигналов?

21. Какие каналы могут быть использованы для соединения TV-тюнера и видеоадаптера при цифровом смешении?
22. Охарактеризуйте структурную схему TV-тюнера с цифровым смешением?
23. Какое дополнительное программное обеспечение должно быть установлено для управления процессом передачи видеоинформации по шине ввода/вывода?
24. Перечислите дополнительные функции TV-тюнера.
25. Чем отличаются возможности TV-тюнера по захвату отдельных кадров и вводу видео сюжетов от возможностей специализированных устройств ввода/вывода видеосигналов?
26. Как может подключаться к ПК датчик пульта инфракрасного дистанционного управления?
27. Каковы возможности TV-тюнера по приему радиовещательных стерео программ?

В этом разделе были использованы материалы из [1].

3. АППАРАТНЫЕ СРЕДСТВА УСКОРЕНИЯ ТРЕХМЕРНОЙ ГРАФИКИ

Для создания иллюзии объемности видео объекта устройства, предназначенные для формирования стереоскопических изображений, должны уметь синтезировать стереопару, т.е. изображение одного и того же объекта под разными углами для каждого глаза. В простейшем случае для этого можно использовать две фотографии одного и того же объекта, отснятые под разными углами (так поступают, например, при съемке стереокино). Однако это возможно только в том случае, когда отображаемый объект существует реально. Если же объект является виртуальным (в играх, например), то необходимо *сконструировать (смоделировать) его объемное изображение,* причем задачу синтеза элементов стереопары вместо фотоаппарата должен выполнять PC. Иными словами, необходимо задать математическую модель объекта (т. е. каждую точку его поверхности) в трехмерной системе координат, аналитически рассчитать всевозможные зрительные эффекты (угол падения света, тени и т. п.), а затем корректно спроецировать трехмерный объект на плоский экран.

Совокупность приложений и задач, в рамках которых реализуется эта схема построения трехмерного изображения на экране монитора PC, называется *трехмерной графикой* или *3D (3-Dimentional* — трехмерный). Наиболее ярким примером трехмерной графики являются многочисленные компьютерные игры, поражающие реалистичностью создаваемых образов (Quake II, Final Realty, Tomb Rider и др.). Другой пример 3D-моделирования — реконструкция на компьютере внешнего вида доисторических животных, позволяющая настолько качественно и правдоподобно их «оживить», что у вас создается полная иллюзия того, что вы видите результат реальной видеосъемки, а не компьютерной анимации (например, документальный фильм «Прогулки с динозаврами»).

Объем вычислений, необходимый для моделирования трехмерного объекта, очень велик. Если все эти вычисления возложить на CPU, то производительность системы упадет настолько, что ни о какой работе в реальном масштабе времени говорить не придется. Для того чтобы обеспечить возможность видеть на экране проекцию динамического трехмерного объекта, в состав PC включают специализированное устройство, самостоятельно выполняющее основную часть операций по расчету трехмерной сцены. Такое устройство принято называть *ускорителем трехмерной графики* или *3D-акселератором.*

Подчеркнем еще раз, что 3D-акселератор необходим только в том случае, когда объемное изображение синтезируется компьютером, т. е. создается программно. Если же вы хотите посмотреть сканированную фотографию на экране монитора, то никакой 3D-акселератор не нужен.

Не следует путать 3D-акселераторы с устройствами формирования трехмерных изображений (VR-шлемы, 3D-очки и т. п. (см. [2], глава 3)) — это совершенно разные по назначению и принципам действия устройства: первые обеспечивают синтез видеосигнала трехмерного изображения, а вторые — его отображение с учетом эффекта бинокулярного зрения. Конечно, в идеале эти устройства должны использоваться совместно, дополняя друг друга. Однако, поскольку стереоскопические устройства отображения пока не получили в нашей стране широкого распространения (из-за достаточно высокой стоимости), в подавляющем большинстве случаев результат работы 3D-ускорителя приходится наблюдать в двумерном виде, т. е. на плоском экране монитора.

К аппаратным средствам ускорения трехмерной графики относятся:

- собственно 3D-акселераторы;
- специализированные высокоскоростные шины ввода/вывода, предназначенные для обеспечения работы 3D-акселератора (шины AGP и AGP Pro, см. разд. 3.2.1).

Первоначально 3D-акселераторы размещались на отдельных платах (отдельно от видеоадаптера), устанавливаемых в слот шины ввода/вывода на материнской плате, и соединявшихся с видеоадаптером специальным кабелем типа Pass-Through. Отсюда, собственно, и возник термин «3D-акселератор», под которым понималось самостоятельное устройство, предназначенное исключительно для работы с трехмерной графикой. При работе в обычном режиме 3D-акселератор выступал в роли пассивного элемента, включенного между видеоадаптером и монитором. Таким, в частности, был 3D-акселератор Voodoo производства фирмы 3Dfx, с появлением которого и связывают начало эры 3D на PC. Однако по мере увеличения степени интеграции микросхем 3D-акселераторы стали размещать непосредственно на плате видеоадаптера. Это позволило не только снизить стоимость видеосистемы, но и устранить необходимость в сквозном кабеле, использование которого порождало массу проблем.

Все современные видеоадаптеры содержат один мощный графический процессор, в состав которого, помимо ускорителя традиционной 2D-графики, входит 3D-акселератор. Поэтому в настоящее время термин «3D-акселератор» практически утратил свой первоначальный смысл: сегодня он означает не отдельную, специализированную плату, а универсальный видеоадаптер, в состав которого входит ускоритель как двумерной, так и трехмерной графики. Мы будем употреблять термин «3D-акселератор» в обоих смыслах, заостряя внимание на способе конструктивного исполнения лишь там, где это необходимо.

Практически все 3D-акселераторы для PC, представленные на рынке, в основном ориентированы на ускорение трехмерной графики в игровых приложениях, поэтому их условно называют *игровыми 3D-*

акселераторами. Кроме них, существуют *профессиональные графические ускорители,* предназначенные для систем автоматизированного проектирования, 3D-моделирования и т. п.

3.1. Основы технологии синтеза 3D-изображений

В общем случае синтез трехмерного изображения включает в себя следующие этапы:

- конструирование (расчет) трехмерного объекта на основе исходных данных, например результатов 3D-сканирования или математического описания объекта;
- расчет движения объекта и трансформации его формы;
- моделирование поверхности объекта с учетом различных факторов (освещения, отражений, рельефа и раскраски поверхности и т. п.);
- проецирование рассчитанного объекта на плоскость экрана с учетом всех возможных визуальных эффектов, обусловленных его строением, особенностями освещения, степенью прозрачности среды и т. п.

Все эти операции выполняются для того, чтобы аналитически рассчитать те особенности конечного изображения, которые будут создавать ощущение его объемности. Эти особенности основаны на вторичных факторах (см. [2], разд. 3.1.1), определяющих способность человека к объемному восприятию изображения. Напомним, что к этим факторам относятся:

- оценка расстояния до предмета на основе информации о его размерах (чем меньше объект — тем он дальше);
- оценка порядка наложения предметов друг на друга (тот, который выше, — ближе);
- определение глубины пространства за счет использования эффекта перспективы — визуального сближения параллельных линий, уходящих вдаль;
- анализ световых эффектов на предмете (теней, бликов и т. п.).

Отметим, что использование этих факторов позволяет создать ощущение объемности изображения не только при бинокулярном, но и при монокулярном зрении. Это дает возможность формировать изображения трехмерных объектов с помощью традиционных, двумерных устройств отображения, например мониторов.

3.1.1. 3D-конвейер

Процесс расчета трехмерного изображения объекта (точнее, его двумерной проекции) называется *3D-конвейером,* поскольку он выполняется в несколько последовательных этапов. В общем виде можно выделить следующие основные этапы 3D-конвейера:

1. Построение геометрической модели поверхности объекта путем задания трехмерных координат опорных (ключевых) точек, а также уравнений соединяющих их линий. В результате формируется так называемая *каркасная модель объекта,* или *wireframe.* На рис. 3.1 изображена каркасная модель тора с центром в начале координат $O(x, y, z)$, внутренним радиусом R_1 и радиусом сечения R_2.

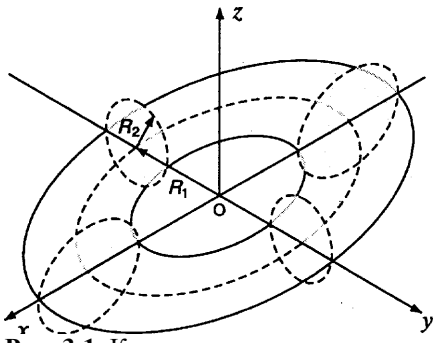


Рис. 3.1. Каркасная модель тора

2. Разбиение поверхности полученного объекта на элементарные плоские элементы: прямоугольники или треугольники. Этот этап называется также *тесселяцией (tesselation)* или *триангуляцией.* В результате тесселяции поверхность объекта, обычно имеющая искривленную форму, становится «граненой», т. е. представляет собой совокупность плоских многоугольников (треугольников или четырехугольников).

Чем больше таких граней и чем меньше их размер, тем точнее воспроизводится поверхность объекта (рис. 3.2).

Необходимость в этом этапе обусловлена тем, что дальнейшие операции гораздо проще выполнять с плоскими участками поверхности объекта. Чаще всего поверхность объекта разбивается на треугольники, поскольку тремя точками (вершинами треугольника) однозначно задается плоскость в пространстве.

На этапе тесселяции математическая модель объекта представляет собой совокупность трехмерных координат всех вершин *(вертексов,* от англ. *vertex)* элементарных треугольников, записанных в определенном порядке (иными словами, порядок следования координат вершин грани определяет направление вектора нормали к этой грани).

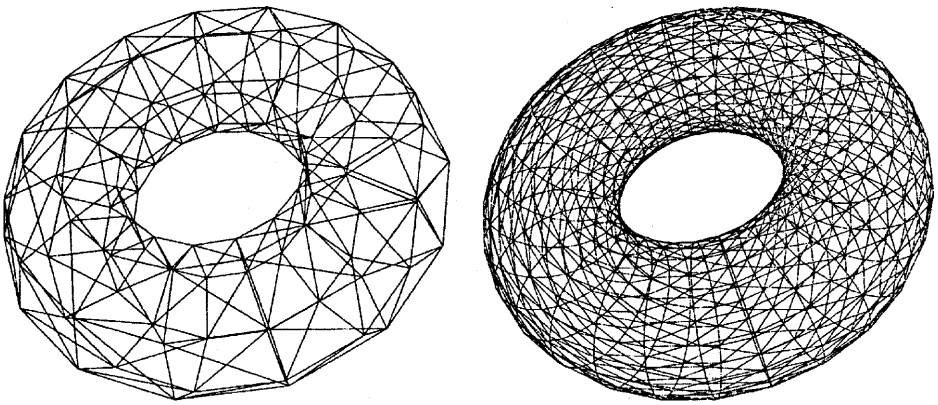


Рис. 3.2. Тесселяция объекта при помощи 420 (слева) и 2668 (справа) треугольников

3. Трансформация *(transformation).* На этом этапе моделируется движение объекта: его перемещение, вращение и изменение размеров (формы). Этап сводится к стандартному преобразованию трехмерных координат вершин граней (вертексов) и требует интенсивного использования операций матричной (линейной) алгебры и тригонометрических функций (рис. 3.3).

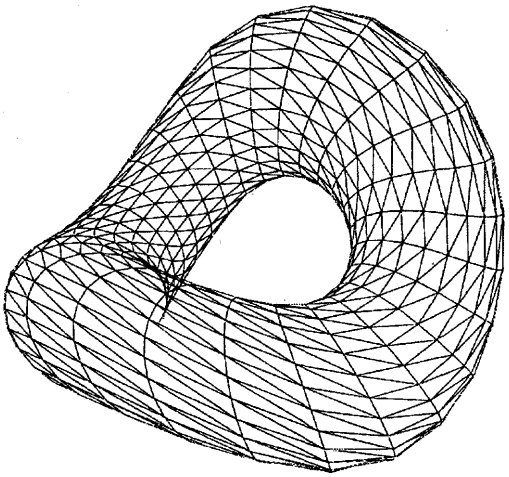


Рис. 3.3. Трансформация формы объекта путем изгиба и скручивания

4. Расчет освещенности *(lighting)* и затенения *(shading)* поверхности объекта. На данном этапе осуществляется расчет освещенности каждого элементарного треугольника в зависимости от его удаленности от источника света и угла падения светового луча (вот для чего элемент поверхности объекта должен быть плоским!).

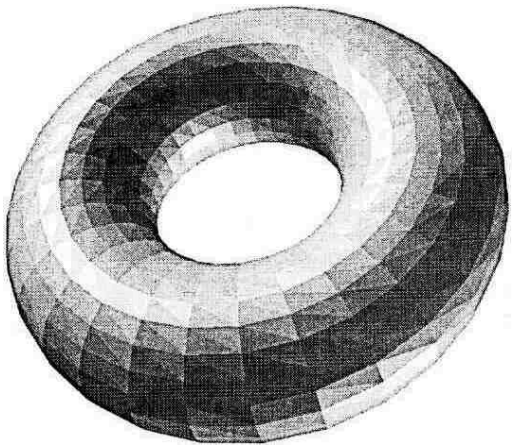


Рис. 3.4. Расчет освещенности объекта без использования алгоритмов интерполяции

Это позволяет сымитировать реальное распределение освещенности по поверхности объекта, подчеркивающее его объемность. Однако, если сделать освещенность всех точек каждого треугольника одинаковой, поверхность объекта будет выглядеть угловатой, состоящей из множества маленьких плоских граней (рис. 3.4). Для того чтобы устранить этот дефект, используют различные методы интерполяции значений освещенности, позволяющие плавно изменить освещенность каждой грани и скрыть резкие переходы между ними. Эти методы принято называть *методами затенения.* Наиболее распространенными из них являются *метод Гуро (Gouraud Shading)* и *метод Фонга (Phong Shading).*

5. Проецирование полученного трехмерного объекта на плоскость экрана. На данном этапе происходит первое, предварительное преобразование трехмерного объекта в совокупность двумерных координат вершин его граней, при этом запоминается информация о расстоянии каждой из вершин грани до плоскости проецирования (координата z). Это позволяет на следующих этапах корректно определить, какие части объекта окажутся видимыми, а какие — нет. Совокупность данных о глубине каждой из вершин, или значения ее третьей координаты *z,* называют *z-буфером.*

6. Обработка (настройка) координат вершин элементарных треугольников *(Triangle Setup),* полученных на предыдущих этапах. На этом этапе выполняется преобразование формы представления координат вершин граней: числа с плавающей точкой (вещественные) преобразуются в целые числа. Кроме того, производится сортировка вершин, отбрасывание задних граней *(culling)* и так называемая *субпиксельная коррекция.*

7. Удаление скрытых поверхностей — исключение из проецирования тех элементов поверхности объекта, которые оказываются невидимыми из точки наблюдения. Этот этап называется также HSR *(Hidden Surface Removal).* Строго говоря, процесс удаления скрытых поверхностей выполняется последовательно, в несколько этапов: на предыдущем (Triangle Setup), на котором обрабатываются векторные изображения, и на последующих, на которых на основе алгоритмов z-буферизации или z-сортировки обрабатываются уже растровые изображения.

8. Закраска элементарных треугольников, или *текстурирование,* выполняется путем наложения текстур *(texture mapping)* на поверхность объекта с учетом данных, полученных в результате выполнения предыдущих этапов (рис. 3.5), *Текстура (texture)* — это элемент обшивки объекта, т. е. изображение участка его поверхности. Она хранится в виде квадратной растровой картинки, состоящей из *текселов (Texel— TEXture ELement—* минимальный элемент текстуры). Иными словами, тексел — это пиксел текстуры. После наложения текстур (рис. 3.5, слева) каркасная модель (см. рис. 3.2) как бы «обрастает кожей» и становится похожа на реальный объект.

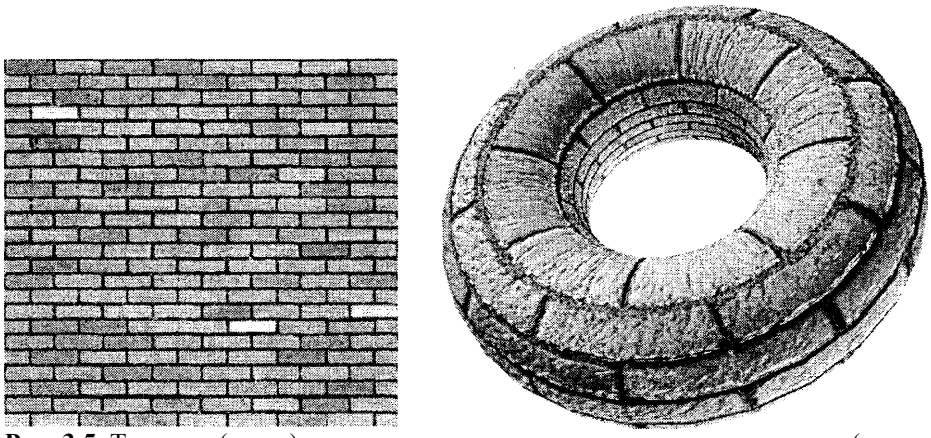


Рис. 3.5. Текстура (слева) и результат ее наложения на каркасную модель (справа)

Наложение текстур — это первый этап конвейера, выполняемый с **растровой** графикой. На данном этапе каждый треугольник заменяется частью текстуры. При этом значение каждого пиксела грани вычисляется по значению соответствующего тексела текстуры. Соответствие между пикселами и текселами обеспечивается путем проецирования текстуры на поверхность грани. В зависимости от формы объекта и его отдельных частей (для объектов сложной формы) используются различные системы проекционных координат: прямоугольная, сферическая, цилиндрическая и т. д.

Текстурирование — довольно трудоемкий процесс, поскольку обработке подвергается растровая графика. При модификации растрового изображения возникают многочисленные дефекты, поэтому на данном этапе применяются различные приемы коррекции изображения: мипмэппинг *(MIP-mapping—* использование текстур с различным разрешением), коррекция перспективы, фильтрация и др.

9. Моделирование эффектов прозрачности и полупрозрачности. На этом этапе на основе информации о взаимной прозрачности объектов и среды выполняется коррекция цвета пикселов — так называемое *альфа-смешение (alpha-blending)* и *затуманивание (fogging).*

10. Коррекция дефектов картинки, вызванных угловатостью линий на границах (краях) объектов, — так называемый *антиалиасинг (antialiasing).*

11. Интерполяция недостающих цветов — *дизеринг (dithering)* — применяется в том случае, когда в текущем видеорежиме 3D-акселератора для кодирования цвета пиксела используется менее 24 бит (например, в режиме High Color). Это приводит к незначительному ухудшению разрешения (резкости), но позволяет эффективно имитировать отсутствующие в палитре цвета.

12. Окончательное формирование кадра. Кадр формируется в кадровом буфере — области локальной памяти 3D-акселератора, в которую помещается спроецированное двумерное изображение. Кадровый буфер используется для формирования выходного аналогового видеосигнала 3D-ускорителя, точно так же, как это делается в обычном видеоадаптере VGA (в совмещенных 2D/3D-акселераторах под кадровый буфер отводится часть видеопамяти).

Для ускорения процесса создания изображения используется механизм *двойной буферизации,* заключающийся в том, что в видеопамяти выделяется область для хранения одновременно двух смежных кадров (по сути, два кадровых буфера): построение следующего кадра начинается еще до того, как закончится отображение (передача в RAMDAC) текущего. В результате удается избежать ненужного «простоя» акселератора и обеспечить более высокую частоту смены кадров.

13. Пост-обработка *(Post-processing)* используется в том случае, когда требуется реализовать какие-либо двумерные эффекты над подготовленным кадром как единым целым.

Этапы 1 - 6 образуют так называемую *геометрическую стадию* 3D-конвейера. На этой стадии выполняются интенсивные тригонометрические вычисления, в большинстве случаев центральным процессором PC, так как он содержит математический сопроцессор с аппаратной поддержкой необходимых функций: вычисление значений тригонометрических функций, обработка чисел с плавающей точкой и др. Лишь профессиональные модели 3D-акселераторов содержат специальный *геометрический процессор,* частично или полностью выполняющий все операции по расчету геометрии объекта.

Подавляющее большинство игровых 3D-акселераторов не имеют геометрического процессора. В лучшем случае возможна аппаратная реализация этапа 6 (Triangle Setup). Однако в новейших (и достаточно дорогостоящих) игровых 3D-акселераторах наиболее известных производителей (например, nVidia GeForce 256, ATI Radeon 256, S3 Savage 2000) геометрический процессор уже имеется и обеспечивает аппаратное ускорение геометрической стадии 3D-конвейера.

В последнее время наметилась тенденция к сближению характеристик игровых и профессиональных 3D-акселераторов. Так, практически все новые модели игровых ускорителей оборудуются геометрическим процессором.

Отметим, что на геометрической стадии все операции выполняются с векторной графикой. Для удобства вычислений используются числа с плавающей точкой.

Этапы 7 - 13 образуют *стадию прорисовки объекта,* или *стадию рендеринга (rendering*— рисование, визуализация). На этой стадии все действия выполняются уже с растровыми объектами, состоящими из отдельных дискретных элементов — пикселов и текселов.

Иногда термин «рендеринг» употребляют для обозначения всего 3D-конвейера, а этапы 7 - 13 называют «растеризацией», подчеркивая тем самым отличие этой стадии конвейера от предыдущей, на которой все операции выполнялись с векторной графикой. Мы же под «рендерингом» будем понимать стадию 3D-конвейера, включающую в себя этапы 7 - 13.

Стадия рендеринга является наиболее сложной и трудоемкой, поэтому именно здесь необходимо аппаратное ускорение. Большинство современных 3D-ускорителей как раз и предназначены для выполнения рендеринга на аппаратном уровне и различаются лишь количеством реализуемых функций.

Приведенные последовательность и содержание этапов 3D-конвейера не являются жесткими. В зависимости от особенностей архитектуры 3D-акселератора и программного обеспечения последовательность этапов может быть изменена, а в некоторых случаях отдельные этапы могут быть и вовсе исключены из 3D-конвейера.

Контрольные вопросы.

1. Что значит "сконструировать (смоделировать)" объемное изображение?
2. Что понимается под трехмерной графикой (3D-графика)?
3. Для чего используются 3D-ускорители (3D-акселераторы)?
4. Что относится к аппаратным средствам 3D-графики?
5. Как подключались первые 3D-ускорители?
6. Что представляют собой современные 3D-акселераторы?
7. Какие этапы, в общем случае, включает в себя синтез трехмерного изображения?
8. Какие вторичные факторы определяют способность человека к объемному восприятию изображения?
9. Что называют 3D-конвейером?
10. Сколько основных этапов можно выделить в 3D-конвейере?
11. Охарактеризуйте первый этап конвейера - построение геометрической модели поверхности объекта.
12. Охарактеризуйте второй этап - разбиение поверхности объекта на элементарные плоские объекты

(треугольники, прямоугольники).

13. Что понимается под тесселяцией, триангуляцией и вертексом?
14. Какие основные операции используются на третьем этапе - этапе трансформации?
15. Какие расчеты ведутся на четвертом этапе и какими методами?
16. Что производится на пятом этапе - этапе проецирования?
17. Что выполняется на этапе обработки (настройки) координат (этап 6)?
18. Что делается на седьмом этапе?
19. Что такое текстурирование и тексел (этап 8)?
20. Что такое мипмэппинг?
21. Что производится на этапе 9?
22. Что производится на этапе 10?
23. Что производится на этапе 11?
24. Как производится окончательное формирование кадра на этапе 12?
25. Что может выполняться на этапе 13?
26. Где выполняется реализация геометрической стадии 3D-конвейера (этапы 1 - 6) и с каким представлением чисел?
27. Что понимается под рендерингом, на каких этапах он реализуется и над какими объектами?
28. На каких этапах конвейера работает большинство 3D-ускорителей?

В данном разделе использованы материалы из [1]. Информацию об устройствах отображения стерео и 3D изображений можно найти в [1, 2].

3.1.2. Методы коррекции трехмерных изображений

Практически каждый 3D-акселератор поддерживает на аппаратном уровне все этапы 3D-конвейера. Однако в процессе 3D-синтеза на изображении неизбежно возникают дефекты. Для коррекции этих дефектов 3D-акселератор должен выполнять ряд дополнительных функций. От качества выполнения и наличия аппаратной поддержки этих функций зависят качество конечного изображения и общая производительность 3D-системы. Набор реализованных дополнительных функций по коррекции дефектов изображения существенно зависит от класса ускорителя (игровой или профессиональный).

Рассмотрим основные методы коррекции изображения, реализованные в современных 3D-ускорителях. Может возникнуть закономерный вопрос: «Если книга посвящена аппаратным средствам, имеет ли смысл рассматривать методы и алгоритмы?». Имеет, и этому есть несколько причин.

Во-первых, описание этих методов позволит лучше понять, что же это все-таки такое, 3D-конвейер, и какие задачи выполняет 3D-акселератор в процессе его реализации.

Во-вторых, именно набор аппаратно реализуемых функций по коррекции изображения является важнейшей характеристикой 3D-акселератора, которую необходимо учитывать при его выборе. Как правило, в документации на устройство этот набор выглядит в виде списка лаконичных англоязычных терминов, разобраться в котором неискушенному пользователю очень сложно. Понимание же сути этих функций позволит сделать правильный выбор.

И, наконец, понимание сущности выполняемых акселератором действий поможет правильно выполнить конфигурирование прикладной программы (игры) и использовать 3D-акселератор с максимальной эффективностью.

3.1.2.1. Затенение

После разбиения на элементарные треугольники (этап 2) поверхность объекта становится «граненой», т. е. состоящей из большого числа плоских граней с ровными краями. Для создания ощущения объемности объекта необходимо корректно рассчитать освещенность его поверхности, т. е. каждой грани. Эта задача решается в процессе *затенения.*

Расчет освещенности точек каждой грани выполняется на основе закона освещенности, согласно которому *освещенность точки поверхности прямо пропорциональна силе света и косинусу угла α между направлением падения света и нормалью к поверхности и обратно пропорциональна квадрату расстояния до источника света* (рис. 3.6).

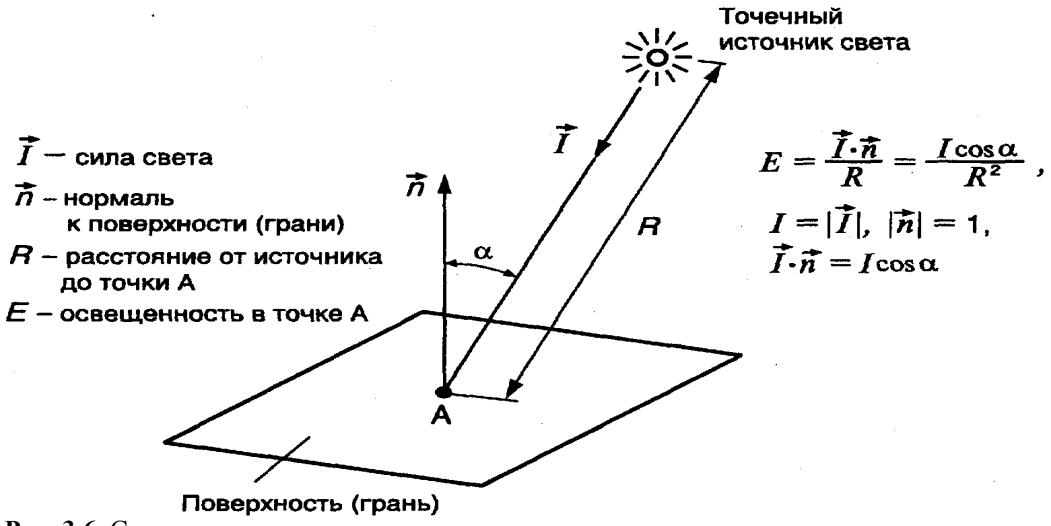


Рис. 3.6. Схема, поясняющая закон освещенности

Очевидно, что рассчитать освещенность грани значительно легче в том случае, когда она плоская (при этом нормаль к ней сохраняет постоянную ориентацию). Именно стремлением получить плоские грани и обусловлена необходимость в этапе 2 3D-конвейера — тесселяции.

В современных игровых 3D-акселераторах реализуются три метода затенения:

- плоское (простое);
- по методу Гуро;
- по методу Фонга.

Кратко рассмотрим эти методы.

3.1.2.1.1. Плоское (простое) затенение

Этот метод является наиболее простым и очевидным: освещенность всех точек грани считается

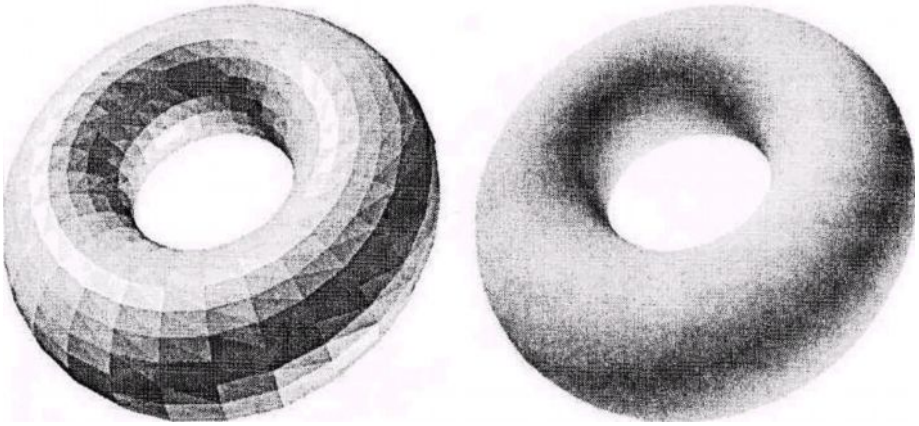


Рис. 3.7. Плоское затенение (слева) и затенение по методу Гуро (справа)

одинаковой и определяется только ориентацией грани относительно падающего света, т. е. косинусом угла α (см. рис. 3.6). В этом случае яркость граней, расположенных на изгибах поверхности, будет резко отличаться от яркости соседних граней, в результате чего поверхность объекта будет иметь ярко выраженную «граненую» структуру.

Поскольку каждая грань плоская, такое затенение называют *Flat Shading— плоское затенение пли простое затенение.* При его использовании подчеркивается дискретный характер тесселяции и объект выглядит угловатым (рис. 3.7, слева). Достоинством плоского затенения является простота и, как следствие, высокая скорость его реализации.

3.1.2.1.2. Затенение по методу Гуро

Метод Гуро, изобретенный еще в 1971 г. и названный так по имени автора, был предназначен для устранения «угловатости», характерной для плоского затенения, и создания иллюзии кривизны плоского объекта (рис. 3.7, справа).

Затенение по методу Гуро выполняется на основе результатов, полученных при плоском затенении. Расчет освещенности по методу Гуро выполняется в два этапа. Сначала вычисляются значения освещенности на ребрах грани (как среднее арифметическое освещенности для смежных граней). На втором этапе выполняется линейная интерполяция освещенности вдоль параллельных линий от одного ребра к другому внутри грани (рис. 3.8).

Линейный характер изменения освещенности вдоль линии позволяет составить простые и эффективные алгоритмы затенения по методу Гуро, не требующие значительных вычислительных затрат.

Интерполяции может подвергаться не только освещенность, но и цвет. В игровых 3D-акселераторах затенение Гуро является наиболее распространенным приемом, позволяющим быстро и легко устранить дефекты, полученные в результате тесселяции поверхности объекта.

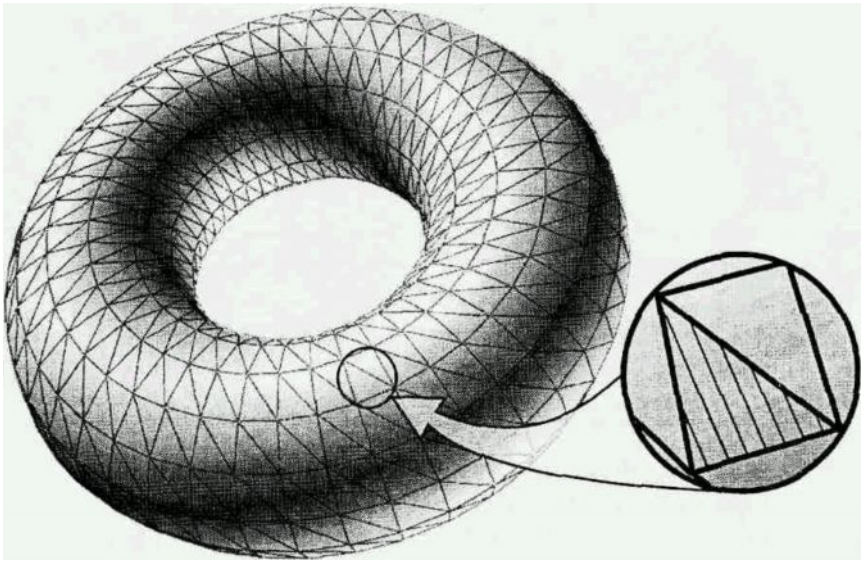


Рис. 3.8. Схема расчета освещенности по методу Гуро

Метод Гуро применим только для небольших граней, расположенных на значительном расстоянии от источника света. Если же размер грани достаточно велик, то расстояние от источника света до ее центра будет значительно меньше, чем до ее вершин, и, согласно закону освещенности, центр грани должен быть

освещен сильнее ребер. Однако модель изменения освещенности, принятая в методе Гуро, предполагает линейное изменение яркости в пределах грани и не позволяет сделать середину грани ярче, чем ее края. В итоге на изображении появляются участки с неестественной освещенностью.

3.1.2.1.3. Затенение по методу Фонга

От недостатка, присущего методу Гуро, свободно затенение Фонга, названное также в честь автора Ву Тонг Фонга (Wu Tong Phong). Данный метод, как и два предыдущих, предполагает постоянство расстояния от источника света до каждой из вершин грани, но учитывает изменение угла падения света для различных точек грани (в идеале — для всех точек, как минимум — для вершин грани). Это позволяет более корректно рассчитать освещенность грани, особенно в том случае, когда источник света расположен на небольшом расстоянии от нее.

Другой важной особенностью затенения по методу Фонга является то, что для каждой точки грани используется своя «нормаль», направление которой может отличаться от направления истинной нормали к грани. Строго говоря, эти «нормали» не являются таковыми в полном смысле. Всем известно, что векторы, перпендикулярные одной плоскости (в данном случае — грани), должны быть параллельны друг другу. Однако «нормалью» эти векторы названы не случайно: согласно методу Фонга, они ориентируются таким образом, что являются нормалями к некоторой воображаемой выпуклой или вогнутой поверхности, которой

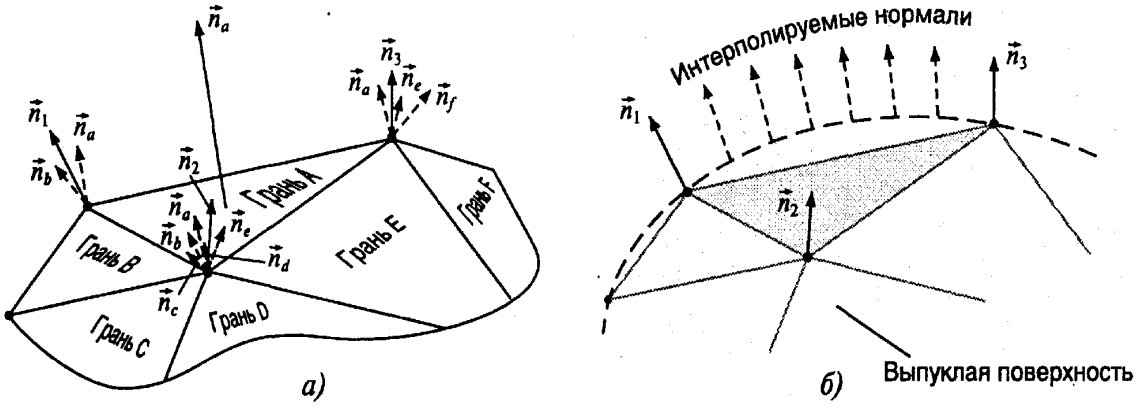


Рис. 3.9. Интерполяция нормалей по методу Фонга

принадлежат вершины грани. В дальнейшем, при расчете освещенности каждой точки грани, в соотношении для закона освещенности используются именно эти «нормали», т. е. фактически рассчитывается освещенность не плоской грани, а искривленной поверхности, что позволяет эффективно сгладить «граненую» структуру объекта.

Сказанное выше поясняется рис. 3.9. Для каждой грани, помимо основной, истинной нормали $\vec{n}$, рассчитываются три «нормали» $\vec{n}_1, \vec{n}_2, \vec{n}_3$. Каждая из них определяется как геометрическая сумма нормалей к граням, которым принадлежит данная вершина (рис. 3.9, *а):*

$$\vec{n}_1 = \frac{1}{2}\left(\vec{n}_A + \vec{n}_B\right);\ \vec{n}_2 = \frac{1}{5}\left(\vec{n}_A + \vec{n}_B + \vec{n}_C + \vec{n}_D + \vec{n}_E\right);\ \vec{n}_3 = \frac{1}{3}\left(\vec{n}_A + \vec{n}_E + \vec{n}_F\right).$$

Благодаря такому способу определения $\vec{n}_1, \vec{n}_2, \vec{n}_3$ каждая дополнительная «нормаль» является истинной нормалью к некоторой воображаемой выпуклой или вогнутой поверхности, проходящей через вершины граней (рис. 3.9, *б).* Далее, для каждой точки грани рассчитывается собственная «нормаль» как результат интерполяции «нормалей» $\vec{n}_1, \vec{n}_2, \vec{n}_3$ по поверхности грани. Этот этап затенения по методу Фонга напоминает линейную интерполяцию освещенности по методу Гуро, однако в данном случае интерполируется не скалярная величина (освещенность), а вектор «нормали», задаваемый тремя координатами. По этой причине реализация затенения по методу Фонга требует значительных вычислительных затрат и редко используется в реальном масштабе времени.

Объем необходимых вычислений можно значительно сократить, если затенение Фонга выполнять с помощью так называемых *карт затенения* (рис. 3.10), подготовленных заранее (этот алгоритм предложил Майкл Абраш).

Карта затенения представляет собой изображение плоской поверхности, освещаемой точечным источником света, расположенным над ее центром. Освещенность каждой точки этой карты рассчитывается на основе закона освещенности. Заметим, что точки карты, лежащие на концентрических окружностях, имеют одинаковую освещенность, поскольку для них угол между нормалью к поверхности и направлением на источник света одинаков.

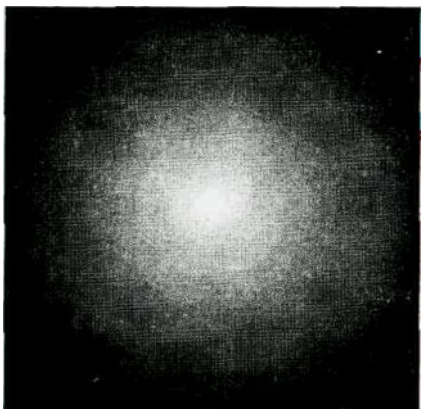


Рис. 3.10. Карта затенения

Карта затенения используется в качестве шаблона, задающего распределение освещенности в пределах грани. Достаточно наложить соответствующим образом контур грани на карту — и освещенность каждой точки грани станет известной (рис. 3.11).

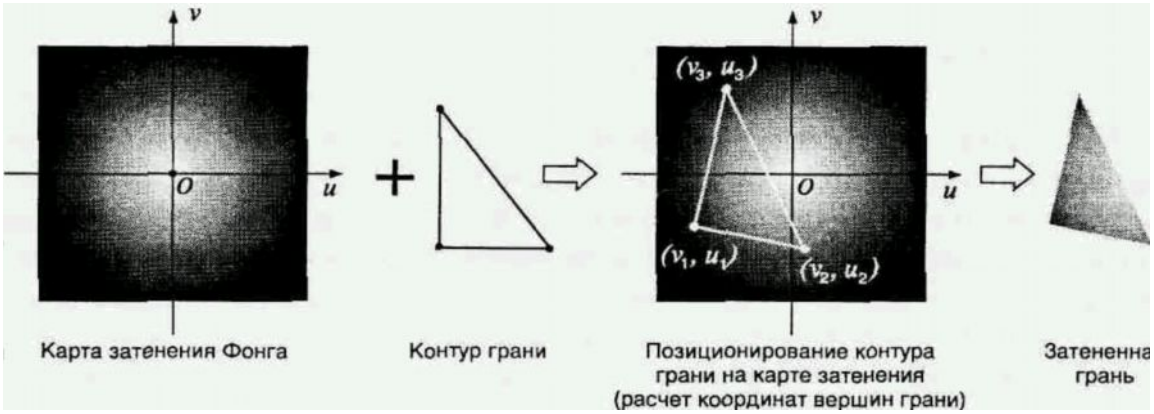


Рис. 3.11. Методика использования карты затенения

Как же определить координаты вершин контура грани на карте затенения? Очевидно, что расположить контур грани надо таким образом, чтобы освещенность каждой вершины в точке карты с координатами *(u, v)* соответствовала фактической освещенности вершины грани на поверхности объекта. С одной стороны, освещенность точки на карте затенения убывает по мере ее удаления от центра карты, с другой — фактическая освещенность уменьшается по мере увеличения угла между нормалью и направлением на источник света. Следовательно, необходимо установить взаимосвязь между смещением вершины контура грани на карте затенения вдоль осей координат *u, v* и изменением угла, под которым освещается вершина грани.

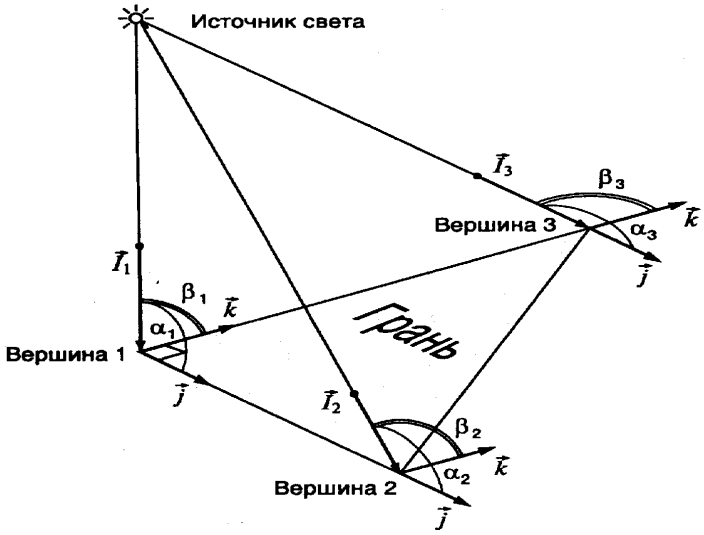


Рис. 3.12. Расчет координат вершин контура грани на карте затенения

Эта задача решается следующим образом. Каждая грань задается прямоугольным треугольником, вдоль катетов которого направлены единичные векторы:

Орты $\vec{j}$ и $\vec{k}$ (рис. 3.12):

$$\vec{j} = \{x_j, y_j, z_j\},\ |\vec{j}| = \sqrt{x_j^2 + y_j^2 + z_j^2} = 1;$$

$$\vec{k} = \{x_k, y_k, z_k\},\ |\vec{k}| = \sqrt{x_k^2 + y_k^2 + z_k^2} = 1.$$

Для каждой из вершин грани рассчитываются координаты векторов падающего света $\vec{I}_1$, $\vec{I}_2$, $\vec{I}_3$, при этом его интенсивность (света) для всех трех вершин считается одинаковой и равной единице:

$$\vec{I}_1 = \{x_{I_1}, y_{I_1}, z_{I_1}\},\ |\vec{I}_1| = \sqrt{x_{I_1}^2 + y_{I_1}^2 + z_{I_1}^2} = 1;$$

$$\vec{I}_2 = \{x_{I_2}, y_{I_2}, z_{I_2}\},\ |\vec{I}_2| = \sqrt{x_{I_2}^2 + y_{I_2}^2 + z_{I_2}^2} = 1;$$

$$\vec{I}_3 = \{x_{I_3}, y_{I_3}, z_{I_3}\},\ |\vec{I}_3| = \sqrt{x_{I_3}^2 + y_{I_3}^2 + z_{I_3}^2} = 1.$$

Теперь для каждой вершины грани рассчитываются две проекции вектора падающего на орты $\vec{j}$, $\vec{k}$ света. Эти проекции и определяют искомое смещение каждой из вершин контура грани на карте затенения. Напомним, что проекция одного вектора на другой вычисляется как результат их скалярного произведения:

$$u_1 = \vec{I}_1 \cdot \vec{j} = |\vec{I}_1| \cdot |\vec{j}| \cdot \cos\alpha_1 = \cos\alpha_1 = x_{I_1} x_j + y_{I_1} y_j + z_{I_1} z_j;$$

$$v_1 = \vec{I}_1 \cdot \vec{k} = |\vec{I}_1| \cdot |\vec{k}| \cdot \cos\beta_1 = \cos\beta_1 = x_{I_1} x_k + y_{I_1} y_k + z_{I_1} z_k;$$

$$u_2 = \vec{I}_2 \cdot \vec{j} = |\vec{I}_2| \cdot |\vec{j}| \cdot \cos\alpha_2 = \cos\alpha_2 = x_{I_2} x_j + y_{I_2} y_j + z_{I_2} z_j;$$

$$v_2 = \vec{I}_2 \cdot \vec{k} = |\vec{I}_2| \cdot |\vec{k}| \cdot \cos\beta_2 = \cos\beta_2 = x_{I_2} x_k + y_{I_2} y_k + z_{I_2} z_k;$$

$$u_3 = \vec{I}_3 \cdot \vec{j} = |\vec{I}_3| \cdot |\vec{j}| \cdot \cos\alpha_3 = \cos\alpha_3 = x_{I_3} x_j + y_{I_3} y_j + z_{I_3} z_j;$$

$$v_3 = \vec{I}_3 \cdot \vec{k} = |\vec{I}_3| \cdot |\vec{k}| \cdot \cos\beta_3 = \cos\beta_3 = x_{I_3} x_k + y_{I_3} y_k + z_{I_3} z_k.$$

Как следует из приведенных соотношений, координаты *u, v* — дробные числа от -1 до +1. Напротив, координаты точки на карте затенения, представляющей собой растровое изображение, задаются целыми числами, определяющими порядковый номер пиксела в строке и столбце растра. Поэтому координаты *u, v,* представленные дробными числами, подвергаются масштабированию и округлению. Например, если карта затенения имеет размер 256×256 пикселов и систему координат с осями *u* и *v* и началом в центре карты, то координаты каждой из трех вершин контура грани на этой карте будут определяться следующим образом:

$(round\{u_1\ 128\},\ round\{v_1\ 128\});$

$(round\{u_2\ 128\},\ round\{v_2\ 128\});$

$(round\{u_3\ 128\},\ round\{v_3\ 128\}),$

где символ *round*{} означает операцию округления до ближайшего целого.

Рассчитав подобным образом координаты *u, v* трех вершин контура грани, можно наложить контур грани на карту затенения и тем самым определить освещенность любой точки в пределах этой грани (см. рис. 3.11).

3.1.2.2. Удаление скрытых поверхностей

При проецировании трехмерного объекта на двумерную плоскость экрана необходимо корректно определить, какие части объекта будут видимыми, а какие нет. Это позволит, во-первых, избежать ошибок при построении изображения, и, во-вторых, — значительно уменьшить количество граней, которые необходимо текстурировать. Этот процесс называется удалением скрытых поверхностей (HSR). Он носит комплексный характер и решается последовательно, на различных этапах 3D-конвейера. В частности, можно выделить следующие этапы выполнения HSR:

- отсечение частей изображения, не попадающих в объем отсечения *(clip volume);* объем отсечения

представляет собой параллелепипед, задающий границы трехмерной сцены;

- отбрасывание задних, невидимых граней *(culling);*
- z-сортировка;
- z-буферизация.

Два первых этапа выполняются на геометрической стадии 3D-конвейера, два последних — на стадии рендеринга.

Суть первого этапа очевидна: все, что оказывается за пределами объема отсечения, исключается из дальнейшей обработки. Эта процедура реализуется всегда.

3.1.2.2.1. Отбрасывание задних граней

Отбрасывание задних граней выполняется после тесселяции, в результате выполнения которой поверхность объекта разбивается на грани — треугольники, заданные координатами их вершин. Суть отбрасывания задних граней заключается в том, чтобы определить, какая грань будет видна при проецировании конечного изображения на плоскость экрана, а какая — нет.

Все грани поверхности объекта ориентированы в пространстве. Зная координаты вершин грани, можно рассчитать координаты нормали — вектора, направленного под прямым углом к плоскости, в которой лежит эта грань. А, зная координаты нормали, легко вычислить угол между нормалью и плоскостью экрана и тем самым определить, в какую сторону «обращена» данная грань: к наблюдателю (рис. 3.13) или в противоположную от него сторону. В последнем случае грань исключается из дальнейшей обработки. За счет отбрасывания задних граней объем операций, необходимых для выполнения последующего рендеринга, сокращается примерно в два раза.

Иногда может возникнуть ситуация, когда две грани обращены к наблюдателю, но при проецировании перекрываются. Какую же из них следует текстурировать, а какую исключить из обработки?

Для ответа на этот вопрос используется третья координата z каждого элемента поверхности объекта, характеризующая удаление этого элемента «вглубь» от экрана. Область видеопамяти, отводимая для хранения этих данных, называется *z-буфером.*

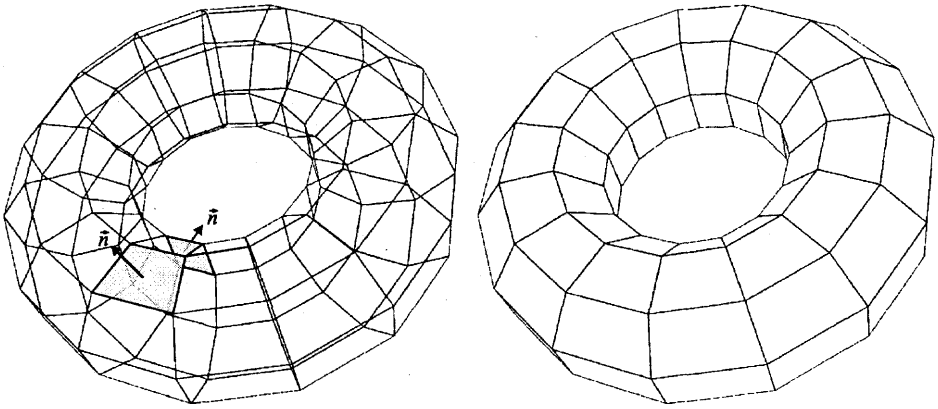


Рис. 3.13. Отбрасывание задних граней: слева — объект до удаления задних граней, справа — после удаления

Важным параметром 3D-акселератора является разрядность z-буфера: она определяет количество возможных градаций глубины трехмерной сцены. Широко распространен 16-разрядный z-буфер (2^{16}= 65536 градаций глубины). Современные модели 3D-акселераторов имеют 24- и 32-разрядный z-буфер.

Кроме разрядности z-буфера, важную роль играет форма представления чисел, описывающих координату z. При использовании целых чисел говорят об «узком» z-буфере *(narrow z-buffer),* при использовании вещественных чисел с плавающей точкой — о «широком» z-буфере *(wide z-buffer).* Для удаления невидимых наблюдателем участков используются два метода: z-сортировка и z-буферизация.

5.1.2.2.2. Z-Сортировка

Z-Сортировка (z-sorting) — наиболее простой метод, при котором целенаправленное удаление скрытых поверхностей специально не производится. Программа выполняет рендеринг всех элементов трехмерной сцены, но в порядке убывания координаты *z.* При этом происходит наложение элементов сцены, причем более близкий к зрителю элемент прорисовывается поверх более удаленного. Естественно, что видимым оказывается последний из элементов, т. е. ближний. Z-сортировка не требует хранения (запоминания) координаты *z* (z-буфер не используется), однако нерационально загружается CPU: прорисовывается весь объект, независимо от видимости его элементов. Кроме того, при прорисовке близко расположенных или пересекающихся объектов данный метод часто дает ошибочный результат (рис. 3.14, слева).

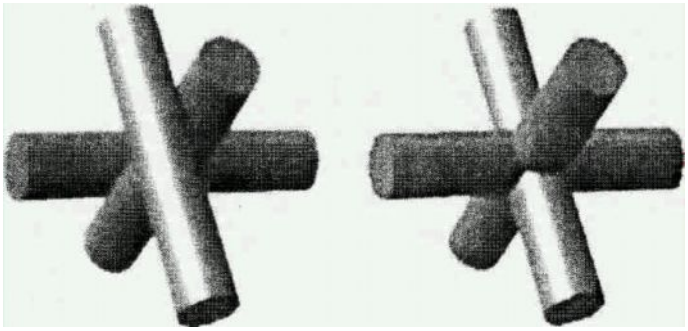


Рис. 3.14. Удаление скрытых поверхностей методами z-сортировки (слева) и z-буферизации (справа)

3.1.2.2.3. z-Буферизация

При z-буферизации выполняется анализ всех z-координат элементов объекта в процессе рендеринга. Для этого, собственно, и нужен z-буфер. Из всех точек объекта с одинаковыми координатами (x, y) проецироваться будет только одна, у которой координата *z* имеет минимальное значение. В результате будут корректно отображаться объекты, образованные пересекающимися поверхностями (рис. 3.14, справа).

3.1.2.3. Оптимизация наложения текстур

Наложение текстур — наиболее трудоемкий и сложный этап 3D-конвейера, который сопровождается появлением многочисленных дефектов на изображении. Именно на этом этапе используется большая часть разнообразного арсенала аппаратных средств 3D-акселератора, предназначенного для коррекции изображения.

Для создания реалистичной картины при рендеринге близко расположенных и удаленных частей сцены необходимо использовать различное разрешение. Большинство поклонников первых трехмерных игр хорошо помнят характерный недостаток Wolf 3D и DOOM: гладкая на расстоянии, стена или дверь при приближении искажается за счет эффекта пикселизации, превращаясь в набор цветных квадратов. Причина проста: в этих играх для рендеринга удаленных и близких поверхностей используются текстуры с одинаковым, низким разрешением.

Увеличение размеров объекта при приближении к нему сопровождается масштабированием растровой картинки текстуры. В результате проявляется хорошо известный всем эффект — пикселизация (лестничный эффект). Для устранения пикселизации можно было бы использовать текстуры с высоким разрешением, однако для их хранения требуется значительный объем памяти, к тому же они совершенно не нужны для текстурирования удаленных объектов.

Очевидно, что необходим какой-либо компромиссный подход, который, с одной стороны, обеспечил бы детальное текстурирование близко расположенных объектов, а с другой — позволил бы рационально использовать ресурсы 3D-акселератора.

3.1.2.3.1. Мип-мэппинг (MIP-mapping)

Пути решения проблемы оптимального использования текстур очевидны: при изменении расстояния до объекта (т. е. координаты z) необходимо оперативно изменять разрешение накладываемых текстур. Для этого в 3D-акселераторе используется технология, называемая *мип-мэппингом (MIP-mapping)* или *MIP-текстурированием.* Суть ее состоит в том, что в локальной памяти акселератора хранятся несколько вариантов одной и той же текстуры, но с разным разрешением, или уровнем детализации (LOD — *Level Of Detalization*), например 64×64,128×128 и т. д. Размер каждой текстуры следующего уровня в 4 раза больше размера предыдущей. Совокупность всех вариантов одной и той же текстуры называют *MIP-каскадом* (рис. 3.15).

При рендеринге ближних поверхностей используются более крупные текстуры, а при прорисовке дальних — более мелкие. Это значительно ослабляет эффект пикселизации.

Название этой технологии образовано аббревиатурой MIP — *Multum In Parvo* — «много в одном», что отражает основную идею описанных выше действий. Очевидный недостаток данной технологии — резкое увеличение требуемого объема памяти акселератора, необходимой для хранения текстур. Для того чтобы хранить текстуры не в локальной памяти 3D- акселератора, а в оперативной памяти PC и при необходимости быстро их подгружать, используется локальная шина с высокой пропускной способностью — шина AGP. Наличие такой шины позволяет организовать хранение и обработку текстур не в ограниченной по объему памяти акселератора, а в оперативной памяти компьютера. Более подробно особенности шины AGP будут описаны в разд. 3.2.1.

Текстуры можно хранить не только в оригинальном, но и в сжатом виде. Лидером в этой области является компания S3, разработавшая технологию сжатия текстур S3TC, — *S3 Texture Compression.* Эту технологию лицензировали многие крупнейшие фирмы, включая Microsoft.

Довольно часто прикладная программа вместо полного MIP-каскада содержит только один вариант текстуры. В этом случае 3D-акселератор может самостоятельно создать MIP-каскад. Имеющейся текстуре

присваивается максимальный уровень детализации, а затем выполняется ее кратное уменьшение. Масштабирование выполняется один раз, в начале работы, и в дальнейшем используется обычная техника мип-мэппинга. Такой способ создания MIP-каскада получил название *авто мип-мэппинга (Auto MIP-mapping).*

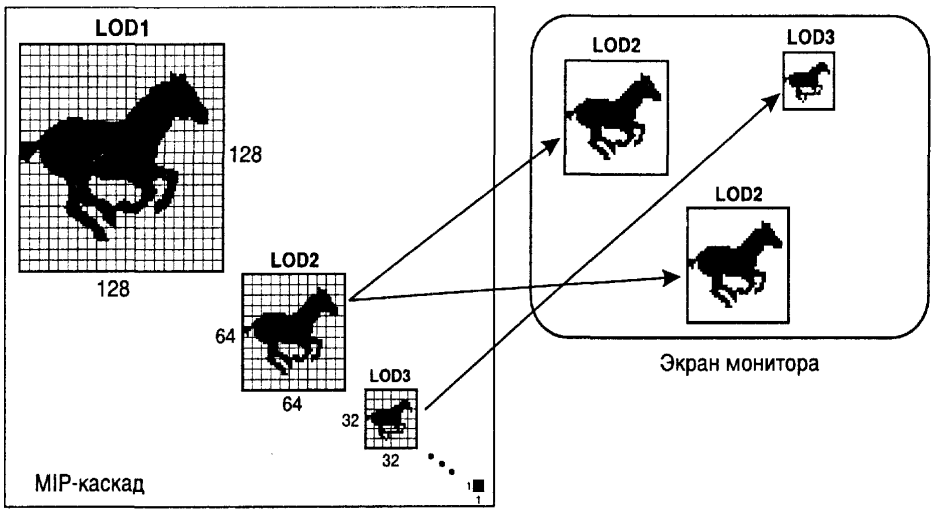


Рис. 3.15. MIP-каскад из нескольких текстур

Рассмотренная выше технология является простейшей и называется *nearest MIP-mapping*— использование ближайшей, т. е. наиболее подходящей по разрешению текстуры. Поскольку текстуры меняются дискретно, на изображении появляются искажения. В частности, возникает целый ряд специфических дефектов:

- *полосы MIP-текстурирования (Banding of MIP-mapping)* — хорошо различимые границы между текстурами с разным разрешением;
- «шум текстуры»; при движении объекта его поверхность приобретает изменяющийся рисунок;
- потеря четкости (размывание) мелких деталей на удаленных текстурах и муар — инородный рисунок на изображении текстуры;
- нарушение перспективы на рисунке текстуры;
- нестыковка текстур на краях граней, приводящая к появлению светлых или темных участков на поверхности объекта;
- рваные края объектов (лестничный эффект); дефект особенно заметен на наклонных линиях (обычно его называют *алиасингом — aliasing);*
- искажение глубины отдельных элементов рисунка — *z-алиасинг (z-aliasing).*

Большинство этих дефектов обусловлено дискретной (скачкообразной) сменой разрешения текстуры при плавном изменении координаты z. По этой причине nearest MIP-mapping в настоящее время практически не используется. Для устранения дефектов MIP-текстурирования используется *фильтрация,* которая сводится к той или иной форме интерполяции значения тексела по текселам соседних текстур MIP-каскада.

3.1.2.3.2. Фильтрация

Суть фильтрации состоит в том, что при вычислении значения каждого тексела ближайшая текстура в MIP-каскаде масштабируется с учетом расстояния от нее до текстурируемого участка поверхности объекта, называемого *LOD-bias* — сдвиг по уровню детализации (рис. 3.16). Иными словами, фильтрация сводится к изменению размеров текстуры с использованием различных методов интерполяции, при этом коэффициент масштабирования определяется на основе LOD-bias. В зависимости от используемых методов интерполяции различают следующие виды фильтрации:

- *линейную (linear filtering, LF);* для интерполяции тексела используется значение тексела в текстуре ближайшего уровня (LOD);
- *билинейную (bi-linear filtering, BLF},* каждый тексел вычисляется как взвешенная сумма значений четырех соседних (слева и справа, сверху и снизу) текселов; это наиболее распространенный метод, однако в результате его применения остаются полосы MIP-текстурирования и изображение немного размывается (рис. 3.17);
- *трилинейную (tri-linear filtering, TLF);* каждый тексел текстуры вычисляется как усредненный результат билинейной фильтрации по двум соседним уровням детализации (используются две текстуры); это наиболее точный из используемых методов, при котором полосы MIP-текстурирования не образуются;
- *анизотропную;* используется при прорисовке сильно наклоненных к наблюдателю граней, когда

текселы перестают быть квадратными (рис. 3.18). В соответствии с этим методом, тексел описывается эллипсом и интерполяция выполняется по большему количеству текселов (до 32). Анизотропная фильтрация значительно сложнее трилинейной и используется в основном в высококачественных игровых и профессиональных 3D-акселераторах.

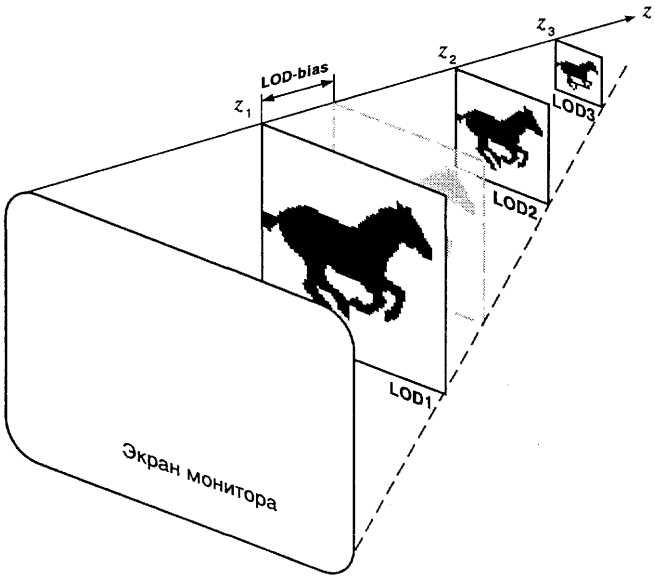


Рис. 3.16. Фильтрация на основе LOD-bias

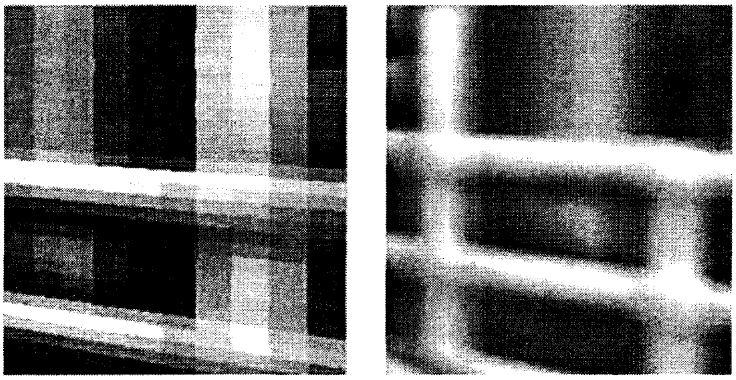


Рис. 3.17. Пример использования билинейной фильтрации: слева — билинейная фильтрация выключена, справа — включена

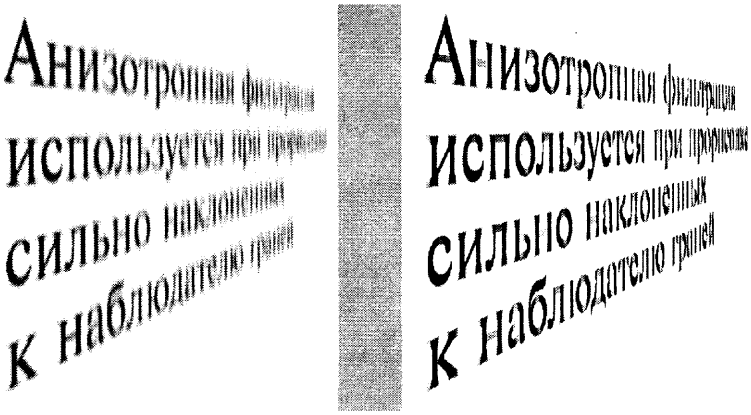


Рис. 3.18. Пример реализации трилинейной (слева) и анизотропной (справа) фильтрации

В игровых 3D-акселераторах в основном используется билинейная и трилинейная фильтрация. Трилинейная фильтрация позволяет устранить практически все перечисленные выше дефекты, за исключением алиасинга. Однако ее аппаратная реализация требует наличия в 3D-акселераторе двух блоков (процессоров) по обработке текстур со своими буферами памяти, поскольку данный алгоритм подразумевает одновременную обработку двух различных текстур MIP-каскада. При наличии только одного блока возможна программная реализация (эмуляция) трилинейной фильтрации, однако при этом значительно снижается производительность видеосистемы.

3.1.2.3.3. Мультитекстурирование

Важным свойством 3D-акселератора является его способность ускорять процесс наложения текстур. Для этого используется технология *мультитекстурирования.* Как правило, результирующее значение каждого пиксела кадрового буфера становится известным только после наложения на него нескольких текстур (рис. 3.19), т. е. текстурирование должно выполняться несколько раз, т. е. в несколько этапов.

Для того чтобы выполнить текстурирование за один этап, в состав 3D-акселератора входит не один, а два блока обработки текстур, работающих параллельно. Наложив на грань одну текстуру, первый блок передает результат работы второму блоку, который накладывает на нее другую текстуру. Первый блок в это время уже занят обработкой следующей грани и т. д. Другими словами, текстурирование выполняется конвейерным способом.

Еще раз подчеркнем, что для аппаратной поддержки мультитекстурирования, так же, как и трилинейной фильтрации, 3D-акселератор должен иметь два блока обработки и наложения текстур.

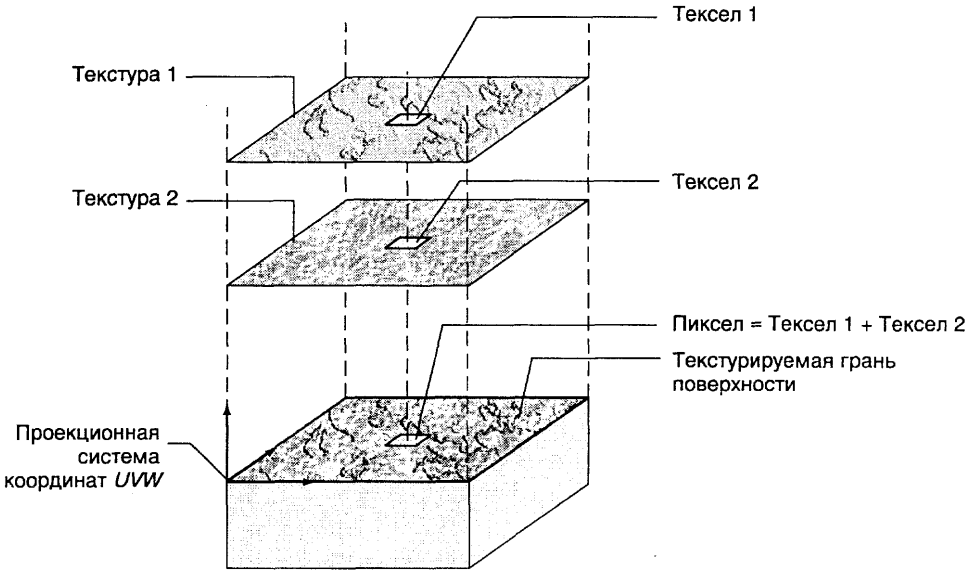


Рис. 3.19. Механизм мультитекстурирования

О наличии этих блоков свидетельствует аббревиатура TNT, входящая в название наиболее популярных в настоящее время игровых 3D-акселераторов, например Riva TNT(2) *(TwiN Texel* — сдвоенный блок обработки текселов).

3.1.2.3.4. Текстурирование с коррекцией перспективы

Еще одной важной задачей, решаемой при наложении текстур, является коррекция перспективы. Поскольку различные грани, образующие поверхность объекта, наклонены под разными углами к линии визирования, при наложении текстуры необходимо выполнить поворот и скос растра исходной, квадратной текстуры. При этом возникают характерные искажения, нарушающие ориентацию рисунка текстуры на поверхности объекта и приводящие к искажению зрительной перспективы (рис. 3.20). Для коррекции этих искажений акселератор должен аппаратно выполнять операцию *Perspective Corrected Texture Mapping*— наложение текстуры с коррекцией перспективы.

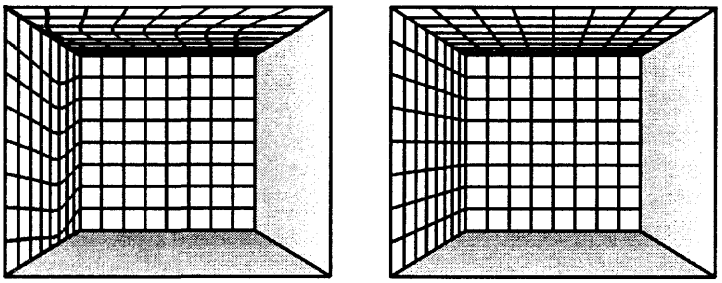


Рис. 3.20. Пример текстурирования с коррекцией перспективы (слева) и без нее (справа)

3.1.2.3.5. Моделирование рельефа при наложении текстур

Строго говоря, рельеф объекта как часть его формы формируется на первых этапах геометрической стадии. Однако это укрупненный, *грубый рельеф,* который передает сравнительно крупные детали поверхности. Предусмотреть на геометрической стадии задание тонкой структуры рельефа не

представляется возможным, так как это значительно усложнит геометрическую модель и сведет на нет выгоду от использования векторной графики. Между тем реалистичность восприятия объекта во многом зависит именно от *тонкого рельефа,* передающего мелкую структуру поверхности объекта и позволяющего создать ощущение шершавости или, напротив, гладкости, зеркальности поверхности.

Оказывается, что эту задачу можно решить на этапе наложения текстур, изменяя соответствующим образом их рисунок. Эта технология получила название *bump-mapping—рельефное текстурирование.* Согласно определению Джима Блинна, автора данной технологии, рельефное текстурирование — это метод получения визуальных выпуклостей и вогнутостей на объекте, находящемся в трехмерном пространстве, без изменения геометрической формы самого объекта.

Основная идея рельефного текстурирования очевидна: наличие неровностей на поверхности объекта приводит к образованию участков с различной освещенностью. Вопрос только в том, каким образом задать правило, по которому будет изменяться освещенность этих участков текстуры.

Заметим, что технология рельефного текстурирования сильно напоминает рассмотренные выше методы затенения, в соответствии с которыми исходные данные рассчитываются на геометрической стадии конвейера для плоских граней. Однако рельефное текстурирование выполняется с растровой графикой. Существует несколько разновидностей рельефного текстурирования:

- текстурирование методом *вдавливания,* или чеканки *(Embossing Bump Mapping);*
- текстурирование с использованием *карт окружающей среды (Environment Map Bump Mapping);*
- текстурирование с использованием *программно-изменяемых рельефных карт (Procedural Texturing).*

3.1.2.3.5.1. Текстурирование методом вдавливания

Наиболее простым является метод вдавливания. Предположим, что необходимо получить изображение рельефной поверхности (рис. 3.21, слева). Для ее описания используется так называемая *рельефная карта,* или *карта высот.* Это растровая картинка, яркость пикселов которой прямо пропорциональна высоте участка поверхности над некоторым нулевым уровнем, соответствующим ровной поверхности (рис. 5.21, справа).

Заметим, что подобным образом обозначаются высоты (глубины) на географических картах: чем глубже участок океана, тем более темным цветом он отображается на карте.

Обратите внимание на рис. 3.21 слева. Ощущение рельефа создается за счет изменения освещенности только тех участков поверхности, которые расположены на границе изменения высот, т. е. на изгибах поверхности. Если изгиб обращен в сторону света, он делается более светлым, и наоборот, т. е. поверхность объекта раскрашивается в соответствии с классическим законом освещенности.

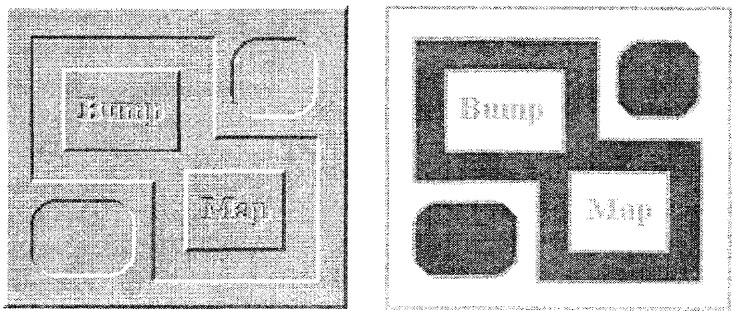


Рис. 3.21. Рельефная поверхность (слева) и рельефная карта (справа)

Для определения освещенности, согласно данному закону, необходимо знать два направления: на источник света и самой нормали к освещаемой поверхности. У плоской поверхности нормаль имеет постоянную ориентацию, а у рельефной изменяется, причем изменение угла наклона нормали происходит как раз в местах изгибов поверхности. Рельефная карта позволяет рассчитать направление и угол, на который необходимо «довернуть» нормаль в каждой точке изгиба, чтобы в дальнейшем корректно рассчитать освещенность этой точки. Для этого используется градиентный метод:

$$du = pixel\,(u - 1,\ v) - pixel\,(u + 1,\ v);$$

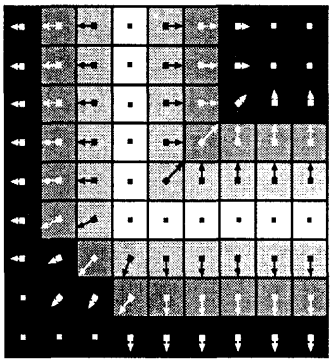
$$dv = pixel\,(u,\ v - 1) - pixel\,(u,\ v + 1),$$

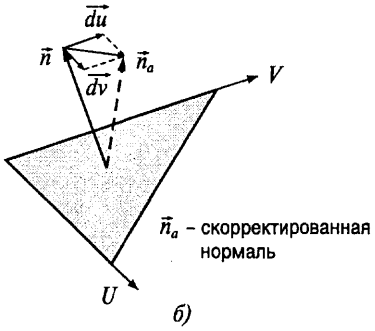
где $u,\ v$ — координаты пиксела, для которого рассчитывается освещенность;

$du,\ dv$ — поправки к координатам нормали.

В результате этих вычислений карта высот заменяется *картой градиентов* (рис, 3.22, *а),* элементами которой являются векторы с координатами *(du, dv),* которые необходимо сложить с вектором нормали (рис. 3.22, *б).*

После определения ориентации нормали в каждой точке поверхности выполняется расчет ее освещенности по методу Фонга. Для ускорения этого процесса могут использоваться заранее подготовленные карты освещенности. В результате получается рельефная поверхность, изображенная на рис. 3.21.





а) б)

Рис. 3.22. Фрагмент карты градиентов *(а);* значения градиентов определяются длиной маленьких стрелок) и вектор коррекции нормали *(б)*

Наложение текстур при рельефном текстурировании по методу вдавливания может выполняться и с использованием технологии *альфа-смешения* (рис. 3.23).

Отметим, что аппаратная поддержка альфа-смешения реализована практически в любом, даже недорогом 3D-акселераторе. Это значительно ускоряет выполнение процесса наложения текстур методом вдавливания. Рельефное текстурирование этим методом имеет и ряд недостатков:

Рис. 3.23. Наложение текстур путем альфа-смешения

- используется только монохромный источник света;
- качество изображения сильно зависит от величины сдвига, вычисляемого по карте высот;
- метод не позволяет реализовать множество реальных эффектов, особенно эффект отражения от поверхности объекта окружающих предметов.

3.1.2.3.5.2. Текстурирование с использованием карт окружающей среды

Для устранения недостатков метода вдавливания применяется текстурирование с использованием *карт окружающей среды — Environment Map Bump Mapping (EMBM).* Пионерами в разработке данного метода являются фирмы Matrox (3D-акселераторы G200 и G400) и 3Dlabs (3D-акселератор Permedia3). В отличие от метода вдавливания, при текстурировании с использованием карт окружающей среды для моделирования рельефа поверхности используются не одна, а две карты:

- карта изменения высот (карта уклонов), рассчитываемая на основе карты высот градиентным методом;
- карта окружающей среды (3D-сцены).

На основе карты уклонов корректируются координаты пикселов на карте окружающей среды. Обе эти карты пересчитываются в текстуры соответствующего разрешения. На заключительном этапе стадии рендеринга скорректированная карта окружающей среды накладывается на основную текстуру, определяющую рисунок поверхности объекта.

Сравните два изображения на рис. 3.24, полученные с использованием каждого из методов рельефного текстурирования. Преимущества метода с использованием карт окружающей среды очевидны.

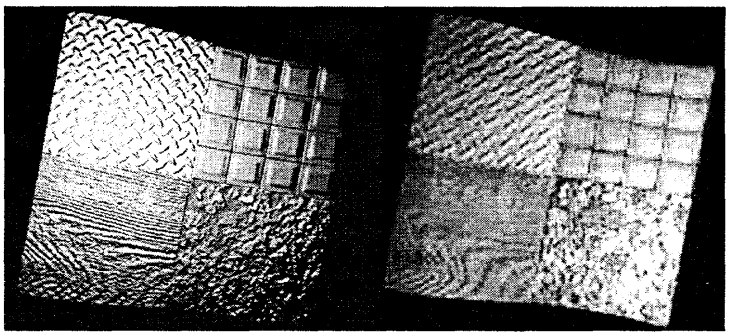


Рис. 3.24. Рельефное текстурирование с использованием карт окружающей среды (слева) и по методу вдавливания (справа)

3.1.2.3.5.3. Динамическое изменение рельефной карты

Текстурирование с использованием карт окружающей среды совместно с *динамическим изменением рельефной корты (Procedural Texturing)* позволяет смоделировать удивительные по своей реалистичности визуальные эффекты:

- отражения от поверхности, особенно водной (рис. 3.25), окружающих объектов; имитация «живых» волн на поверхности воды;
- искажения предметов в водной среде, кривые зеркала и линзы (рис. 3.26);

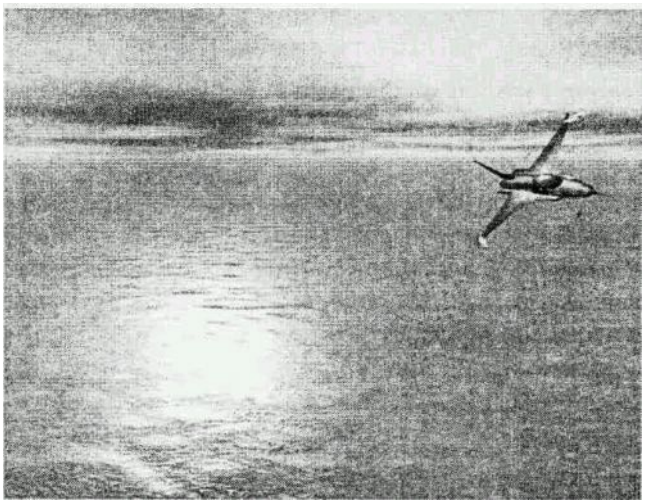


Рис. 3.25. Пример моделирования водной поверхности

Рис. 3.26. Пример моделирования отражения человека в кривом зеркале: исходная текстура окружающей среды (слева) и поверхность зеркального мяча, в котором отражена исходная текстура (справа)

- изменение размеров или исчезновение отверстий на поверхности (например, знаменитые исчезающие пулевые отверстия в фильме «Терминатор-2»);
- искажение изображения (дрожание и размывание) под действием поднимающегося над землей теплого воздуха и др.

3.1.2.4. Управление прозрачностью и цветовой насыщенностью текстур

В реальной жизни изменяется не только яркость объектов, но и их прозрачность и цветовая насыщенность. Для того чтобы учесть эти изменения при движении объектов (например, изменение цветовой насыщенности удаляющегося объекта) и особенно при их наложении, необходимо управлять насыщенностью или прозрачностью каждого пиксела. Для этой цели используется альфа-смешение.

3.1.2.4.1. Альфа-смешение

Технология, позволяющая управлять насыщенностью цвета и прозрачностью каждого пиксела при наложении объектов, получила название *альфа-смешение* или *alpha pixel blending*— смешение цветов пикселов с учетом коэффициента α.

Суть альфа-смешения заключается в следующем. Цвет пикселов, участвующих в наложении, определяется не тремя, как обычно, а четырьмя компонентами: R, G, B и α, где α — весовой коэффициент,

учитывающий *степень прозрачности объекта (Object Transparency).* Значению $\alpha = 1$ соответствует пиксел с максимальной прозрачностью, а значению $\alpha = 0$ — непрозрачный пиксел. Такой способ задания цвета пиксела позволяет корректно выполнять смешение цветов, необходимость в котором возникает при наложении друг на друга полупрозрачных объектов.

Обычно при наложении объектов обрабатываются два пиксела: *исходный (source pixel)* и *целевой (destination pixel),* который уже сформирован в кадровом буфере. После смешения результирующее значение присваивается целевому пикселу.

Для реализации смешения цветов каждый пиксел, хранящийся в кадровом буфере, должен иметь свой атрибут α. Совокупность этих значений принято называть альфа-буфером.

Технология альфа-смешения используется при выполнении многих функций 3D-акселерации. В первую очередь, это наложение текстур. Кроме того, путем альфа-смешения создается множество таких эффектов, как антиалиасинг, создание теней и зеркальных отражений, полупрозрачности, тумана и др.

Задавая значение α в диапазоне от 0 до 1 для исходного и хранящегося в кадровом буфере пикселов, можно выполнить суммирование цветов. Например, если исходный пиксел представлен в виде R_1, G_1, B_1, α_1 , а содержащийся в кадровом буфере — R_2, G_2, B_2, α_2, то цвет результирующего пиксела может быть определен в результате выполнения следующих операций:

- добавление прозрачности исходному пикселу или накладываемой текстуре (эту операцию называют также *Source Alpha Pixel Blending*):
  $R = \alpha_1 R_1 + R_2(1 - \alpha_1)$, $G = \alpha_1 G_1 + G_2(1 - \alpha_1)$, $B = \alpha_1 B_1 + B_2(1 - \alpha_1)$;
- повышение прозрачности цвета исходного пиксела и подчеркивание цвета второго, результирующего пиксела путем уменьшения значения α_1;
- простое суммирование цветов *(Add Pixel Blending):*
  $R = R_1 + R_2$, $G = G_1 + G_2$, $B = B_1 + B_2$;
- взвешенное суммирование цветов *(Weighted Add Pixel Blending):*
  $R = \alpha_1 R_1 + \alpha_2 R_2$, $G = \alpha_1 G_1 + \alpha_2 G_2$, $B = \alpha_1 B_1 + \alpha_2 B_2$;
- модуляция (умножение) цветов *(Modulate Pixel Blending):*
  $R = R_1 R_2$, $G = G_1 G_2$, $B = B_1 B_2$.

5.1.2.4.2. Затуманивание

Плавное уменьшение коэффициента α, до нуля, по мере удаления объекта, позволяет создать эффект, называемый *затуманиванием (Fogging):* очертания удаляющегося объекта размываются, постепенно сливаясь с фоном (рис. 3.27).

Рис. 3.27. Пример использования техники затуманивания

В зависимости от вида функции $\alpha(z)$ и правила выбора значения α для конкретного пиксела различают несколько видов затуманивания. Если значение α линейно уменьшается с увеличением расстояния до объекта (координаты z), то такое затуманивание называют *линейным (Linear Fog).* Если же значение α уменьшается по экспоненциальному (показательному) закону

$\alpha(z) = e^{-kz} = \exp(-kz)$, где $k = \text{const} > 0$,

то говорят об *экспоненциальном тумане (Exponential Fog).*

Туман бывает также *полигонным (Vertex Fog)* и *табличным (Table Fog).* Для создания полигонного тумана рассчитывается одно значение коэффициента α для всей грани, заданной координатами ее вершин. Во втором случае используется специальная таблица, содержащая значения α для каждого пиксела. Использование эффекта «табличный туман» позволяет создать более реалистичную картину чем при использовании полигонного тумана. Однако для реализации этого эффекта требуется мощный 3D-акселератор.

3.1.2.4.3. Морфинг уровней детализации

Для того чтобы приближающиеся издалека объекты возникали на экране не резко, а плавно, появляясь «ниоткуда», применяется так называемый *морфинг уровней детализации (LOD Morphing).* Суть этого метода заключается в том, что по мере приближения объекта уменьшается степень прозрачности его текстур.

3.1.2.4.4. Управление прозрачностью с использованием маски

При формировании изображения перекрывающихся объектов эффект полупрозрачности достигается не только путем альфа-смешения, но и за счет использования приема, называемого *Chroma Keying* или *Texture Color Keying*- настройка цвета текстуры. В этом случае степень прозрачности задается специальной маской, определяющей, какие из элементов объекта должны быть прозрачными а какие - нет. Для обозначения полупрозрачных и просвечивающихся объектов используются термины *Transparency* и *Translucency.* Пример эффекта полупрозрачности приведен на рис. 3.28.

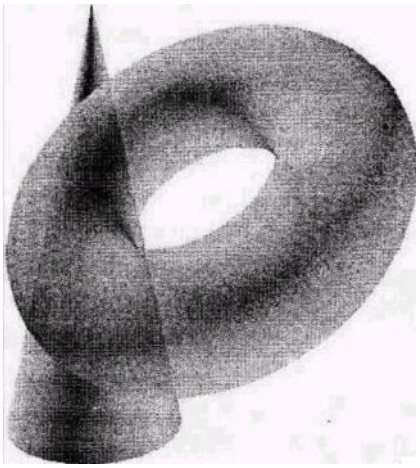


Рис. 3.28. Полупрозрачный тор

3.1.2.4.5. Зеркальные блики

Путем моделирования световых бликов и отражений можно еще больше повысить реалистичность изображения. Эта функция 3D-акселератора (имеется в виду аппаратная поддержка) обозначается *Specular Highlights* - зеркальные блики. В процессе наложения текстур цвета̀ их элементов изменяются таким образом что создается эффект отражения объекта от гладкой поверхности.

3.1.2.5. Способы хранения текстур

От способа хранения текстур зависит необходимый размер локальной памяти акселератора. Самым простым способом является хранение текстур в формате RGB, когда отводимые для хранения цвета каждого тексела разряды делятся между красной, зеленой и синей составляющими. Например, типичным для большинства 3D-акселераторов является 16-битный режим High Color, в котором используется формат RGB 5:6:5, т. е. 5 разрядов для кодирования интенсивности красной компоненты, 6 разрядов — для зеленой и оставшиеся 5 — для синей компоненты. В режиме True Color используется формат RGB 8:8:8.

При необходимости смоделировать эффекты полупрозрачности и затуманивания к RGB-компонентам добавляют четвертую — параметр α. В этом случае говорят о формате RGBα. В частности, в режиме High Color будет использован формат RGBα 5:5:5:1, в котором для учета свойств прозрачности отведен один бит (реализуются так называемые *простые эффекты прозрачности).* Наибольшими возможностями по управлению цветом обладает 32-битный режим True Color, в котором используется формат RGBα 8:8:8:8.

Формат RGB является единственно возможным для хранения данных в кадровом буфере, так как на трехканальный RAMDAC они подаются именно в этом формате. Поэтому при хранении текстур в формате RGB нет необходимости в каких-либо преобразованиях данных и обеспечивается наибольшая скорость работы. Однако для хранения текстур в этом формате требуется значительный объем локальной памяти.

3.1.2.5.1. Режим с палитрой

Для снижения требований к объему локальной памяти 3D-акселератора разработчики аппаратных средств предложили использовать не всю доступную палитру цветов, а только ту ее часть, которая используется в конкретной текстуре. Размер палитры текстуры зависит от ее назначения. Так, для текстурирования простейших поверхностей используются 2-, 4- и 16-цветные текстуры, а для прорисовки более сложных объектов — 256-цветные и т. д. Цвет каждого тексела таких текстур определяется по специальной таблице — *CLUT* (*Color Look-Up Table*— таблица поиска цветов). Таблицы для 2-, 4-, 16-и 256-цветных текстур называются CLUT-1,CLUT-2, CLUT-4 и CLUT-8 соответственно. Эти таблицы задают соответствие между условным порядковым номером цвета тексела, представленным n-разрядным двоичным

кодом *(n* = 1, 2, 4, 8), и фактическим RGB-кодом цвета в 16- или 24-битном формате. При рендеринге и заполнении кадрового буфера, естественно, используется формат RGB.

Такой режим косвенного задания цвета называется *режимом с палитрой (palletized),* а текстуры — *палетизированными* или *оптимизированными (palletised textures).*

3.1.2.5.2. Технология сжатия текстур S3TC

Технология сжатия текстур S3TC основана на хорошо известном свойстве человеческого зрения: мелкие детали на изображении различаются только по градациям яркости, т. е. кажутся черно-белыми (это свойство широко используется в телевидении). Субъективно, разрешающая способность изображения оценивается наличием мелких деталей и резкостью границ отдельных участков и зависит от ширины полосы частот сигнала яркости. Чем выше верхняя граничная частота сигнала яркости, тем больше мелких деталей на изображении. Иными словами, качество изображения в большей степени определяется разрешением сигнала яркости и в меньшей — разрешением сигналов цветности. Следовательно, цвет тексела может изменяться значительно медленнее, чем его яркость.

Таким образом, использование одинакового количества разрядов для кодирования цвета каждого те ксела текстуры (например, RGB 5:6:5) является избыточным, поскольку цвета соседних текселов различаются не сильно (коды цветов различаются лишь значениями младших разрядов). Следовательно, в пределах небольшого участка поверхности объекта можно определить один базовый, превалирующий цвет, а для каждого тексела этого участка задавать лишь отличия от этого цвета. Очевидно, что в этом случае для кодирования цвета тексела понадобится значительно меньшее количество разрядов.

Итак, сжатие текстур по технология S3TC выполняется следующим образом (рис. 3.29). Исходная текстура разбивается на блоки размером 4×4 тексела. В режиме High Color для их хранения в локальной памяти выделено 32 + 32 = 64 бита. Первые 32 бита отводятся для хранения так называемой *матрицы цветов* (матрицы выборки). Она содержит полный код основного, базового цвета (16 разрядов в формате RGB 5:6:5) и поправки для определения еще трех, дополнительных цветов (остальные 16 разрядов). Один из этих дополнительных цветов используется для моделирования простого эффекта прозрачности. При сжатии каждому из 16 текселов исходного блока 4×4 присваивается один из этих четырех цветов (основной, два дополнительных цвета и прозрачность). Поскольку для кодирования четырех цветов достаточно двух разрядов, то сжатый блок займет оставшиеся 16 × 2 = 32 разряда.

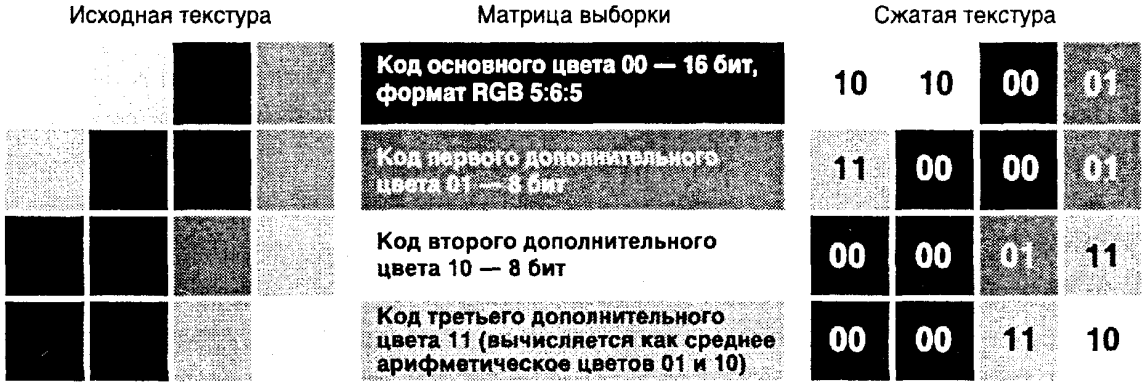


Рис. 3.29. Сжатие текстур по технологии S3TC

Каков же эффект от сжатия? Вместо 16×16 = 256 разрядов, которые понадобились бы для хранения текстуры в формате RGB 5:6:5, при использовании S3TC достаточно всего 64 разряда, т. е. в 4 раза меньше. При этом возникает закономерный вопрос: сколько же цветов будет отображаться? Ведь при двухразрядном кодировании цвета тексела вместо 65 536 оттенков цветов получается всего 4! Действительно, всего 4 разных цвета, но только в пределах одного блока текстуры размером 4×4 тексела. Другой блок будет иметь иной базовый цвет, а поскольку он кодируется 16 разрядами, то может иметь любое из 65 536 значений.

Очевидно, что в результате применения этой технологии снижается четкость изображения, однако это снижение незначительное и практически не влияет на качество изображения. Общепризнанным считается мнение о том, что, по сравнению со всеми используемыми в настоящее время методами компрессии (например, JPEG), технология S3TC обеспечивает наилучшее качество изображения при достаточно высоком коэффициенте сжатия.

Если вместо 16-битного цвета исходной текстуры использовать 32-битный (режим True Color), степень сжатия текстур по технологии S3TC можно увеличить почти в 8 раз.

Помимо высокой компрессии и хорошего качества, технология S3TC имеет еще одно важное преимущество: очень простой алгоритм декомпрессии. Для восстановления изображения необходимо рассчитать три 16-битных кода дополнительных цветов (причем основной цвет уже задан в этом формате) и присвоить каждому из 16 текселов блока один из четырех возможных цветов. Это реализуется путем выполнения элементарных арифметических и логических операций, поэтому декомпрессия может

осуществляться либо аппаратно, на простейшем устройстве, либо программно, «налету» (эта функция включена в DirectX 6.0).

Справедливости ради отметим, что несмотря на широкие перспективы, открывающиеся с появлением технологии S3TC, она применяется лишь в некоторых графических приложениях, не говоря уже об играх. Поэтому все преимущества от использования S3TC пока можно ощутить лишь при проведении специальных тестов.

Для хранения текстур могут использоваться и другие методы сжатия, со своими форматами (например, сжатие по методу JPEG, формат YUV и др.). Многие фирмы-производители разрабатывают для этого собственные оригинальные технологии (в частности, фирмы ATI, Matrox, S3 и др.).

3.1.2.6. Антиалиасинг

Как уже отмечалось, в процессе MIP-текстурирования на изображении появляются дефекты, называемые алиасингом. Основными проявлениями алиасинга являются лестничный эффект, зазубренность наклонных линий (как правило, на краях объекта), инородный рисунок (муар) и др.

Для устранения этих дефектов выполняется коррекция изображения — *антиалиасинг (anti-aliasing),* или сглаживание. Различают *краевой (edge anti-aliasing)* и *полный (FSAA — Full-Screen Anti-Aliasing)* антиалиасинг. В первых моделях игровых 3D-акселераторов использовался только краевой антиалиасинг. Однако в настоящее время практически все современные 3D-акселераторы поддерживают полный антиалиасинг.

3.1.2.6.1. Краевой антиалиасинг

Краевой антиалиасинг заключается в усреднении цвета пикселов на краях грани (ребрах) на основе взвешенного суммирования цветов прилегающих граней. При этом используется техника усреднения по площади (рис. 3.30). Считается, что каждая точка на краю грани имеет фиксированную ненулевую площадь. В зависимости от того, какую часть этой площади перекрывают прилегающие грани, определяются весовые коэффициенты, которые прибавляются к коду цвета краевого пиксела.

Результат применения краевого антиалиасинга приведен на рис. 3.31.

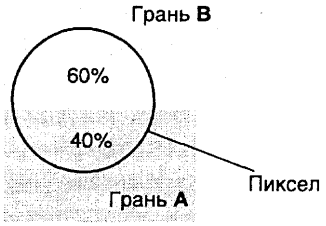


Рис. 3.30. Правило площадей, используемое при краевом антиалиасинге

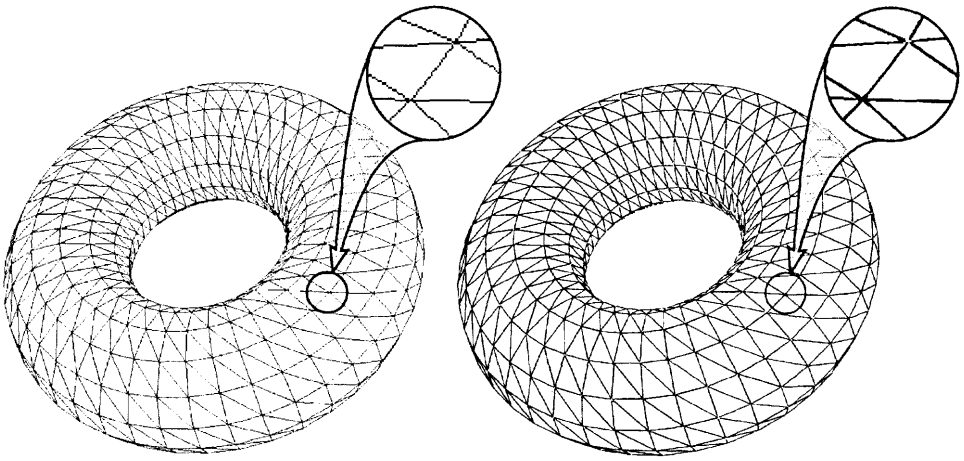


Рис. 3.31. Пример применения краевого антиалиасинга: слева — антиалиасинг выключен, справа — включен

Реализация краевого антиалиасинга достаточно проста, однако в результате его применения между смежными гранями появляется небольшой просвет и внутренние ребра объекта окрашиваются в цвет фона. Этот дефект называется *bleeding — «кровотечение».*

3.1.2.6.2. Полный антиалиасинг

Для полного устранения всех дефектов алиасинга используется *полный антиалиасинг,* или *субпиксельный антиалиасинг.* Основная идея данного метода состоит в следующем. Каждому пикселу

итогового изображения ставится в соответствие совокупность *виртуальных субпикселов,* при этом так называемое *виртуальное разрешение* изображения оказывается выше. Антиалиасингу подвергаются именно субпикселы. После выполнения коррекции, когда цвета всех субпикселов определены, субпикселы объединяются в пикселы путем интерполяции, в результате чего восстанавливается исходное разрешение.

Субпикселы получают методом *суперсэмплинга,* или *мультисэмплинга* (от англ. *sample* — отсчет).

В первом случае субпикселы получаются на стадии рендеринга путем растрирования изображения с заведомо более высоким, виртуальным разрешением, чем требуется. По окончании стадии рендеринга выполняется интерполяция, в результате которой группы субпикселов объединяются в пикселы изображения с итоговым разрешением.

Во втором случае субпикселы формируются по окончании рендеринга путем разбиения исходного пиксела на части. Характер разбиения пиксела определяется маской, задающей пространственное положение субпикселов.

Высокая плотность расположения субпикселов позволяет корректно определить цвет каждого субпиксела на основе его принадлежности к той или иной грани поверхности объекта (рис. 3.32). Значение итогового пиксела изображения получается в результате пространственного усреднения значений входящих в него субпикселов. В результате на границе между гранями будут отсутствовать субпикселы, имеющие цвет фона, т. е. негативный эффект bleeding наблюдаться не будет.

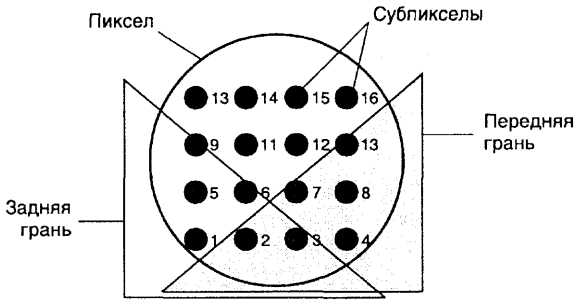


Рис. 3.32. Разбиение пиксела на субпикселы по методу мультисэмплинга

Применение субпиксельного антиалиасинга позволяет добиться хорошего результата на достаточно сложных изображениях, содержащих множество резких переходов цвета и яркости (рис. 3.33)

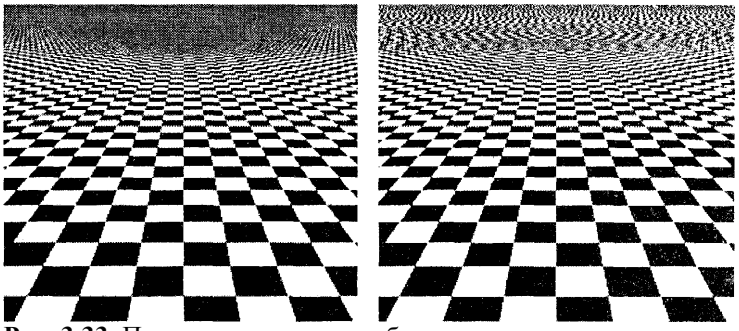


Рис. 3.33. Пример применения субпиксельного антиалиасинга: слева — антиалиасинг включен, справа — выключен

При работе с высоким виртуальным разрешением резко увеличивается объем необходимых вычислительных операций. Кратность увеличения объема этих операций примерно равна количеству субпикселов в пикселе. Кроме того, типового размера кадрового буфера недостаточно для выполнения рендеринга всего объекта с виртуальным разрешением. Поэтому для реализации субпиксельного антиалиасинга изменяют архитектуру локальной памяти 3D-акселератора. Используется так называемая *тайловая архитектура,* в соответствии с которой акселератор не сразу конструирует весь объект в кадровом буфере, а последовательно прорисовывает его фрагменты — *тайлы (tile).* Примером акселераторов с тайловой архитектурой памяти являются WARP 5 фирмы Oak Technology (см. разд. 3.3.5) и PVRSG5 фирмы PowerVR.

3.1.2.7. Интерполяция недостающих цветов

Если текущий режим работы видеоадаптера не позволяет отобразить требуемое количество оттенков цветов (имеет недостаточный объем видеопамяти), используется метод интерполяция цветов, или *Color Dithering (дизеринг),* заключающийся в получении недостающих цветов за счет смешения имеющихся. Этот метод широко распространен и используется не только в трехмерной графике, но и во многих программах обработки растровых изображений.

Суть метода подобна идее, положенной в основу работы цветного монитора: для создания цвета

пиксела экрана используется триада зерен люминофора основных цветов. При дизеринге цвета соседних пикселов изображения выбираются из доступной палитры таким образом, чтобы в результате их смешения создавалась иллюзия появления нового цвета, которого нет в исходной палитре. Очевидно, что при этом немного ухудшается разрешение изображения.

Иногда, в литературе можно встретить ошибочный перевод термина Color Dithering как «растрирование» или «растеризация». Эти термины означают процесс перевода векторных изображений в растровые, но никак не интерполяцию цветов.

Таковы основные приемы и методы, реализуемые в 3D-ускорителях для создания и коррекции трехмерных изображений. В некоторых изделиях могут использоваться и другие, более сложные и изощренные технологии, однако их детальное рассмотрение выходит за рамки данных материалов.

Контрольные вопросы.

1. Зачем нужна коррекция изображений и какую роль играет в ней 3D-ускорители?
2. Сформулируйте закон освещенности.
3. Какие методы затенения реализуются в современных игровых 3D-ускорителях?
4. Охарактеризуйте плоское (простое) затенение.
5. Охарактеризуйте метод Гуро и рамки его применимости.
6. Охарактеризуйте метод Фонга; почему он редко применяется в реальном масштабе времени?
7. Как и для чего применяется карта затенения?
8. Как определяются координаты вершин контура грани на карте затенения?
9. Что такое HSR, и какие этапы его выполнения можно выделить?
10. Какие этапы HSR выполняются на геометрической стадии 3D-конвейера, а какие на стадии рендеринга?
11. По какому критерию ведется отбрасывание задних граней и после какого процесса?
12. Как отбрасываются перекрывающиеся грани?
13. Что понимается под глубиной z-буфера и каковы ее значения, применяемые в 3D-акселераторах?
14. Что понимается под "широким" и "узким" z-буфером?
15. Охарактеризуйте z-сортировку.
16. Охарактеризуйте z-буферизацию.
17. Что такое "Мип-мэппинг", для чего он нужен и каковы его недостатки?
18. Где и как можно хранить текстуры для мип-мэппинга?
19. Что такое мип-каскад и как он может быть создан?
20. Какие разновидности дефектов возникают при дискретной замене текстур, предусмотренной мип-мэппингом?
21. Зачем нужна фильтрация и в чем ее суть?
22. Какие виды фильтрации различают в зависимости от используемых методов интерполяции?
23. Какой тип фильтрации наиболее распространен и почему?
24. Что понимается под мультитекстурированием и как оно реализуется в 3D-акселераторах?
25. В чем суть текстурирования с коррекцией перспективы?
26. Что понимается под рельефным текстурированием?
27. Какова основная идея рельефного текстурирования?
28. Перечислите основные разновидности рельефного текстурирования.
29. В чем суть текстурирования методом вдавливания?
30. Где и для чего применяется рельефная карта и как она получается?
31. Как используется рельефная карта при расчете освещенности?
32. Охарактеризуйте недостатки метода вдавливания.
33. Какие карты используются при текстурировании методом карт окружающей среды и как они используются?
34. Какие эффекты можно получить текстурированием с использованием карт окружающей среды совместно с динамическим изменением рельефной карты?
35. Для чего используется α-смешение?
36. В чем суть альфа-смешения?
37. Что понимается под альфа-буфером?
38. При выполнении каких функций 3D-акселерации применяется альфа-смешение?
39. Какие операции могут выполняться с использованием исходного пиксела и содержащегося в кадровом буфере?
40. Как реализуется эффект затуманивания?
41. Какие виды затуманивания могут использоваться на практике?
42. В чем суть метода морфинга уровней детализации (LOD-Morphing)?
43. В чем заключаются прием "настройка цвета текстуры" при управлении прозрачностью объекта?
44. В чем суть функции 3D-акселератора "Зеркальные блики" (Specular Highlights)?

45. Как хранится информация в формате RGB и RGBα для режимов High Color и True Color?
46. Охарактеризуйте хранение расцветки текстур в режиме с палитрой.
47. На чем основана технология сжатия текстур S3TC?
48. Каким образом осуществляется сжатие текстур по технологии S3TC?
49. Во сколько раз сжимается информация по технологии S3TC для 16-разрядного RGB и для 32-разрядного RGBα режимов?
50. Как производится декомпрессия формата S3TC?
51. Какие методы сжатия, помимо S3TC, могут использоваться для хранения текстур?
52. Что такое алиасинг и антиалиасинг?
53. В чем суть краевого антиалиасинга?
54. В чем достоинства и недостатки краевого антиалиасинга?
55. В чем состоит основная идея полного антиалиасинга (субпиксельного антиалиасинга)?
56. Какими методами получают субпикселы?
57. В чем заключается эффект блидинга (bleeding)?
58. Что понимают под субпикселом?
59. Что понимают под суперсемплингом?
60. Что понимают под мультисемплингом?
61. Что понимается под тайловой архитектурой локальной памяти 3D-акселератора?
62. Для чего и как используется метод интерполяции цветов (дизеринг)?

В разделе использованы материалы из [1].

3.1.3. Интерфейсы прикладного программирования

Поскольку 3D-акселератор имеет собственную систему команд, его эффективное применение возможно лишь в том случае, если прикладная программа использует эти же команды. Но поскольку различных программ и моделей 3D-акселераторов существует великое множество, возникает проблема совместимости: невозможно написать такую программу, которая одинаково хорошо поддерживала бы системы низкоуровневых команд различных акселераторов. Очевидно, что и разработчики прикладного программного обеспечения, и производители 3D-акселераторов нуждаются в некоем «арбитре» или «переводчике», который выполнял бы следующие функции:

- эффективное преобразование запросов прикладной программы в оптимизированную последовательность низкоуровневых команд 3D-акселератора с учетом особенностей его аппаратного построения;
- программную эмуляцию запрошенных функций, если в используемом акселераторе отсутствует их аппаратная поддержка;
- стимулирование производителей акселераторов к унификации используемой системы команд с учетом нужд производителей программного обеспечения.

Роль подобного «арбитра» была возложена на специальный пакет служебных программ, называемый *интерфейс прикладного программирования* или *API (Application Program Interface).* API занимает промежуточное положение между высокоуровневыми прикладными программами и низкоуровневой программой — драйвером акселератора, который имеет собственную систему команд. Использование API избавляет разработчика прикладной программы от необходимости работать с низкоуровневыми командами, облегчая процесс создания программ и делая их более универсальными.

Таким образом, остается один вопрос: какие API использовать для обеспечения ускорения 3D-графики и как реализована их поддержка производителями программ и аппаратных средств?

В настоящее время существует несколько API, сферы применения которых довольно четко разделены.

В области игровых приложений, ориентированных на работу под управлением операционной системы Windows 95/98, господствует API DirectX, разработанный фирмой Microsoft.

В сфере профессиональных приложений (САПР, системы трехмерного моделирования, симуляторы и т. п.), работающих под управлением операционной системы Windows NT, наибольшее распространение получил API OpenGL, разработанный общепризнанным лидером в области мощных графических станций — компанией Silicon Graphics, Inc. (SGI). Однако есть и классический пример игры, ориентированной на работу под управлением OpenGL, — это легендарный Quake II. Более того, в последнее время совершенно четко наметилась тенденция к расширению сфер применения OpenGL. Практически все новейшие аппаратные средства для ускорения 3D-графики поставляются с полноценными драйверами для OpenGL.

И, наконец, существует значительное число API, создаваемых производителями непосредственно для своих Chipset с целью наиболее эффективного и оперативного использования новейших аппаратно поддерживаемых функций. Использование такого API полностью освобождает программиста от необходимости работы с низкоуровневыми командами акселератора. Такие интерфейсы называют *«родными»* или *native API.* В качестве примера можно назвать API Glide компании 3Dfx (предназначен для Chipset семейства Voodoo), RenderGL компании Intergraph, Heidi компании Autudesk, MeTAL компании S3 (для Chipset семейства Savage) и др.

3.1.3.1. DirectX

Подавляющее число игр ориентировано на работу совместно с DirectX, который включает в себя несколько специализированных API:

- DirectDraw; обеспечивает использование аппаратных средств ускорения обычной, двумерной графики;
- Direct3D; отвечает за работу графической системы в режиме создания трехмерных изображений;
- DirectInput; обеспечивает аппаратно независимый ввод информации в PC с помощью клавиатуры, мыши и джойстика;
- DirectPlay; используется при совместной игре на нескольких компьютерах, объединенных в сеть или соединенных непосредственно — через параллельный или последовательный порт;
- DirectSound; управляет использованием ресурсов звуковой системы PC.

DirectX является жестко регламентированным, закрытым стандартом, который не допускает изменений до выхода в свет своей очередной, новой версии. Это, с одной стороны, ограничивает возможности разработчиков программ и особенно производителей акселераторов, а с другой — значительно облегчает пользователю настройку программного и аппаратного обеспечения.

При работе с трехмерной графикой используется Direct3D. Этот API проверяет функциональные возможности установленного аппаратного обеспечения: прикладная программа сначала запрашивает Direct3D-совместимый драйвер об аппаратно поддерживаемых данным акселератором 3D-функциях, а затем, в зависимости от ответа, разрешает активизацию только тех функций, которые реально поддерживаются

данным 3D-акселератором. Очевидно, что такой алгоритм работы является наилучшим, так как избавляет от необходимости вникать в детали и производить ручную настройку драйвера 3D-акселератора.

Запрашиваемый список 3D-функций строго регламентирован. В частности, до недавнего времени API Direct3D не поддерживал работу с геометрическим процессором, предназначенным для ускорения первой стадии 3D-конвейера (только начиная с версии 7.0 DirectX поддерживает эту функцию).

3.1.3.2. OpenGL

В отличие от DirectX, API OpenGL (табл. 3.1) построен на основе концепции открытого стандарта, имеющего небольшой базовый набор функций и множество расширений, реализующих более сложные функции. Производитель Chipset (карты 3D-акселератора) обязан создать BIOS и драйверы, реализующие базовые функции OpenGL, но не обязан обеспечивать поддержку всех расширений. С другой стороны, любой *независимый производитель аппаратного обеспечения* (IHV — *Independent Hardware Vendor*) имеет право создавать собственные расширения стандарта, обеспечивающие реализацию специфических возможностей своих изделий. В результате появилось множество расширений OpenGL.

Создана ассоциация OpenGL ARB (*Architecture Review Board* — совет по пересмотру архитектуры). В нее входят компании DEC, Evans & Sutherland (E&S), IBM, Intel, Intergraph, Microsoft, Silicon Graphic, Inc. Такая стратегия облегчает переносимость наиболее важных функциональных возможностей, но крайне затрудняет создание универсальных программ для различных акселераторов многочисленными разработчиками программного обеспечения. Поэтому API OpenGL пока не получил широкого распространения и используется в основном в профессиональных приложениях, разработчики которых активно взаимодействуют с производителями 3D-ускорителей. Но даже несмотря на это взаимодействие, для работы большинства профессиональных графических программ, ориентированных на OpenGL, требуется ручная настройка функций, поддерживаемых конкретным установленным в системе 3D-акселератором.

3.1.3.2.1. ICD

Полная версия OpenGL-совместимого драйвера носит название ICD *(Installable Client Driver*— драйвер приложения-клиента). Этот драйвер обеспечивает максимальное быстродействие, поскольку содержит оптимизированные низкоуровневые коды не только базового набора функций, но и его расширений. Естественно, что с учетом концепции OpenGL создание подобного драйвера— исключительно сложный и трудоемкий процесс. Это, кстати, во многом объясняет более высокую стоимость профессиональных 3D-акселераторов по сравнению с игровыми.

3.1.3.2.2. MCD

Усеченная версия OpenGL-совместимого драйвера называется *MCD (Mini Client Driver).* Такой драйвер гораздо проще написать. Он содержит оптимизированный код функций, выполняемых лишь на некоторых этапах 3D-конвейера, поэтому акселератор под его управлением работает медленнее. Более того, допускается работа только в Windows NT, а при отсутствии аппаратной поддержки запрошенной функции MCD и вовсе отключает акселерацию. Как правило, драйверы типа MCD поставляются с первыми версиями новых ускорителей, а полноценные ICD появляются позже.

3.1.3.2.3. Минипорт

Кроме ICD и MCD, существуют и специализированные OpenGL-совместимые драйверы, специально созданные для работы с какой-либо конкретной программой (игрой). Такая ситуация возникает, например, при желании поиграть в QuakeGL или Quake II на PC с Windows 95 и 3D-акселератором, не рассчитанным на использование. OpenGL. Такой драйвер принято называть *минипортом.*

3.1.3.2.4. Раппер

Также как и ICD, минипорт может работать используя собственный, native API для установленного в системе акселератора или путем перевода инструкций OpenGL в инструкции Direct3D. В последнем случае минипорт называют *раппером (wrapper).* Из всех типов драйверов раппер обеспечивает самую низкую скорость работы 3D- акселератора.

3.1.3.2.5. Game Engine

Существует еще одно родственное минипорту понятие — *Game Engine,* называемое на компьютерном жаргоне «игровой движок». Фактически это игровой драйвер, разработанный для конкретного 3D-ускорителя и ориентированный на достижение максимальной производительности за счет непосредственного использования низкоуровневых команд акселератора в обход API.

В целом API OpenGL ориентирован на корректность создаваемых изображений, тогда как у DirectX на первом месте стоят скорость прорисовки и естественность изображения (что наиболее важно в играх). Не случайно в OpenGL, начиная с его первой версии 1.1, реализована поддержка аппаратного ускорения геометрической стадии конвейера, которая особенно важна при работе с системами моделирования и САПР. В табл. 3.1 приведен перечень 3D-функций, аппаратное ускорение которых поддерживается DirectX и

OpenGL.

Таблица 3.1. Сравнительная характеристика возможностей API DirectX и OpenGL

Функция	API			
	Текущая версия		Следующая версия	
	OpenGL 1.1	Drect3D 5.0	OpenGL 1.2	Drect3D 7.0
Поддержка геометрического процессора	•	o	•	•
Поддержка тайловой архитектуры	o	•	•	•
Зеркальные блики	o	•	•	•
Авто-мип-мэппинг	o	•	•	•
Табличный туман	o	•	•	•
Мультитекстурирование	•	o	•	•
Рельефное текстурирование	o	o	•	•
Компрессия текстур	o	o	o	•

3.1.3.3. Проект Fahrenheit

Поскольку наличие двух конкурирующих API затрудняет продвижение на рынок продукции различных производителей, логично предположить, что в будущем эти два API должны объединиться в один, вобрав в себя все позитивные свойства каждого. Первым шагом на пути к этому стало создание стратегического союза между фирмами Microsoft и SGI, объединивших свои усилия в рамках нового грандиозного проекта.

Данный проект получил название *Fahrenheit* и был анонсирован 17 декабря 1997 г. В соответствии с ним, планируется создать мощный универсальный API, который позволит охватить практически все сферы 3D-приложений: системы автоматизированного проектирования и моделирования, медицинские и научные приложения, игры, Internet и многое другое. Данный API сможет применяться не только на компьютерах с ОС Windows, но и на мощных графических станциях фирмы SGI, работающих под управлением ОС UNIX.

Первым, предварительным шагом на этом пути стала совместная разработка пакета *3D Graphics Device Driver Kit (DDK)* — набора драйверов 3D-устройств. Данный пакет позволил разработчикам программного обеспечения использовать OpenGL под управлением Windows 95/NT. В целом проект Fahrenheit предусматривает создание трех новых API.

Первый из них получил условное название *Fahrenheit Scene Graph.* Это низкоуровневый API, основанный на технологиях *Scene Graph* фирмы SGI и *COM (Common Object Model)* компании Microsoft. Этот API поддерживает многопроцессорные системы.

Второй API носит название *Fahrenheit Large Model Visualization* — интерфейс визуализации больших моделей. Он является расширением API Scene Graph и предназначен в основном для профессиональных приложений. Основан на интерфейсах Graphic OpenGL Optimizer фирмы SGI и Direct Model фирм Hewlett-Packard и Microsoft.

Третий API получил название *Fahrenheit Low Level API*— API низкого уровня. Он основан на DirectX 5.x и OpenGL и будет основным графическим интерфейсом в среде Windows. В нем предусмотрена полная совместимость «сверху вниз» с приложениями и драйверами, написанными для DirectDraw, Direct3D и OpenGL.

Если проекту Fahrenheit будет сопутствовать успех, то в ближайшем будущем отпадет необходимость в большинстве узкоспециализированных и особенно native API.

3.1.3.4. Оптимизация 3D-конвейера средствами API

Заметим, что все сказанное об API в основном касается либо производителей аппаратных средств, либо разработчиков программного обеспечения. Надо ли все это знать обычному пользователю? По нашему мнению, надо, хотя бы для того, чтобы грамотно настроить видеосистему своего PC. Если вы хотите добиться от своего 3D-акселератора максимальной производительности, оправдав тем самым вложенные в него средства, необходимо убедиться в том, что аппаратно реализованные в нем 3D-функции используются в полном объеме. Как это сделать?

1. При выборе 3D-акселератора необходимо четко представлять себе область его будущего применения, т. е. с какими программами он будет использоваться и какой API (с учетом версии) нужен для его работы. Следовательно, необходимо выяснить у продавца или по документации, совместимы ли прилагаемые к плате драйверы с требуемой версией выбранного API. Для любителей игр, работающих в Windows 95/98, достаточно убедиться в наличии поддержки DirectX.

2. Следует выяснить, как устанавливается в систему требуемый API. Если речь идет об играх или других программах, ориентированных на DirectX, то они, как правило, при инсталляции автоматически

копируют необходимые элементы нужного им API. Возможен также случай, когда программа инсталляции предложит самостоятельно установить (обновить) файлы библиотек API, используя его отдельный дистрибутив. Как правило, такие дистрибутивы поставляются вместе с игрой.

3. После инсталляции следует дополнительно убедиться в совместимости установленного драйвера с той версией **API,** которая требуется прикладной программе. Для этого достаточно запустить эту программу Если появится сообщение о ее несовместимости с API установленной версии, необходимо либо обновить драйвер 3D-акселератора, либо установить API более новой версии.

4. Следует проверить настройку параметров драйвера и/или прикладной программы, задействовав при необходимости реализованные в 3D-акселераторе функции вручную. Настройка драйвера монитора выполняется путем выбора соответствующих опций диалогового окна **Свойства: Экран.** На рис. 3.34 в качестве примера представлены два диалоговых окна драйвера 3D-акселератора на основе популярного Chipset Riva TNT.

Из рисунка 3.34 видно, что с помощью опций этих окон можно управлять большинством рассмотренных функций аппаратного ускорения 3D-графики. Настройки, выполненные в этих окнах, будут использоваться по умолчанию.

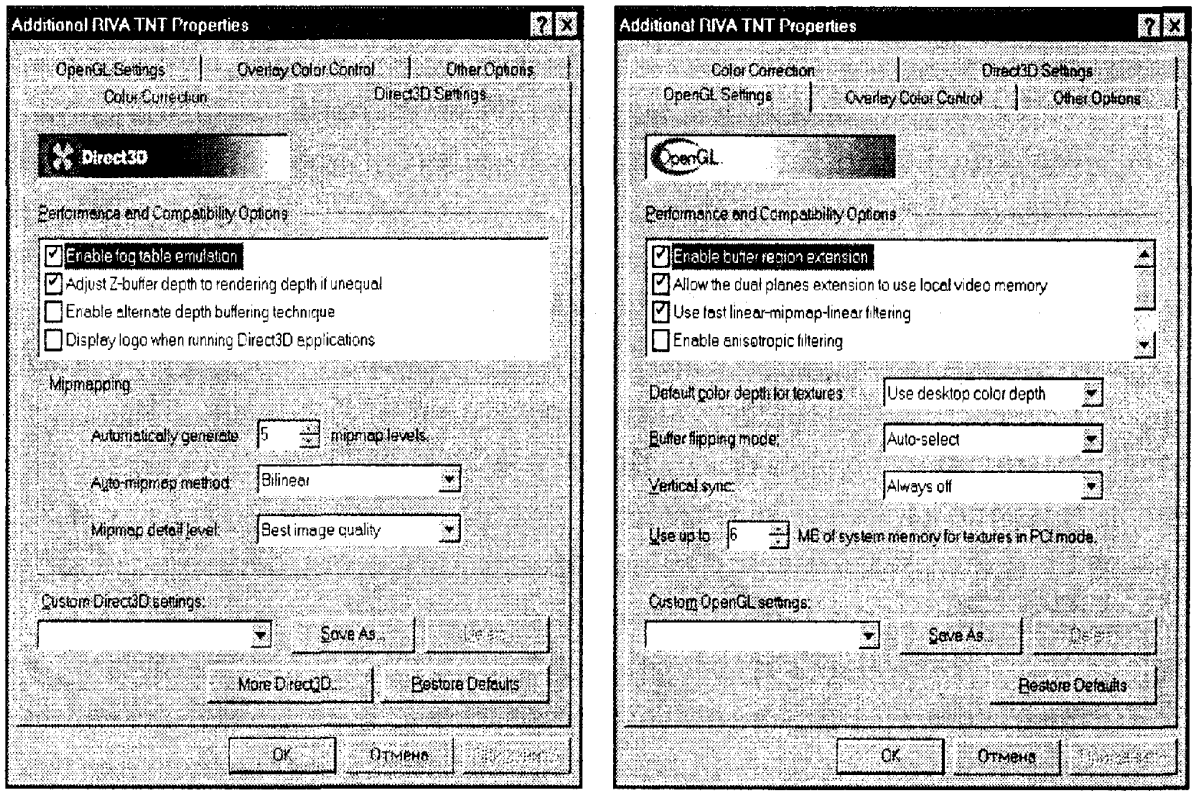


Рис. 3.34. Диалоговые окна для настройки опций 3D-ускорения API DirectX (слева) и OpenGL (справа) для 3D-акселератора на основе Chipset Riva TNT

Кроме того, с помощью программы DXDiag.exe (рис. 3.35) можно отредактировать настройки библиотеки DirectX.

Отметим, что с помощью этой программы можно получить исчерпывающую информацию об установленных компонентах DirectX и драйверах различных мультимедийных устройств, а также проверить корректность функционирования программного и аппаратного обеспечения.

Помимо настройки системного программного обеспечения, к которому относятся драйверы и библиотеки API, необходимо настроить прикладную программу, которая использует функции 3D-ускорения. В качестве примера на рис.3.36 приведено диалоговое окно, содержащее список функций 3D-акселерации, используемых в игре Microsoft Combat Flight Simulator.

Используя тесты и субъективно оценивая качество изображения, подберите такой набор функций, при котором скорость работы и качество изображения будут наилучшими. Если при включении каких-либо функций значительно повышается производительность, значит они поддерживаются аппаратно. Если же, напротив, их включение приводит к замедлению работы, то акселерация отсутствует.

Возможно при использовании программ, ориентированных на DirectX, вам, вообще не придется выполнять ни одну из перечисленных операций: все будет сделано автоматически. Однако никогда не мешает знать, что именно установлено на вашем компьютере и как оно работает, и при возникновении любых проблем вы сможете осознанно искать их решение.

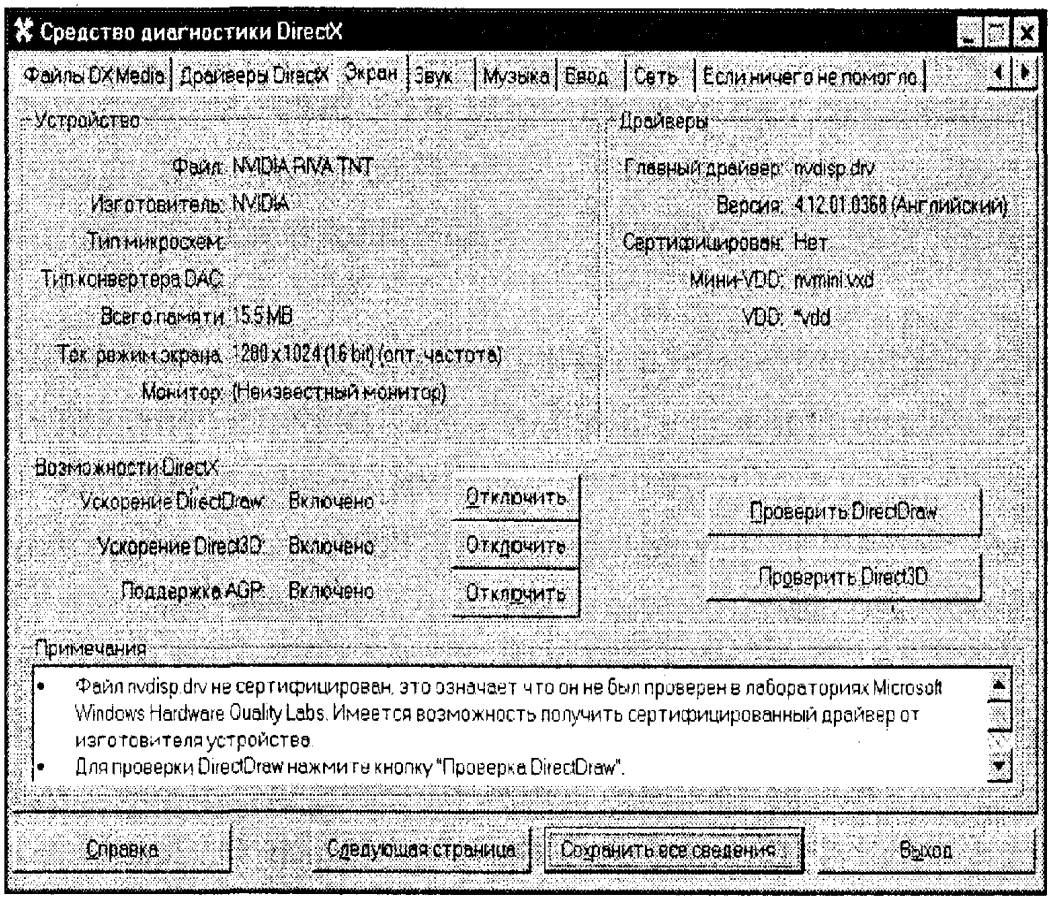


Рис. 3.35. Диалоговое окно для настройки функций 3D-акселерации API DirectX

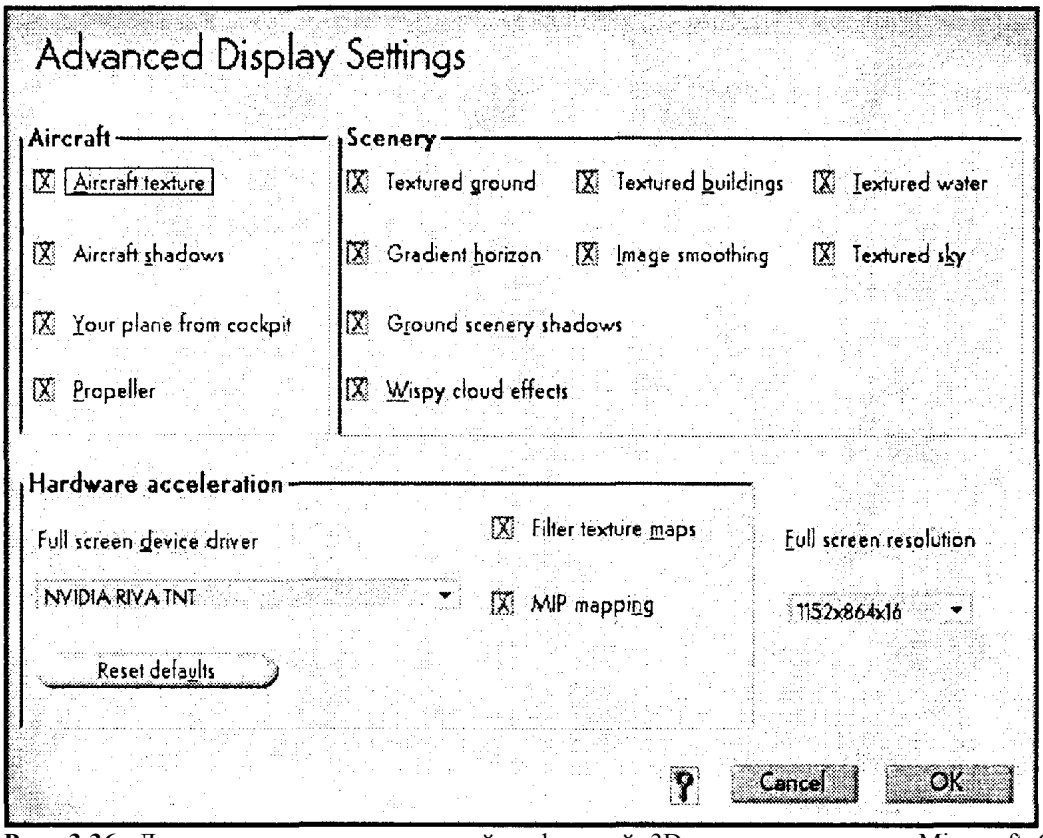


Рис. 3.36. Диалоговое окно для настройки функций 3D-акселерации игры Microsoft Combat Flight Simulator

Контрольные вопросы.

1. Какие функции выполняет интерфейс прикладного программирования (API)?
2. Какое место занимает API в иерархии программного обеспечения?
3. В какой операционной среде и для чего применяется API DirectX?
4. В какой операционной среде и для чего применяется API OpenGL?
5. Охарактеризуйте другие API для 3D-графики.
6. Какие специализированные API включает в себя DirectX?
7. Охарактеризуйте API DirectX.
8. Как действует Direct3D?
9. Кратко охарактеризуйте API OpenGL.
10. Какие функции поддерживает OpenGL 1.2 и Direct3D 7.0?
11. Охарактеризуйте ICD.
12. Охарактеризуйте MCD.
13. Что понимается под минипортом?
14. Что понимается под раппером?
15. Что понимается под Game Engine?
16. Какова цель проекта Fahrenheir?
17. Что представляет собой DDK и что он позволяет делать?
18. Что предусматривает проект Fahrenheir?
19. В чем заключается оптимизация 3D-конвейера?
20. Сформулируйте рекомендации по выбору 3D-акселераторов и API.
21. Как производится настройка параметров драйвера 3D-ускорителя и/или прикладной программы?

В данном разделе использованы материалы из [1].

3.2. Устройство и основные характеристики ускорителей 3D-графики

Для грамотного выбора 3D-акселератора необходимо хотя бы в общих чертах представлять себе его устройство, определяющее потенциальные возможности акселератора, и знать основные характеристики, отражающие качество реализации этих возможностей. Кроме того, чтобы оптимальным образом вложить средства, необходимо четко определиться, для решения каких конкретных задач планируется использовать 3D-акселератор. Не стоит переплачивать за те функции видеоадаптера с 3D-акселератором, которые вы не собираетесь использовать.

3.2.1. Структура типового 3D-акселератора

Несмотря на значительные различия в характеристиках и возможностях, все 3D-акселераторы имеют несколько базовых элементов, обеспечивающих аппаратное ускорение основных этапов 3D-конвейера (рис. 3.37):

- геометрический процессор (*Geometry Processor*);
- механизм прорисовки *(Rendering Engine)*;
- видеопамять;
- цифро-аналоговый преобразователь (RAMDAC);
- дополнительные устройства (в зависимости от набора дополнительных мультимедийных функций).

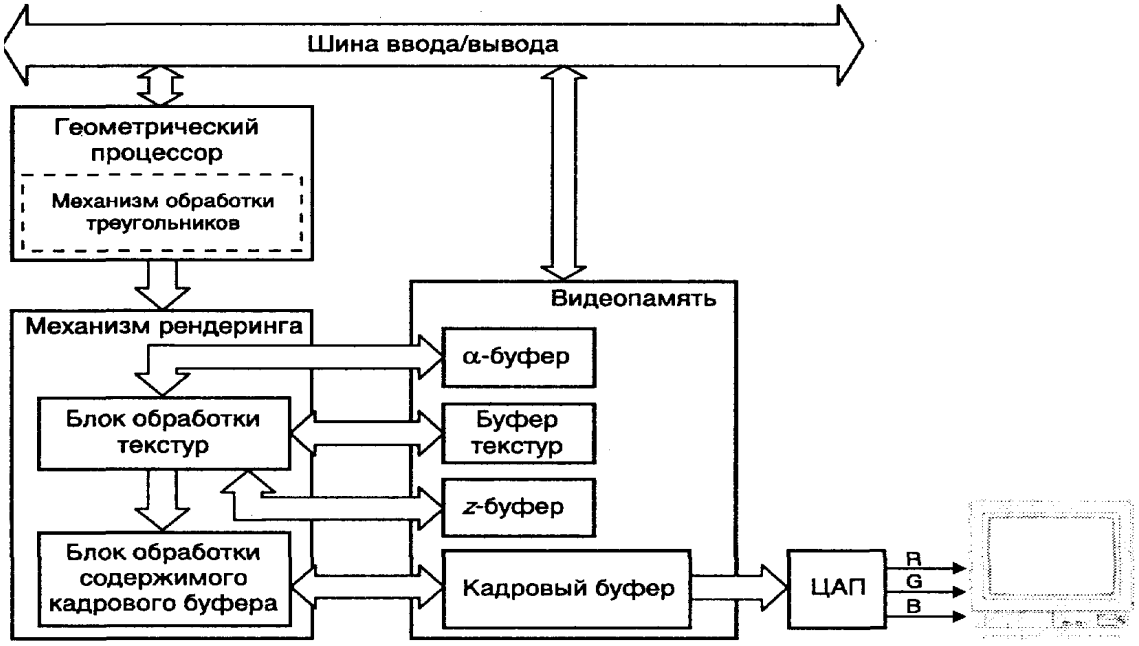


Рис. 3.37. Обобщенная структурная схема 3D-акселератора

Геометрический процессор предназначен для аппаратного ускорения этапов 2—6 геометрической стадии 3D-конвейера. В большинстве недорогих 3D-акселераторов геометрический процессор отсутствует и его функции выполняет CPU. В лучшем случае такой акселератор имеет только *механизм установки (Setup Engine),* выполняющий преобразование формы представления координат вершин граней, сортировку и отбрасывание задних граней (этап Triangle Setup).

Для ускорения работы блока Setup Engine применяется технология *strips&fans.* Суть этой технологии заключается в следующем. Как минимум три смежных грани имеют одну общую вершину, т. е. при обработке каждой смежной грани координаты этой вершины будут обрабатываться в блоке Setup Engine также как минимум три раза. Чтобы по несколько раз не обрабатывать одни и те же вершины, обработке последовательно подвергаются грани, расположенные в определенном порядке. Этот порядок и называют *strips&fans* — полоски и лоскутки. Подобные продольные «полоски» видны на рис. 3.4.

Примером такого устройства является процессор Delta на плате Permedia 3 фирмы 3Dlabs, хотя сама фирма называет его геометрическим процессором. Другое дело — профессиональные 3D-акселераторы, ориентированные на работу под управлением API OpenGL. Многие из них содержат специальные мощные процессоры, обеспечивающие аппаратное ускорение операций матричной (линейной) алгебры, т. е. ускорение выполнения этапов геометрической стадии 3D-конвейера.

Механизм прорисовки, часто называемый *конвейером рендеринга,* является основной частью типового 3D-акселератора и включает в себя два элемента: механизм обработки элементов текстур *(Texel Engine)* и механизм обработки итогового кадра *(Pixel Engine).* Каждый из этих элементов использует свой участок локальной памяти, называемый *буфером текстур* и *кадровым буфером* соответственно. Часто для обозначения Texel Engine и Pixel Engine используются термины *блок* или *модуль.*

Помимо кадрового буфера и буфера текстур, в локальной памяти акселератора обычно выделяется область для дополнительных буферов:

- z-буфера, необходимого для корректного удаления скрытых поверхностей;
- α-буфера, необходимого для эффективного выполнения альфа-смешения;
- второго кадрового буфера, используемого при двойной буферизации.

Такую архитектуру в общих чертах имеет 3D-акселератор на основе Chipset Voodoo Graphics производства фирмы 3Dfx Interactive. Базовая, минимальная конфигурация данного Chipset включает в себя две микросхемы, имеющие условные обозначения PixelFX и TexelFX. Согласно терминологии фирмы-разработчика, название первой из них звучит как *FBI* (*Frame Buffer Interface* — интерфейс кадрового буфера), а второй — *TMU* (*Texture Mapping Unit* — блок наложения текстур). Каждая микросхема работает со своим буфером памяти, размер которого может достигать 4 Мбайт. Максимальная конфигурация платы 3D-акселератора: один блок FBI и три блока TMU.

Однако существует плата (Obsidian 100SB компании Quantum3D), которая имеет конфигурацию 2 × (1 PixelFX + 1 (2) TexelFX), т. е. фактически представляет собой два одинаковых 3D-ускорителя, смонтированных на одной плате. Такая избыточность позволяет реализовать очень интересный режим, называемый SLI (*Scan Lines Interleave* — чередование обработки строк растра). Суть его состоит в том, что одна часть системы генерирует только четные строки кадрового буфера, а вторая — только нечетные. При этом вдвое увеличивается требуемый объем кадрового буфера (значит, возможно получение более высокого разрешения!), но также вдвое возрастает и общая производительность.

Размер кадрового буфера, по аналогии с видеопамятью обычного видеоадаптера VGA, определяет максимально допустимые разрешение изображения и размер палитры. В большинстве акселераторов для ускорения работы используется метод двойной буферизации, в соответствии с которым кадровый буфер разбивается на две части: передний *(front buffer)* и задний *(back buffer)* буферы. В то время как выполняется отображение (передача в RAMDAC) первого, «переднего» кадра, акселератор приступает к построению следующего кадра в заднем буфере. Благодаря этому обеспечивается более плавная и быстрая смена кадров.

3.2.2. Основные характеристики 3D-акселераторов

Каковы же основные технические характеристики, определяющие потенциальные возможности 3D-акселератора, и ожидаемый эффект от его использования?

Во-первых, для оценки 3D-акселератора используются стандартные характеристики видеоадаптеров, рассмотренные ранее: разрядность, объем и пропускная способность видеопамяти, тактовые частоты графического процессора (ядра), видеопамяти и RAMDAC, тип интерфейса с шиной ввода/вывода и т. п. Поскольку основной сферой применения рассматриваемых акселераторов являются трехмерные игры, очень важен и удобен интегральный показатель производительности — обеспечиваемая *частота смены кадров,* т.е. *frame per second,* сокращенно *fps*.

Во-вторых, для описания возможностей 3D-акселератора вводится целый ряд специфических показателей:

- *пропускная способность,* измеряемая количеством обрабатываемых элементарных треугольников в секунду *(Triangle Throughput);*
- *максимальная скорость закраски (Fill Rate),* измеряемая количеством пикселов в секунду;
- набор аппаратно реализуемых (ускоряемых) 3D-функций, а также качество (корректность) их реализации;
- наличие расширенных режимов (например, SLI, MAXX, VSA);
- перечень поддерживаемых API, а также степень поддержки (например, ICD или MCD для OpenGL).

Особенностью первого показателя (Triangle Throughput) является то, что при отсутствии в составе 3D-акселератора геометрического процессора он характеризует не только и даже не столько сам акселератор, сколько производительность центрального процессора, поскольку в этом случае геометрическая стадия 3D-конвейера реализуется программно. Чем выше скорость работы центрального процессора, тем больше треугольников в секунду он сможет сгенерировать и тем больше треугольников сможет закрасить блок рендеринга.

Скорость закраски (Fill Rate) характеризует потенциальную скорость выполнения этапа рендеринга; фактическая же скорость закраски зависит от темпа поступления на вход блока рендеринга координат обработанных вершин граней, т. е. от скорости выполнения геометрической стадии 3D-конвейера. В свою очередь, скорость выполнения геометрической стадии определяется быстродействием центрального процессора. Таким образом, производительность 3D-акселератора сильно зависит от быстродействия CPU. Эту зависимость называют *масштабируемостью* 3D-акселератора.

Однако значимость показателя Triangle Throughput возрастает для тех плат, которые имеют аппаратно реализованный блок Setup Engine (например, упомянутая выше Permedia 3 фирмы 3Dlabs) или полноценный геометрический процессор (nVidia GeForce 256, ATI Radeon 256, S3 Savage 2000).

Помимо чисто аппаратных возможностей, большое значение имеет качество программного обеспечения

для 3D-акселератора, т. е. качество драйверов. Как уже неоднократно отмечалось, этот фактор часто имеет решающее значение в обеспечении устойчивой и эффективной работы 3D-акселератора.

Важной характеристикой ускорителя является также набор поддерживаемых API. В настоящее время обязательным требованием, предъявляемым к 3D-акселератору, является полноценная поддержка ускорителем как минимум двух API: DirectX и OpenGL (на уровне ICD).

3.2.3. 3D-акселераторы фотореалистичной графики

В конце прошлого века на рынке игровых 3D-акселераторов появились новые мощные устройства, обеспечивающие качественно новый, более высокий уровень синтеза трехмерной сцены. Эти 3D-акселераторы были в состоянии синтезировать изображение со следующими характеристиками:

- высоким разрешением (до 2048×1536);
- 32-битным цветом;
- очень высокой степенью детализации, обусловленной использованием сложных геометрических моделей и текстур очень больших размеров (до 2048×2048).

Изображение с такими характеристиками (рис. 3.38) настолько похоже на естественное, полученное, например, при фотографировании, что возникло новое понятие — *фотореалистичная графика.*

Рис.3.38. Пример фотореалистичного изображения

Для того чтобы достичь такого результата, характеристики основных элементов 3D-акселератора были радикально улучшены. В качестве примера 3D-акселератора фотореалистичной графики рассмотрим структуру и основные параметры изделия канадской фирмы ATI Technologies — видеоадаптера Radeon 256 (рис. 3.39), выполненного на базе графического процессора RADEON 256 с мощным 3D-ядром.

Структура Radeon 256 имеет множество отличий от базовой структуры 3D-акселератора, наиболее существенными из которых являются следующие.

1. Как и все изделия этого класса, Radeon 256 представляет собой не специализированный 3D-акселератор, а мощный многофункциональный видеоадаптер, у которого ускорение 3D-графики — одна из множества функций, среди которых: аппаратное ускорение 2D-графики, ввод/вывод видеосигналов, формирование цифрового видеосигнала для управления плоскопанельными ЖК-мониторами, декодирование цифровых видеопотоков, сжатых по технологии MPEG-1/2, и др.

2. Наличие специального геометрического процессора *Charisma Engine* обеспечивает аппаратное ускорение функций, выполняемых на геометрической стадии 3D-конвейера: преобразование координат (при моделировании движения и вращения), расстановка источников света и расчет освещенности вершин граней, позволяющая использовать до 8 источников света, отсечение невидимых граней, коррекция перспективы, преобразование формата представления данных о вершинах (замена чисел с плавающей точкой числами с фиксированной точкой), модификация граней в местах резкого изгиба поверхности (функция *Vertex Skinning),* интерполяция промежуточных кадров по ключевым кадрам *(Key Frame Interpolation),* позволяющая быстро рассчитать анимационную картинку, основываясь только на начальном и конечном кадрах. Столь внушительный арсенал средств позволяет многократно повысить скорость выполнения геометрической стадии 3D-конвейера: плата обрабатывает до 30 млн треугольников в секунду (для сравнения: 3D-акселератор Voodoo Graphics — только 500 тыс.).

3. Значительно увеличены объем и скорость работы видеопамяти; на карте видеоадаптера может быть установлено от 8 до 128 Мбайт быстродействующей видеопамяти типа SDRAM/SGRAM или DDR SDRAM, работающей на частоте до 200 МГц.

4. Увеличена разрядность шины памяти до 256.

5. За счет использования 0,18-микрометровой технологии изготовления микросхем тактовая частота графического 256-разрядного ядра, состоящего из двух 128-разрядных блоков, лежит в пределах от 200 до

400 МГц.

6. Графическое ядро имеет ярко выраженную конвейерную архитектуру: два конвейера рендеринга, каждый из которых содержит по 3 блока (эта технология получила название *Pixel Tapestry).* Благодаря этому производительность конвейера рендеринга составляет 400 млн пикселов в секунду (у 3D-акселератора Voodoo Graphics только 50 млн).

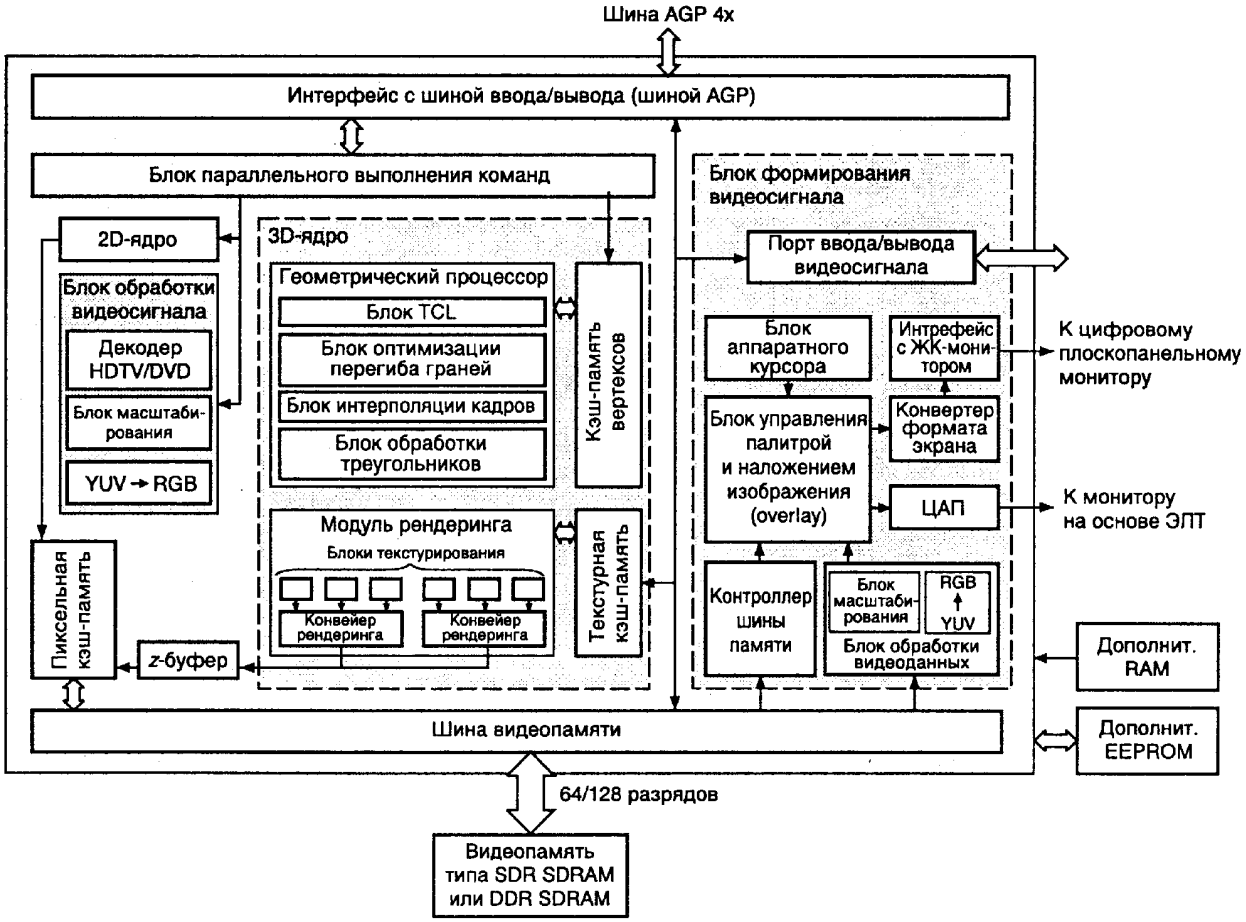


Рис. 3.39. Структурная схема видеоадаптера ATI Radeon 256

7. Широко используется кэш-память, ускоряющая построение изображения: текстурная, пиксельная и кэш-память вершин граней.

8. Аппаратная поддержка наиболее эффективных методов текстурирования: рельефное текстурирование, специальные методы наложения и сжатия текстур и др.

9. Высокоскоростной интерфейс с шиной ввода/вывода (порт AGP, карта поддерживает режимы «4×», DIME, SBA Fast Writes).

Благодаря перечисленным особенностям, Radeon 256 обеспечивает более высокий, фотореалистический уровень моделирования 3D-сцены, что позволило отнести этот видеоадаптер к новому поколению 3D-акселераторов. К этой же группе относятся изделия GeForce 256 (NV10) и GeForce 2 GTS (NV15) производства фирмы nVidia, Savage 2000 производства фирмы S3 и ряд других.

В настоящее время такие архитектуры видеоплат продолжают бурно развиваться в основном за счет применения более мощных современных графических процессоров (см. раздел 3.3).

5.2.4. Ускоренный графический порт (AGP)

Для аппаратной реализации 3D-конвейера акселератор должен иметь локальную память значительно большего объема (не менее 8—12 Мбайт), чем требуется видеоадаптеру. В первую очередь, это обусловлено необходимостью хранения и обработки в видеопамяти множества растровых изображений — текстур. Чем выше качество прорисовки 3D-сцены (разрешение и размер палитры), тем более крупные текстуры приходится использовать, и тем больше места они занимают в локальной памяти 3D-акселератора. А увеличение размера видеопамяти влечет за собой повышение стоимости платы 3D-акселератора.

С другой стороны, видеопамять используется полностью только тогда, когда в программе используются текстуры больших размеров. В остальных случаях (особенно при работе с 2D-приложениями) значительная часть дорогостоящей видеопамяти оказывается незадействованной.

С целью удешевления своих изделий, производители 3D-акселераторов предложили хранить текстуры в

обычной, системной памяти PC и загружать их в видеопамять по мере необходимости. Благодаря этому появилась возможность значительно сэкономить на локальной памяти 3D-акселератора.

Отметим, что эта идея не нова. Она уже давно используется в PC, например, для реализации так называемой технологии UMA *(Unified Memory Architecture* — архитектура унифицированной памяти), когда видеоадаптер для обеспечения выбранного видеорежима задействует часть системной памяти. Эта технология обычно использовалась во встроенных в материнскую плату видеоадаптерах с видеопамятью небольшого объема. В результате стоимость такой материнской платы оказывалась значительно ниже, чем суммарная стоимость обычной материнской платы и высококачественного видеоадаптера с видеопамятью большого объема. Однако низкая производительность такой видеосистемы, обусловленная главным образом недостаточной пропускной способностью шины ввода/вывода, не позволяла использовать архитектуру UMA для обработки 3D-графики.

Для повышения скорости обмена данными между системной памятью PC и локальной памятью 3D-акселератора фирма Intel разработала высокоскоростную локальная шину ввода/вывода AGP, предназначенную исключительно для нужд видеоадаптера с 3D-акселератором в системе с процессорами шестого поколения (Pentium Pro, Celeron, Pentium II/III). Поскольку шина AGP связывает только видеоадаптер (одно устройство) с системной памятью PC, не возникает характерной для шины PCI проблемы арбитража, когда несколько устройств одновременно требуют доступа к шине, что значительно снижает скорость обмена данными. Соответственно, на материнской плате имеется только один разъем (слот) AGP.

Шина AGP была разработана на основе архитектуры шины PCI, поэтому она является также 32-разрядной. Вместе с тем шина AGP имеет ряд важных отличий (режимов) от шины PCI, благодаря чему удалось в несколько раз повысить ее пропускную способность:

- режимы «2x» и «4x»;
- режим Pipelining;
- режим SBA;
- режим DIME.

Рассмотрим эти режимы более подробно.

3.2.4.1. Режимы «2x» , «4x»

Если частота тактового сигнала стандартной шины PCI (32-разрядной) 33 МГц, благодаря чему ее теоретическая пропускная способность составляет 33×32 = 1056 Мбит/с = 132 Мбайт/с, то частота тактового сигнала шины AGP — 66 МГц, поэтому ее пропускная способность составляет уже 66×32 = 264 Мбайт/с (в режиме «1x»). Кроме режима «1x», стандартом AGP Revision 1.0 предусмотрен режим «2x», в котором передача данных производится не только по переднему, но и по заднему фронту тактового импульса. В режиме «2x» эквивалентная тактовая частота составит 132 МГц, а пропускная способность — 528 Мбайт/с.

Версией стандарта на шину AGP (AGP Revision 2.0) предусмотрено пониженное (1,5 В) напряжение питания и режим «4x», в котором за один такт передается уже не два, а четыре 32-разрядных слова. При этом пропускная способность шины достигает 1 Гбайт/с. Кроме того, наиболее современные видеоадаптеры, выполненные на основе прогрессивных технологий 0,25 мкм и 0,18 мкм, обычно устойчиво работают на еще более высоких частотах, чем 66 МГц. Так, например, Chipset материнской платы Intel 440BX позволяет повысить тактовую частоту системной шины до 133 МГц (вместо номинальной 100 МГц). При этом тактовая частота шины AGP, определяемая как 66% от частоты системной шины, составит не 66 МГц (при 100 МГц), а уже около 90МГц.

В конце 2000г. фирма Intel опубликовала проект спецификации AGP8x, которая рассматривается как отдельная спецификация, а не просто развитие предыдущих. Спецификацией предусмотрена пиковая производительность 2.132 Гбайт/с., установка нескольких портов AGP и ряд других особенностей.

3.2.4.2. Режим Pipelining

При обращении к памяти посредством шины ввода/вывода между моментом выставления на шине кода адреса и моментом получения данных проходит какое-то время (задержка). При обмене данными по шине PCI эта задержка возникает при каждом обращении к системной памяти (рис. 3.40, *верхняя диаграмма*).

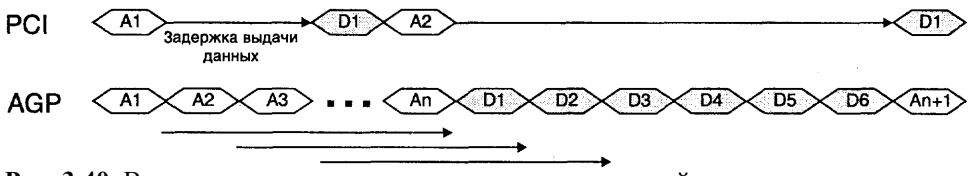


Рис. 3.40. Временные диаграммы, иллюстрирующие конвейерную передачу данных по шине AGP

Стандартом AGP, в отличие от PCI, предусмотрен режим *Pipelining — конвейерная (пакетная) передача данных,* в котором следующий код адреса выставляется на шине сразу же после предыдущего, т. е. коды адреса как бы выстраиваются в очередь, длина которой может достигать 256 адресов. Данные,

последовательность адресов которых была передана, пересылаются по шине также в виде пакета (рис. 3.40, *нижняя диаграмма*). В результате этого задержка в получении данных имеет место только один раз, благодаря чему значительно повышается скорость обмена данными.

3.2.4.3. Режим SBA

Как известно, шина PCI является *мультиплексированной (переключаемой),* т. е. одни и те же 32 линии AD0—AD31 используются сначала для передачи кода адреса, а затем — кода данных. Иногда такой режим называют *AD (Address—Data).* Мультиплексирование позволяет значительно (в два раза) сократить количество проводников, но при этом снижается пропускная способность шины. Шина AGP также может работать в режиме AD, но основным для нее является режим SBA (*Side-Band Addressing* — адресация по боковой полосе), в котором для передачи кода адреса используются восемь дополнительных линий шины SB0 - SB7.

Режим SBA используется только совместно с режимами «2x» и Pipelining. Для выполнения адресации в режиме SBA используются три такта. При этом по линиям SB0—SB7 передается шесть байт: за первые два такта — 4 байта адреса, а за третий — 1 байт длины пакета данных и 1 байт команды.

Линии AD0—AD31 шины в режиме SBA используются исключительно для передачи данных, поэтому пропускная способность шины в данном режиме значительно выше, чем в режиме AD.

5.2.4.4. Режим DiME

Видеоадаптер с интерфейсом AGP может работать с системной памятью в двух режимах: DMA и DiME (DME).

Традиционным является режим DMA (*Direct Memory Access* — прямой доступ к памяти), причем этот режим используется не только видеоадаптером, но и другими периферийными устройствами PC, например накопителем на жестких магнитных дисках, звуковой картой и др. Назначение режима DMA, понятное из его названия, — обеспечить прямой обмен данными между устройством и системной памятью, минуя регистры центрального процессора.

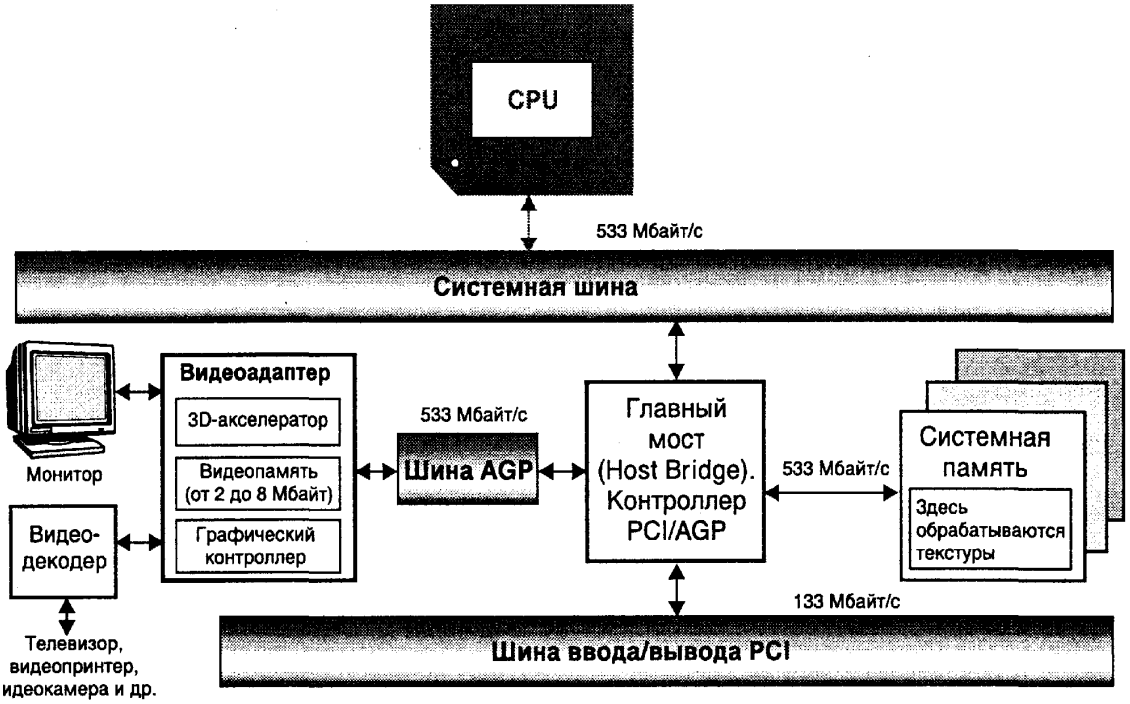


Рис. 3.41. Структура видеосистемы с интерфейсом AGP

Если 3D-акселератор работает в режиме DMA, основной для него является локальная память (именно в ней производятся все операции по обработке текстур), а системная память PC используется только в качестве «хранилища». Обмен данными по шине AGP в этом режиме выполняется большими последовательными пакетами и значительного выигрыша в скорости обмена данными, по сравнению с шиной PCI, достичь не удается.

В режиме *DME,* или *DiME* (*Direct Memory Execution* — непосредственное выполнение операций с текстурами в системной памяти), локальная и системная память являются для графического процессора 3D-акселератора равноценными. В результате в системной памяти можно выполнить предварительную обработку текстуры, а в локальную загрузить уже ее окончательный вариант. Это кардинальным образом меняет характер информационного обмена: в режиме DiME обмен данными выполняется в основном короткими пакетами. Именно в этом режиме значительно ускоряется выполнение операций с текстурами в

системной памяти и только в нем можно действительно ощутить все преимущества интерфейса AGP перед интерфейсом PCI.

В общем виде структура видеосистемы с интерфейсом AGP представлена на рис. 3.41

5.2.4.5. Слот AGP

Конструктивно слот AGP напоминает слот PCI, но контакты в нем расположены в 2 ряда, подобно тому, как это было сделано в слотах шины EISA или процессора Slot 1.

В зависимости от используемого напряжения питания различают 3 типа слотов AGP (рис. 3.42):

- слот, рассчитанный на 3,3 В;
- слот, рассчитанный на 1,5 В;
- универсальный слот (ключ-перемычка отсутствует).

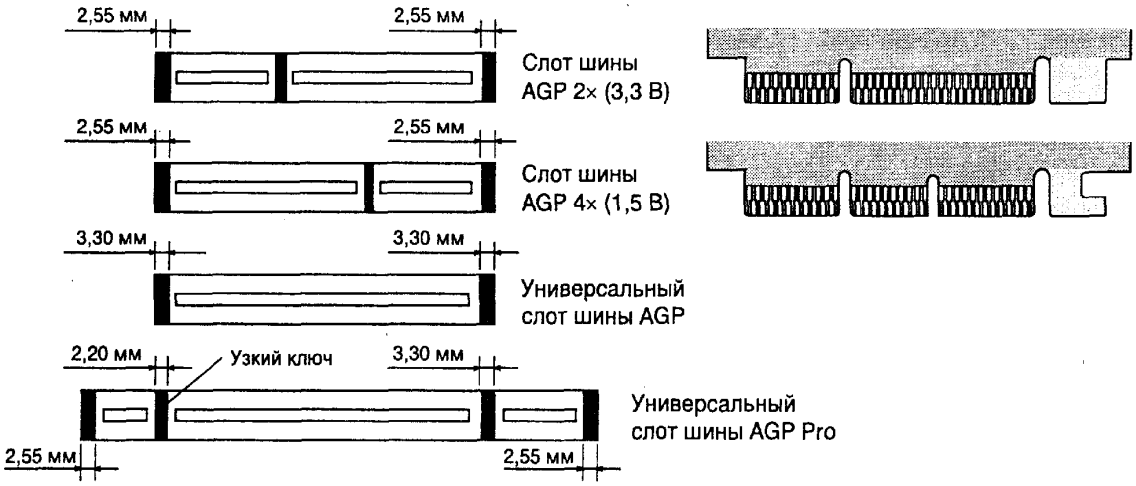


Рис. 3.42. Слоты AGP на материнской плате (слева) и разъемы видеоадаптеров с интерфейсом AGP (справа)

Поскольку в режиме «4x» используется пониженное напряжение питания 1,5В, то по количеству и расположению прорезей на разъеме видеоадаптера можно однозначно определить, поддерживает он этот режим или нет. Адаптеры, поддерживающие режим «4x», обычно устанавливаются в универсальный слот.

В 2002 году появились материнские платы со слотом AGP, поддерживающим напряжение питания 1,3В.

5.2.4.6. Преимущества и недостатки интерфейса AGP

Шина AGP в настоящее время де-факто стала стандартом для мультимедийного PC: все современные материнские платы и видеоадаптеры оборудованы интерфейсом AGP. Однако развитие видеосистемы PC с этим интерфейсом происходит не совсем так, как планировали специалисты фирмы Intel. Поначалу замена интерфейса видеоадаптера с PCI на AGP практически не повлияла на быстродействие видеосистемы. Это произошло по ряду причин.

Во-первых, интерфейс AGP разрабатывался исключительно для нужд 3D-акселерации. Сколько-нибудь заметного ускорения в работе традиционных 2D-приложений, например офисных, при замене видеоадаптера не происходит, поскольку:

- стандарт PCI тщательно отработан;
- для используемых в настоящее время видеоадаптеров с интерфейсом PCI имеются хорошо отлаженные драйверы;
- тактовая частота шины PCI в большинстве компьютеров на базе процессоров Pentium и Pentium II значительно превышает стандартное значение 33 МГц и может достигать 66 МГц, в результате пропускная способность этой шины сопоставима с пропускной способностью шины AGP;
- при работе с 2D-приложениями не используются ни текстуры, ни 3D-ядро видеоадаптера, поэтому преимущества шины AGP никак не реализуются.

Не случайно по сравнительным тестам на быстродействие в офисных приложениях видеоадаптеры с интерфейсом AGP (особенно видеоадаптеры первых моделей) и PCI практически не различаются.

Во-вторых, качество реализации интерфейса AGP может существенно отличаться, поскольку представленная фирмой Intel первая спецификация AGP является весьма нестрогой. По заявлениям представителей фирмы, AGP-совместимой платой считается любая плата, которую можно установить в слот AGP на материнской плате, при этом о реализуемых функциях не говорится ни слова, т. е. поддержка таких режимов как DiME, SBA и Pipelining, не является обязательной. В результате единственным преимуществом AGP-платы над ее PCI-модификацией может быть только более высокая тактовая частота ее интерфейса. Именно этим и отличались первые модели видеоадаптеров с интерфейсом AGP. Однако в настоящее время

эта «детская болезнь» практически полностью преодолена и выпускаемые большинством известных фирм материнские платы и видеоадаптеры поддерживают спецификацию AGP в полном объеме.

В-третьих, наметилась любопытная тенденция изменения размера локальной памяти видеоадаптеров с интерфейсом AGP. Казалось бы, объем памяти должен оставаться таким же небольшим, как у первых моделей 3D-акселераторов (не более 4—12 Мбайт), ведь именно ради этого и был введен стандарт AGP. Однако на практике быстродействие и объем видеопамяти стремительно растут: в настоящее время стандартным считается 16 или 32 Мбайт памяти типа SDRAM или SGRAM, И только благодаря снижению цен на память стоимость игровых 3D-акселераторов практически не изменилась. На платах последних моделей 3D-акселераторов фотореалистичной графики уже сегодня устанавливается память емкостью 64 - 128 Мбайт и более, причем не обычная, типа SDRAM, а более быстродействующая DDR SDRAM.

Благодаря этому актуальность AGP-текстурирования значительно снизилась. Дело в том, что пропускная способность шины локальной памяти видеоадаптера (за счет более высокой разрядности и тактовой частоты) в несколько раз выше, чем у шины AGP (даже в режиме «4x»). Если текстуры обрабатывать в системной памяти и загружать их по шине AGP, то скорость выполнения этапов 3D-конвейера значительно снизится. Чтобы не допустить этого, производители 3D-акселераторов вынуждены увеличивать объем локальной памяти. Именно поэтому на платах 3D-акселераторов фотореалистичной графики, работающих с текстурами большого размера, устанавливается не менее 32 Мбайт локальной памяти.

В-четвертых, следует помнить, что выделение части системной памяти для нужд 3D-акселераторов может привести к заметному снижению производительности всей системы, если в PC установлено менее 48 Мбайт памяти.

Таким образом, шина AGP является жизненно необходимой только в том случае, если PC предназначен для работы с 3D-графикой. В остальных случаях вполне достаточно возможностей интерфейса PCI.

Однако с появлением стандарта AGP шина PCI вовсе не торопится сдавать свои позиции. Во-первых, это разные по функциональному назначению шины: AGP — это локальная шина для видеосистемы, а PCI по-прежнему остается стандартной высокоскоростной шиной ввода/вывода. Во-вторых, частота тактового сигнала шины PCI уже сегодня составляет 66 МГц, а в ближайшем будущем планируется увеличить ее до 100 МГц. Кроме того, непрерывно совершенствуются алгоритмы протокола передачи данных по шине. Все это позволяет значительно повысить пропускную способность шины PCI и обеспечить достаточной скоростью внешние устройства всего PC и видеосистемы в частности. В пользу шины PCI говорит и такой факт: пионер в области игровых 3D-акселераторов, фирма 3Dfx, дольше всех выпускала свои изделия с интерфейсом PCI.

Однако следует признать, что будущее все-таки за видеоадаптерами с интерфейсом AGP. Если они по каким-либо причинам не намного превосходят по быстродействию своих «PCI-конкурентов», то, во всяком случае, не отстают от них.

3.2.4.7. Программно-аппаратная настройка видеосистемы с интерфейсом AGP

Для эффективной работы видеоадаптера с интерфейсом AGP необходимо соответствующее программное обеспечение.

Очевидно, что, во-первых, необходима материнская плата, поддерживающая стандарт AGP (на ней должны быть установлены соответствующий Chipset и слот). Первым Chipset (набором микросхем) фирмы Intel, поддерживающим AGP, был Chipset 440LX. В 2000 году самым распространенным и наиболее удачным считлся Chipset 440BX (как правило, он обеспечивал наибольшую скорость работы видеосистемы). Считается, что поддержка шины AGP в Chipset фирмы Intel реализована наиболее корректно и эффективно. Немного меньшее быстродействие видеосистемы с интерфейсом AGP обеспечивал Chipset тайваньской фирмы VIA Technologies (Apollo Pro, Apollo Pro 133), однако возможности каждой новой модификации этого Chipset все более расширяются и приближают его к Chipset фирмы Intel.

Во-вторых, необходимо убедиться в том, что материнская плата и видеоадаптер поддерживают одинаковые режимы работы, в частности, «2x» или «4x» (а в настоящее время и «8x»). Соответствующий режим работы материнской платы устанавливается с помощью программы CMOS Setup, видеоадаптера — обычно при помощи перемычек на его плате (сведения об этом, как правило, содержатся в техническом описании на видеоадаптер).

В-третьих, необходима операционная система, поддерживающая стандарт AGP. Можно использовать Windows 95 OSR 2.1, но лучше— Windows 98 (скорость и стабильность работы видеоадаптера с интерфейсом AGP под управлением Windows 98 несколько выше, чем под управлением Windows 95). В Windows 95 отсутствует встроенная поддержка стандартов AGP и USB. Поэтому после установки операционной системы необходимо дополнительно запустить программу USBSUP.EXE (поставляется в составе дистрибутива Windows 95 OSR 2.1), которая скопирует в каталог Windows\System файл VMM32.VxD, обеспечивающий поддержку AGP, и в частности режим DiME. Этот драйвер предназначен для управления работой шин USB и AGP.

И, наконец, необходим пакет библиотек API DirectX версии 5.0 и выше. Для Chipset фирмы VIA необходимо дополнительно установить драйверы из пакета VIA Service Pack, чаще называемого «четыре в

одном» (он содержит драйверы производителя для контроллеров AGP—PCI, IDE и расширенного управления питанием). Эти драйверы поставляются вместе с материнской платой, и приемлемую производительность и стабильность работы видеосистемы можно получить только при их использовании.

Теперь можно приступать к стандартной процедуре установки драйвера видеоадаптера.

В Windows 98 реализована встроенная поддержка шины AGP, поэтому какие-либо дополнительные драйверы устанавливать не требуется (к пакету VIA Service Pack это не относится, он по-прежнему нужен).

В июле 1998 г. фирма Intel объявила о дальнейшем развитии стандарта AGP, выпустив его новую спецификацию — AGP Pro. Шина AGP Pro имеет серьезные конструктивные отличия (см. рис. 3.42) от своей предшественницы:

- видоизменен разъем (слот);
- значительно повышена нагрузочная способность шины (в слот AGP Pro можно устанавливать видеоадаптеры, потребляемая мощность которых может достигать 110 Вт).

Интерфейс AGP Pro удовлетворяет требованию совместимости «сверху вниз», т. е. плату с интерфейсом AGP какой-либо прежней версии можно установить в слот AGP Pro, но не наоборот.

Контрольные вопросы.

1. Какие базовые элементы имеют все 3D-акселераторы?
2. Охарактеризуйте обобщенную структурную схему 3D-акселератора.
3. Какие этапы 3D-конвейера призван реализовывать геометрический процессор, и как он реализуется в различных 3D-ускорителях?
4. Какой механизм может использоваться для ускорения работы блока Setup Engine и какова его суть?
5. Какие два основных элемента содержит конвейер рендеринга, и с какими областями локальной памяти они работают?
6. Какие буферы, помимо кадрового и текстур, располагаются обычно в локальной памяти?
7. Какие микросхемы входят в состав Chipset Voodoo Graphics, каково их назначение и какова максимальная конфигурация платы 3D-акселератора, выполненного на их основе?
8. Какова возможная конфигурация использования чипсета Voodoo в 3D-ускорителе Obsidan 100SB, и какой режим работы она позволяет использовать?
9. Для чего в 3D-акселераторах используются два кадровых буфера?
10. Какие общие характеристики используются для оценки 3D-ускорителей?
11. Какие специфические показатели используются для оценки 3D-ускорителей?
12. Что характеризует первый специфический показатель 3D-ускорителя?
13. Что характеризует второй специфический показатель 3D-ускорителя и что понимается под масштабируемостью 3D-ускорителя?
14. Как изменяется значимость первого параметра при наличии у 3D-акселератора аппаратно реализованного блока Setup Engine?
15. Какие требования предъявляются к 3D-акселераторам по драйверам и API?
16. Какие характеристики изображения можно получить на 3D-акселераторах фотореалистичной графики?
17. Какова первая отличительная особенность видеоадаптера Radon 256, отличающая его от базовой структуры 3D-ускорителя?
18. Какова вторая отличительная особенность видеоадаптера Radon 256?
19. Каковы третья, четвертая и пятая отличительные особенности видеоадаптера Radon 256?
20. Какова шестая отличительная особенность видеоадаптера Radon 256?
21. Каковы седьмая, восьмая и девятая отличительные особенности видеоадаптера Radon 256?
22. Почему архитектура UMA плохо подходит для обработки 3D-графики?
23. Кем и для чего была разработана коростная шина ввода/вывода AGP?
24. Чем отличается AGP от PCI?
25. Какова тактовая частота шины AGP и ее пропускная способность в режимах "1x" и "2x"?
26. Какова отличительная особенность AGP Revision 2.0?
27. В чем суть режима Pipelining?
28. В чем суть режима SBA?
29. В каких режимах взаимодействия с системной памятью может работать видеоадаптер AGP?
30. В чем особенность взаимодействия 3D-акселератора с системной памятью в режиме DMA?
31. В чем особенность режима DME (DiME)?
32. Охарактеризуйте общий вид структуры видеосистемы с интерфейсом AGP.
33. Каковы разновидности слотов AGP?
34. Для чего разрабатывался интерфейс AGP и почему первые видеоадаптеры AGP не обеспечивали сколь-нибудь заметного ускорения в работе 2D-приложений?
35. Почему первоначальная замена интерфейса с PCI на AGP практически не повлияла на быстродействие видеосистемы?

36. Какова тенденция изменения локальной памяти видеоадаптеров с интерфейсом AGP?
37. Дайте сравнительную характеристику шины PCI.
38. Какой чипсет стал первым поддерживать шину AGP и какой чипсет являлся наиболее оптимальным для AGP в начале этого века?
39. Каковы особенность поддержки AGP и USB ОС Windows-95 и Windows-98?
40. В чем особенность спецификации AGP Pro?

В данном разделе использованы материалы из [1, 8, 9]. Более подробную информацию по AGP можно найти в [9].

3.3. Chipset для 3D-акселераторов

В настоящее время рынок 3D-акселераторов для PC испытывает невероятный бум. Лидеров в этой области пока не много, но ассортимент производимых ими изделий так велик и многообразен, что выбрать конкретную модель, наиболее подходящую для решения стоящих перед вами задач, становится очень сложно. Кроме того, крайне агрессивная реклама, которой сопровождается появление каждой новинки, повергает неискушенного пользователя в уныние.

Чтобы как-то помочь пользователю грамотно подойти к решению проблемы выбора конкретной модели 3D-акселератора, приводится краткий обзор наиболее популярных наборов микросхем (Chipset), на основе которых выполняются акселераторы. Именно функциональными возможностями Chipset определяются потенциальные возможности всей платы 3D-акселератора. Поскольку в настоящее время чипсет для видеоплат выпускаются в основном в одном корпусе (на одном кристалле), то часто эту микросхему называют графическим процессором, подчеркивая ее основную особенность - реализация на основе ее ресурсов 2D и 3D геометрических и графических функций непосредственно "на борту" видеоплаты.

3.3.1. Chipset фирмы 3Dfx Interactive

Как уже отмечалось, фирма 3Dfx Interactive выпустила первый удачный 3D-акселератор — Chipset Voodoo Graphics. Успех 3Dfx не является случайным. Прежде она специализировалась на производстве игровых приставок, профессиональных симуляторов и тренажеров. Главное внимание при разработке своих изделий фирма уделяет их производительности, поэтому ее главный лозунг: «60 кадров в секунду любой ценой!». За последние несколько лет фирмой разработаны Chipset Voodoo^2, Voodoo Rush, Voodoo Banshee, Voodoo^3, модели: Voodoo^4 и Voodoo^5 и другие.

3.3.1.1. Voodoo Graphics

Voodoo Graphics является классическим вариантом 3D-акселератора, выполненного в виде отдельной платы с интерфейсом PCI. Помимо нее, в системе должен быть установлен обычный видеоадаптер. В настоящее время этот пионер 3D безнадежно устарел: обеспечивая приемлемую скорость работы при разрешении 640×480, он не обеспечивает приемлемой частоты кадров при более высоких разрешениях. Поэтому на плате редко устанавливают больше 4 Мбайт видеопамяти для кадрового буфера. Основные характеристики Chipset Voodoo Graphics:

- тактовая частота — до 60 МГц (управляется программно);
- максимальная частота RAMDAC — 135 МГц;
- тип локальной памяти — EDO (время доступа 30-40 нс);
- размер кадрового буфера — до 4 Мбайт (обычно 2 Мбайт);
- размер буфера текстур — до 8 Мбайт (обычно 2-4 Мбайт);
- максимальное разрешение:
  -800×600 (кадровый буфер размером 4 Мбайт);
  -640×480 (кадровый буфер размером 2 Мбайт);
  -1024×768 (кадровый буфер размером 4 Мбайт в режиме SLI; только для платы Obsidian 100SB);
- глубина цвета — 16 бит (High Color);
- максимальное значение параметра Fill Rate — 50 млн пикселов в секунду;
- максимальное значение параметра Triangle Throughput — 500 тыс. треугольников в секунду;
- типичная производительность в игре Quake — 30—35 кадр/с при разрешении 640×480 и CPU Pentium 166MMX;
- поддерживаемые API: DirectX, Glide (API разработки 3Dfx), частично OpenGL;
- работает только в полноэкранном режиме;
- является специализированной платой, выполняющей только 3D-функции, поэтому требует наличия в системе обычного видеоадаптера с 2D-ускорителем;
- соединяется с видеоадаптером специальным кабелем типа Pass-Through (возможно ухудшение качества изображения при работе с высоким разрешением).

Наиболее известными платами на основе Chipset Voodoo Graphics являются Diamond Monster 3D, 3D Canopus Pure3D, 3D Voodoo Biostar Venus, A-Trend Helios, Cardex Dragon 1000, miroMEDIA miroHISCORE 3D, RealVision Flash3D, Skyweli Magic 3D и др.

3.3.1.2. Voodoo Rush

Стремление фирмы 3Dfx создать интегрированную 2D/3D-плату, а также устранить искажения, вносимые кабелем типа Pass-Through, привело к появлению Chipset Voodoo Rush. Этот Chipset предназначен для 2D- и 3D-акселерации и включает в себя два чипа (микросхемы): PCfx и Texelfx. Наличие в составе Chipset специального видеоконтроллера, работающего непосредственно с кадровым буфером, позволило реализовать оконный режим работы. К сожалению, реальная производительность Chipset оказалась ниже проектной и даже ниже, чем Voodoo Graphics. В частности, Voodoo Rush не позволяет одновременно

выполнять z-буферизацию и альфа-смешение, что значительно снижает быстродействие. В целом этот Chipset имеет аналогичные с Voodoo Graphics характеристики, но неустойчив в работе, поэтому считается далеко не самой удачной разработкой 3Dfx.

Примеры видеоадаптеров на основе Voodoo Rush: A-Trend Helios Voodoo Rush, Biostar Venus 3D Rush, Hercules Stingray 128/3D, Jazz Adrenaline Rush 3D, RealVision Flash AT3D Rush и др.

3.3.1.3. Voodoo^2

Chipset Voodoo^2 явился принципиально новым изделием, имеющим гораздо более высокие характеристики, чем у его предшественников. Также как и Voodoo Graphics, он является специализированным 3D-акселератором с интерфейсом PCI, построенным на новых чипах: одном Pixelfx^2 и двух Texelfx^2. Каждый из трех чипов имеет свой модуль памяти: 4 Мбайт для кадрового буфера и 2 или 4 Мбайт для буфера текстур. В результате возможны две конфигурации памяти: 4 + 2×2 = 8 Мбайт и 4 + 2×4 = 12 Мбайт.

Наличие второго чипа и буфера памяти для обработки текстур стало ключевым моментом, обеспечившим этому Chipset аппаратную поддержку мультитекстурирования: позволяет одновременно либо закрашивать два пиксела, либо накладывать сразу два тексела.

Помимо прироста в скорости работы, наличие второго блока текстурирования позволило аппаратно реализовать трилинейную фильтрацию. Новой функцией явилась также аппаратная поддержка рельефного текстурирования. Для устранения дефектов наложения текстур используется краевой антиалиасинг.

За счет аппаратной реализации в Voodoo^2 механизма Triangle Setup Engine значительно повысилась скорость выполнения этапов стадии геометрической обработки 3D-конвейера, а значение параметра Triangle Throughput выросло по сравнению с Voodoo Graphics в 5 раз!

Как и его предшественник Voodoo Graphics, Chipset Voodoo^2 поддерживает только 16-битный режим High Color.

В течение довольно длительного времени 3D-акселераторы на основе Voodoo^2 по производительности являлись лидерами рынка.

Наличие в Voodoo^2 эффективного конвейера рендеринга привело к тому, что узким местом всего 3D-конвейера стала геометрическая стадия, реализуемая центральным процессором (за исключением этапа Triangle Setup). Оптимальным CPU для Voodoo^2 является процессор не хуже Pentium II с тактовой частотой не ниже 300 МГц.

Платой за высокую производительность данного Chipset является интенсивное выделение тепла (больше всего нагреваются микросхемы Texelfx^2), поэтому при использовании 3D-акселератора на основе Voodoo^2 необходимо предусмотреть дополнительное принудительное охлаждение. К сожалению, лишь немногие платы оборудуются вентилятором (такие, как, например, Canopus Pure3D II).

Основные характеристики Voodoo^2:

- максимальная тактовая частота ядра — 100 МГц (у Voodoo Graphics 50-60 МГц);
- максимальное разрешение:
  -800×600 с z-буферизацией;
  -1024×768 без z-буферизации;
  -1024×768 с z-буферизацией в режиме SLI, при наличии второй платы;
- максимальное значение параметра Fill Rate — 90 млн пикселов в секунду (или 180 млн.пикселов в секунду в режиме SLI);
- максимальное значение параметра Triangle Throughput — 2,5 млн треугольников в секунду;
- типичная производительность в игре Quake — 70 кадр/с при разрешении 640×480 и использовании CPU Pentium II с тактовой частотой 300 МГц.

Примеры 3D-акселераторов на основе Chipset Voodoo^2: Canopus Pure3D II, A-Trend Helios 3D Voodoo II, BiostarVenus 3D II, Creative 3D Blaster Voodoo^2, Diamond Monster 3D II, STB BlackMagic 3D, Quantum3D Obsidian2 X-24.

Последняя плата продолжает традицию своего предшественника Obsidian 100SB и фактически содержит два экземпляра Chipset Voodoo^2, причем на плате установлено 12 Мбайт видеопамяти. При этом работа в режиме SLI обеспечивается, однако не решена проблема теплоотвода. На плате не установлен вентилятор и нет возможности установить его самостоятельно. Очевидно, что и стоимость этого изделия достаточно высока. Дешевле будет приобрести две одинаковые платы 3D-акселератора на основе Chipset Voodoo^2, оборудованные 8-мегабайтной памятью каждая.

3.3.1.4. Voodoo Banshee

Потерпев неудачу с Voodoo Rush, компания 3Dfx тем не менее не отказалась от самой идеи сделать полнофункциональный 2D/3D-акселератор на одной плате. Взяв за основу 3D-ядра элемента Chipset Voodoo^2 и разработав с нуля 2D-акселератор, 3Dfx создала весьма удачный Chipset, объединивший в себе все лучшие черты своих предшественников, и назвала его Voodoo Banshee. Этот Chipset выполняет все необходимые функции: 2D/3D-акселерацию и обработку видео. Он также содержит RAMDAC с тактовой частотой 250

МГц.

3D-ядро Chipset Voodoo Banshee такое же, как и у Voodoo^2, за исключением того, что вместо двух чипов Texelfx в Voodoo Banshee используется только один, что не позволяет аппаратно реализовать мультитекстурирование и трилинейную фильтрацию. Кроме того, Chipset Voodoo Banshee не поддерживает режим SLI.

Однако по остальным характеристикам этот Chipset намного опережает Voodoo^2. Наиболее важные из них связаны с иной архитектурой локальной памяти. Если в Voodoo Graphics и Voodoo^2 использовалась 64-разрядная память типа EDO с тактовыми частотами 60 и 90 МГц соответственно, то Voodoo Banshee оснащен более современной и быстродействующей 128-разрядной памятью типа SGRAM или более дешевой SDRAM с тактовой частотой 100 МГц.

Кроме того, Chipset поддерживает архитектуру UMA, в соответствии с которой элементы 3D-изображения могут храниться не только в локальной памяти акселератора, но и в системной памяти компьютера. Для реализации такого подхода необходима поддержка стандарта AGP. Поэтому платы на основе Voodoo Banshee выпускаются в двух модификациях: с интерфейсом PCI (тактовая частота 33 МГц) или AGP (тактовая частота 66 МГц). Это лишний раз доказывает, что поддержка стандарта AGP в изделиях 3Dfx носит формальный характер, поскольку производительность 3D-акселераторов на основе Chipset этой фирмы незначительно зависит от типа используемого интерфейса.

На плате 3D-акселераторов на основе Chipset Voodoo Banshee, как правило, устанавливается 4 — 16 Мбайт памяти типа SGRAM и только 16 Мбайт более дешевой памяти типа SDRAM. Это также подтверждает тезис о незначительной зависимости быстродействия Chipset от типа используемого интерфейса.

3.3.1.5. Voodoo^3

Выпустив Chipset Voodoo^3, компания 3Dfx окончательно отказалась от выпуска специализированных 3D-ускорителей и перешла на производство универсальных видеоадаптеров, содержащих 2D- и 3D-ядро в одном чипе. По этой причине Chipset Voodoo^3 очень напоминает Voodoo Banshee, но превосходит его почти по всем показателям (из-за этого некоторые острословы именовали Chipset Voodoo^3 не иначе как «Voodoo Banshee-2»).

Chipset Voodoo^3 имеет три модификации: Voodoo^3 2000, Voodoo^3 3000 и Voodoo^3 3500, различающиеся тактовыми частотами RAMDAC и ядра, а также набором дополнительных функций (функция TV-out, возможность подключения плоскопанельного монитора и т. п.). Использование 0,25-микрометровой технологии позволило получить достаточно высокие тактовые частоты ядра — от 143 до 183 МГц.

В качестве локальной используется память типа SDRAM или SGRAM емкостью 4-16 Мбайт. Chipset имеет интерфейс AGP с поддержкой режима «2x», однако не позволяет обрабатывать текстуры в системной памяти PC (отсутствует поддержка режима DiME) — фирма 3Dfx вновь отличилась неудачной реализацией интерфейса AGP. Отсутствие AGP-текстурирования в совокупности с небольшим объемом локальной памяти (не более 16 Мбайт) ограничило максимальный размер используемых текстур (не более 256×256).

Производители учли недостатки Voodoo Banshee и добавили в Voodoo^3 второй блок текстурирования, благодаря чему у него появилась аппаратная поддержка мультитекстурирования, трилинейной и анизотропной фильтрации. Кроме того, Voodoo^3 поддерживает краевой и субпиксельный антиалиасинг, затенение Гуро, коррекцию перспективы, технологию strips&fans, табличный туман, до 14 форматов текстур, включая сжатые текстуры. Однако Chipset сохранил и характерный недостаток изделий производства компании 3Dfx: при работе с 3D-графикой поддерживается только режим High Color, т. е. 16-битное кодирование цвета. Это частично компенсируется очень высоким разрешением, которое он может обеспечить, — до 2048×1536 при частоте кадров 75 Гц (тактовая частота RAMDAC составляет 300 или 350 МГц).

Chipset Voodoo^3 является, безусловно, неплохим изделием, особенно с точки зрения быстродействия, однако безоговорочным лидером, как в свое время Voodoo^2, он не стал — Chipset семейства Riva TNT2 имеет аналогичные (и даже лучшие) показатели при значительно более низкой цене.

Нельзя также не отметить еще одну характерную слабость Chipset производства фирмы 3Dfx— неэффективную работу под управлением API OpenGL. Правда, этот недостаток частично компенсируется достаточно хорошей совместимостью с системой команд 3Dnow!, которая используется процессорами AMD K6-II и выше.

3.3.2. Chipset фирмы Intel

Знаменитая фирма, ранее производившая в основном только центральные процессоры и Chipset для материнских плат, удивила всех стремительным взлетом на рынке аппаратных средств для ускорения 3D-графики, выпустив достаточно мощный и при этом весьма недорогой графический процессор под названием i740. Это стало заметным событием, поскольку видеоадаптер на основе данного Chipset рассматривается фирмой Intel как базовый комплектующий элемент для компьютеров с процессорами Pentium II.

3.3.2.1. i740

Chipset i740 является 64-разрядным совмещенным 2D/3D-акселератором, реализующим несколько

дополнительных функций: TV-out, возможность ввода/вывода видеосигналов, декодирование данных формата DVD. Chipset предназначен для использования на компьютерах с процессором Pentium II и шиной AGP, хотя имеются и PCI-модификации видеоадаптеров на его основе.

Забегая вперед и не вдаваясь в детали, можно коротко охарактеризовать данный Chipset следующим образом: он полностью соответствует традиции фирмы Intel выпускать добротные изделия. При средней производительности (на уровне Voodoo Graphics) и рекордно низкой для 3D-акселераторов цене Chipset обеспечивает очень высокое качество изображения, поэтому некоторое время он служил оптимальным решением для домашнего компьютера начального уровня.

Поскольку фирма Intel является автором стандарта AGP, реализация его поддержки в i740 не вызывает нареканий: поддерживаются режимы «2x», SBA, и DiME.

Chipset работает с локальной памятью типа SDRAM или SGRAM емкостью от 2 до 8 Мбайт и с тактовой частотой 100МГц. Видеоадаптеры на основе Chipset i740 первоначально выпускались в трех модификациях: с 2 (*Mainstream* — основной уровень), с 4 (*Performance* — производительный) и с 8 (*PC Theater*— видеотеатр на основе PC) Мбайт видеопамяти. Позже на плате видеоадаптеров на основе Chipset i740 устанавливали 8 Мбайт памяти.

Chipset содержит RAMDAC, работающий на тактовой частоте 205 МГц при палитре 256 цветов и на частоте 220 МГц в режимах High Color и True Color.

Chipset i740 реализована аппаратная поддержка следующих 3D-функций:

- наложение текстур в реальном масштабе времени, двойная буферизация;
- 16-разрядный z-буфер;
- билинейная фильтрация при реализации MIP-текстурирования;
- поддержка 11 уровней детализации текстур с разрешением до 1024×1024;
- программная эмуляция трилинейной фильтрации;
- коррекция перспективы при текстурировании;
- возможность хранения текстур в нескольких форматах:
  - оптимизированном *(palletised)* 1-, 2-, 4- и 8-битном;
  - RGB 5:6:5, RGBα 1:5:5:5 и RGBα 4:4:4:4;
  - в сжатых YUV 4:2:2 , YUV 5:5:5 и YUV 1:5:4:4;
- отображение 3D-изображений в окне, на полном экране и в нескольких окнах;
- значение параметра Triangle Throughput — 450—500 тыс. треугольников в секунду;
- скорость закраски (Fill Rate) — 45—55 млн пикселов в секунду;
- типичная производительность в игре Quake — около 30 кадр/с.

Помимо стандартных, Chipset i740 выполняет ряд дополнительных 3D-функций:

- точная интерполяция пикселов по технологии PPI *(Precise Pixel Interpolation);* используется при коррекции перспективы и субпиксельном антиалиасинге;
- параллельная обработка потоков данных (технология PDP — *Parallel Data Processing).*

Chipset имеет ряд дополнительных возможностей по обработке видеоданных в формате MPEG-2, т. е. данных, считываемых DVD-приводом, а также по вводу/выводу видеосигналов. При этом в более дешевых модификациях Chipset i740 реализуется только интерфейс для подключения дочерних карт, аппаратно реализующих перечисленные функции.

Помимо указанных возможностей, видеоадаптер на основе i740 обеспечивает поддержку нескольких мониторов.

Примеры видеоадаптеров на основе Chipset i740: Real3D StarFighter, A-Trend Speedy 3DI0740, A-openPA740, Aristo ART- i740-G, ASUS3DexPlorerV2740, Biostar Orion 740, Cardex Cardexpert I740, Diamond Stealth II G460, ChainTech Desperado, Intel Express 3D, STB Lightspeed 740, Tekram AGP-6000 3D Fire.

3.3.3. Chipset фирмы nVidia

Компания nVidia получила широкую известность благодаря Chipset Riva 128, представляющему собой совмещенный 2D/3D-акселератор. В отличие от всех рассмотренных выше 64-разрядных Chipset, Riva 128 и его модификации имеют 128-разрядную шину локальной памяти. В настоящее время фирма является признанным лидером на рынке 3D- акселераторов, поскольку ее продукция отличается высоким качеством, наилучшей производительностью и разумной ценой.

Наиболее известными продуктами nVidia являются Chipset Riva 128, Riva I28ZX, Riva TNT, Riva TNT2 (включая их многочисленные модификации), а также относительно новые ускорители фотореалистичной графики GeForce 256 (NV10) и GeForce 2 GTS (NV15), оборудованные геометрическими процессорами.

3.3.3.1. Riva 128

Платы на основе Chipset Riva 128 предоставляют пользователю полный набор функций по 2D/3D-ускорению и вводу/выводу видеосигнала. Chipset выпускается как с интерфейсом PCI, так и с интерфейсом AGP. Объем локальной памяти типа SGRAM, предназначенной для организации кадрового и z-буферов,

постоянный — 4 Мбайт. Текстуры хранятся в системной памяти компьютера. Для ускорения их подкачки в локальную память используется кэш-память емкостью 12 Кбайт.

Для 3D-акселераторов на основе Chipset Riva 128 характерна сильная зависимость их производительности от быстродействия CPU и пропускной способности шины ввода/вывода. По этой причине быстродействие плат с интерфейсом PCI может быть значительно ниже, чем с интерфейсом AGP, особенно если пропускная способность шины окажется недостаточной. RAMDAC этого Chipset работает на тактовой частоте 230 МГц. При работе с 3D-графикой Chipset обеспечивает 16-битный цвет, с 2D-графикой — 32-битный.

Имеется возможность ввода/вывода видеосигнала, однако программное обеспечение для поддержки этой функций оставляет желать лучшего. Ввод видеосигнала происходит медленно, так как реализуется программно и требует участия центрального процессора.

Основная особенность плат на основе Chipset Riva 128 — сочетание достаточно высокой производительности с весьма посредственным качеством изображения. Такие дефекты, как нестыковка текстур, темные полосы и др., характерные для ускорителей на основе Riva 128, стали классическим примером дефектов изображения при реализации 3D-конвейера, поэтому данный Chipset не пользовался популярностью.

Chipset Riva 128 имеет следующие характеристики, определяющие его быстродействие:

- предельное значение параметра Triangle Throughput — до 5 млн треугольников в секунду;
- скорость закраски (Fill Rate) — до 100 млн пикселов в секунду;
- максимальное разрешение при использовании 16-разрядного z-буфера — 960×712;
- типичная производительность в игре Quake (с процессором Pentium-200MMX) - 30 кадр/с.

Для работы плат на основе Chipset Riva 128 требуется не менее 32 Мбайт системной оперативной памяти. Поставляемые вместе с ними драйверы обеспечивают полную поддержку API OpenGL (на уровне ICD). Заметим, что хорошая поддержка интерфейса AGP и качественные драйверы для работы под управлением OpenGL — характерная черта изделий производства компании nVidia.

Примеры видеоадаптеров на основе Chipset Riva 128: Asus 3DexPlorer 3000, CanopusTotal3DV128, Diamond Viper V330, ELSA Victory Erazor, STB Velocity 128, Tekram AGP 5000.

3.3.3.2. Riva 128ZX

Chipset Riva 128ZX компания nVidia разрабатывала с целью устранить недостатки предыдущей модели, однако полностью решить проблему ей не удалось. Основным отличием Chipset Riva 128ZX является увеличенный вдвое объем видеопамяти типа SGRAM — 8 Мбайт, а также более высокое качество 3D-изображения. Максимальное разрешение увеличено до 1280×1024. Тактовая частота RAMDAC повышена до 250 МГц. Улучшено качество обработки видеосигнала (в частности, реализована фильтрация телевизионных бликов — *TV flicker filtering).*

Пример платы на основе Riva 128ZX: Cardex Cardexpert 128ZX.

3.3.3.3. Riva 128TNT (Riva TNT)

Chipset Riva 128 TNT стал, пожалуй, первым по-настоящему удачным изделием nVidia, позволившим ей отобрать пальму первенства у компании 3Dfx. Не случайно этому Chipset (в дальнейшем его стали называть просто Riva TNT) отводилась роль «киллера» знаменитого Voodoo[2], что отчасти и произошло.

Как следует из названия, Chipset Riva TNT (TwiN Texel) имеет два процессора для обработки текстур, благодаря чему обеспечивается аппаратное ускорение мультитекстурирования, трилинейной и анизотропной фильтрации. Ядро Chipset выполнено по технологии 0,35 мкм и работает на частоте 90 МГц На плате видеоадаптера размещается видеопамять типа SGRAM или SDRAM емкостью 8 или 16 Мбайт, работающая на частоте 125 и 110 МГц соответственно. Встроенный RAMDAC работает на частоте 250 МГц.

Chipset Riva TNT допускает использование двух интерфейсов — PCI и AGP. Как обычно, качественная реализация AGP — сильная сторона Chipset фирмы nVidia: реализованы все возможности, предусмотренные стандартом AGP.

Наряду со стандартными функциями 3D-акселерации, Chipset обеспечивает:

- 32-битный цвет в 3D-приложениях;
- максимальное разрешение в 3D-приложениях при 32-битном цвете и частоте кадров 85 кадр/с — 1600×1200;
- 24-битный z-буфер, допускающий использование чисел с фиксированной и плавающей точкой (большинство 3D-акселераторов имеют 16-битный z-буфер);
- аппаратную поддержку анизотропной фильтрации;
- рельефное текстурирование (bump mapping);
- полный антиалиасинг (FSAA)

Важным преимуществом Chipset Riva TNT, по сравнению с конкурентами, явилась его высокая производительность:

- пиковая пропускная способность (Triangle Throughput) — до 8 млн треугольников в секунду;
- максимальная скорость закраски (Fill Rate) — до 250 млн пикселов в секунду и до 125 млн — в режиме мультитекстурирования;
- производительность в игре Quake II при разрешении 1600×1200 в режиме True Color (на PC с процессором Pentium II 400 МГц) — 30 кадр/с.

Chipset имеет также дополнительные возможности по обработке видеоданных:

- аппаратная акселерация воспроизведения видеосюжетов с использованием кодеков MPEG-1, Indeo и частично MPEG-2;
- наличие выхода TV-out в стандартах NTSC и PAL (в зависимости от комплектации платы).

Примеры видеоадаптеров на основе Riva TNT: Diamond Viper V550, STB Velocity 4400.

3.3.3.4. Chipset семейства Riva TNT2

Стремясь закрепить очевидный успех, достигнутый за счет выпуска Chipset Riva TNT, компания nVidia вскоре выпустила его усовершенствованную версию — Riva TNT2. Это было ответным шагом на выпуск Chipset Voodoo[3] фирмой 3Dfx.

В основном Chipset Riva TNT2 сохранил архитектуру своего предшественника Riva TNT. Главными отличиями новой модели являются:

- использование новой, 0,25-микрометровой технологии при изготовлении ядра;
- увеличенный с 16 до 32 Мбайт объем видеопамяти;
- использование RAMDAC с тактовой частотой 300 МГц;
- поддержка режима «4x» при AGP-текстурировании.

Все это позволило значительно повысить тактовые частоты ядра (с 90 до 125 МГц) и видеопамяти (со 110 до 150МГц). Конвейер рендеринга был также усовершенствован, поэтому при одинаковых тактовых частотах производительность Chipset Riva TNT2 примерно на 10—15% выше, чем у Riva TNT. Благодаря увеличению объема видеопамяти, появилась возможность использовать текстуры более крупных размеров, не задействуя при этом шину ввода/вывода. Это обеспечило лучшую детализацию 3D-сцены при сохранении высокой производительности.

Видеоадаптеры на основе Chipset Riva TNT2 образуют достаточно большое семейство, представители которого значительно различаются по производительности и цене. Главными отличительными признаками их являются тип, объем и быстродействие установленной на плате видеопамяти. В настоящее время на рынке, помимо базовой модели, имеются следующие представители семейства TNT2:

- Riva TNT2-A;
- Riva TNT2 Ultra;
- Riva TNT2 Vanta;
- Riva TNT2 M64.

3.3.3.4.1. Riva TNT2-A

Chipset Riva TNT2-A является полным аналогом базовой модели за исключением того, что он выполнен по технологии 0,22 мкм вместо 0,25. Это позволило повысить стандартную тактовую частоту ядра до 143 МГц вместо 125, как у Riva TNT2. Практика показывает, что при наличии быстродействующей видеопамяти тактовую частоту ядра и видеопамяти можно увеличить до 180 МГц.

3.3.3.4.2. Riva TNT2 Ultra

Chipset Riva TNT2 Ultra является наиболее быстродействующей (и дорогостоящей) моделью: стандартная тактовая частота ядра составляет 150 МГц, а памяти — 183 МГц. Ядро Chipset выполнено также по технологии 0,22 мкм. Как правило, на плате видеоадаптера на основе Riva TNT2 Ultra устанавливается 32 Мбайт видеопамяти типа SDRAM или SGRAM со временем доступа не более 7 нс.

Наиболее характерная модель видеоадаптера на основе данного Chipset — Hercules Dynamite TNT2 Ultra, оборудованная видеопамятью типа SDRAM емкостью 32 Мбайт с временем доступа 5,5 нс. Благодаря этому тактовая частота ядра составила 175 МГц, памяти — до 200 МГц.

Видеоадаптеры на основе Chipset Riva TNT2-A и особенно Riva TNT2 Ultra являлись дорогостоящими (от 110 до 170 USD) и недоступными широкому кругу пользователей изделиями. Поэтому фирма nVidia, стремясь расширить свое господствующее положение на рынке, выпустила упрощенные и гораздо более дешевые модификации Chipset TNT2, имеющие 64-разрядную шину памяти. К ним относятся две модели: TNT2 Vanta и TNT2 M64, каждая из которых поддерживает 8, 16 либо 32 Мбайт видеопамяти типа SDRAM.

3.3.3.4.3. Riva TNT2 Vanta

Наиболее дешевым является Chipset Riva TNT2 Vanta, RAMDAC которого работает на той же частоте, что и RAMDAC Riva TNT, т. е. 250 МГц. Первоначально Chipset TNT2 Vanta предназначался для

портативных компьютеров, у которых видеоадаптер устанавливался на материнскую плату, однако низкая цена этого Chipset стимулировала производителей к выпуску на его основе самостоятельной платы. В большинстве случаев видеоадаптер на основе TNT2 Vanta оборудован всего 8 Мбайт видеопамяти.

Более высокие тактовые частоты ядра/памяти у TNT2 Vanta (100/125 МГц), по сравнению с RivaTNT (90/110 МГц), компенсируются вдвое меньшей разрядностью шины памяти (64 разрядов вместо 128), поэтому новый Chipset не смог опередить по производительности своего предшественника. Более того, при высоких разрешениях в 2D-приложениях и при работе с 3D-графикой, когда пропускная способность шины видеопамяти особенно критична, TNT2 Vanta заметно уступает по быстродействию Chipset Riva TNT (этим объясняется то странное обстоятельство, что более новая модель TNT2 Vanta стоит дешевле, чем Riva TNT). Не спасает положения и предусмотренная на многих видеоадаптерах на основе TNT2 Vanta поддержка режима AGP «4x».

3.3.3.4.4. Riva TNT2 M64

Chipset TNT2 M64 представляет собой «разогнанный» вариант TNT2 Vanta: по умолчанию используются тактовые частоты ядра/памяти 125/150 МГц, однако при наличии внешнего охлаждения они могут быть увеличены до 140/183 МГц. Тактовая частота RAMDAC также увеличена до 300 МГц. Благодаря этим мерам, производительность Chipset стала выше, чем производительность Riva TNT, но не достигла быстродействия Riva TNT2. Как и в случае с Riva TNT2 Vanta, отставание наиболее сильно проявляется при работе в режимах с высоким разрешением.

3.3.3.5. Chipset семейства GeForce

Одним из последних достижений компании nVidia в области 3D-акселерации стало семейство мощных ускорителей с геометрическим процессором, обеспечивающим аппаратное ускорение наиболее трудоемких этапов геометрической стадии 3D-конвейера: преобразование координат вершин граней и расчет освещенности, особенно при наличии нескольких источников освещения. Набор ускоряемых функций реализован узлом *HWT&L (HardWare Transform and Lighting*— аппаратное преобразование координат и расчет освещенности). Часто используется упрощенное обозначение: T&L. Перечень функций геометрического процессора T&L аналогичен приведенному в разд. 3.2.1. Благодаря наличию этого узла, значение показателя Triangle Throughput у GeForce 256 достигает 15 млн треугольников в секунду. Среди других важных отличий Chipset этого семейства можно назвать следующие:

- выполнен по технологиям 0,22 и 0,18 мкм, что обеспечило дальнейший рост тактовой частоты ядра;
- разрядность ядра и шины памяти увеличена до 256;
- использованы новые типы памяти, например DDR SDRAM, что, наряду с увеличенной разрядностью, позволяет радикально повысить пропускную способность видеопамяти;
- объем видеопамяти может достигать 128 Мбайт;
- значительно повышена мощность модуля рендеринга: он включает в себя 4 конвейера, каждый из которых имеет один или два блока текстурирования.

Все эти особенности позволяют отнести видеоадаптеры на основе Chipset GeForce 256 к новому *типу ускорителей фотореалистичной графики.*

3.3.3.5.1. GeForce 256

Первой моделью семейства стал GeForce 256, именуемый также NV10 (рис. 3.43, слева). Хотя он выполнен по технологии 0,22 мкм, стандартная тактовая частота ядра невелика — 120 МГц, а памяти — до 200 МГц. RAMDAC работает на частоте 350 МГц, что позволяет получить максимальное разрешение 2048×1536 при частоте кадров 75 кадр/с. Помимо стандартных режимов AGP, поддерживается режим 4x Fast Writes, при котором обмен данными между центральным процессором и ядром Chipset ведется напрямую, минуя системную память. На плате видеоадаптера на основе Chipset GeForce 256 может быть установлено от 8 до 128 Мбайт локальной памяти типа SDRAM, SGRAM или DDR SDRAM (в перспективе планируется поддержка наиболее быстрой памяти типа Rambus). Пропускная способность шины локальной памяти составляет 6,4 Гбайт/с (как видим, она более чем в 6 раз выше, чем у шины AGP в режиме «4x»!).

Значительно усовершенствован в GeForce 256 и модуль рендеринга: теперь он включает в себя 4 конвейера, каждый из которых имеет один блок текстурирования. Благодаря этому скорость закраски (Fill Rate) составила 480 млн пикселов в секунду и 240 млн — в режиме мультитекстурирования.

3.3.3.5.2. GeForce 2 GTS

Следующими моделями данного семейства стали Chipset GeForce 2 GTS (рис. 3.43, справа), называемый также NV15, и его упрощенная модификация NV11. Аббревиатура GTS происходит от Gyga Texel Shading, что говорит о возможности обработки миллиарда текселов в секунду, т. е. об исключительно мощном конвейере рендеринга.

Chipset NV15 и NV11 произведены по технологии 0,18 мкм, поэтому тактовые частоты ядра и памяти удалось увеличить до 200 МГц. Другое существенное, по сравнению с GeForce 256, улучшение: в каждом

конвейере рендеринга в NV15 используется не один, а два блока текстурирования. Благодаря этому не происходит снижения скорости закраски (Fill Rate) в режиме мультитекстурирования: у NV15 она составляет 800 млн пикселов в секунду, при этом обрабатывается 1600 млн текселов. В Chipset NV11 количество конвейеров рендеринга снижено с четырех до двух.

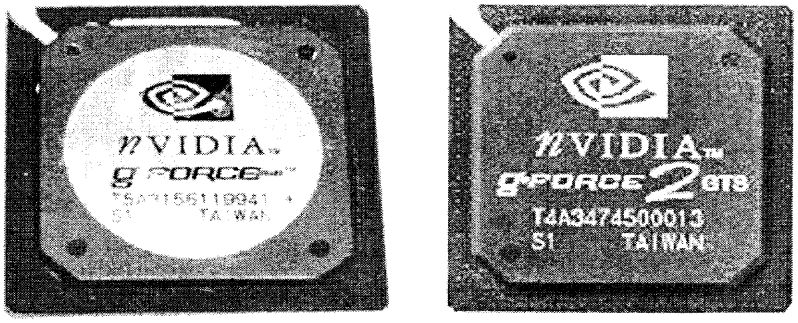


Рис. 3.43. Графические процессоры фирмы nVidia: слева — GeForce 256 (NV10), справа — GeForce2GTS(NV15)

Геометрический процессор обоих Chipset также улучшен по сравнению с GeForce 256: он дополнительно выполняет аппаратное удаление невидимых граней (Clipping), поэтому название данного узла немного изменилось — TCL вместо T&L. В результате производительность NV15 составила 25 млн треугольников в секунду против 15 млн у GeForce 256.

Модели NV15 и NV11 различаются разрядностью шины локальной памяти: в NV11 в целях удешевления модели использована 128-разрядная шина.

Как и ее предшественник, NV15 обладает широкими дополнительными возможностями в обработке сигналов от бытовой видеоаппаратуры (рис. 3.44).

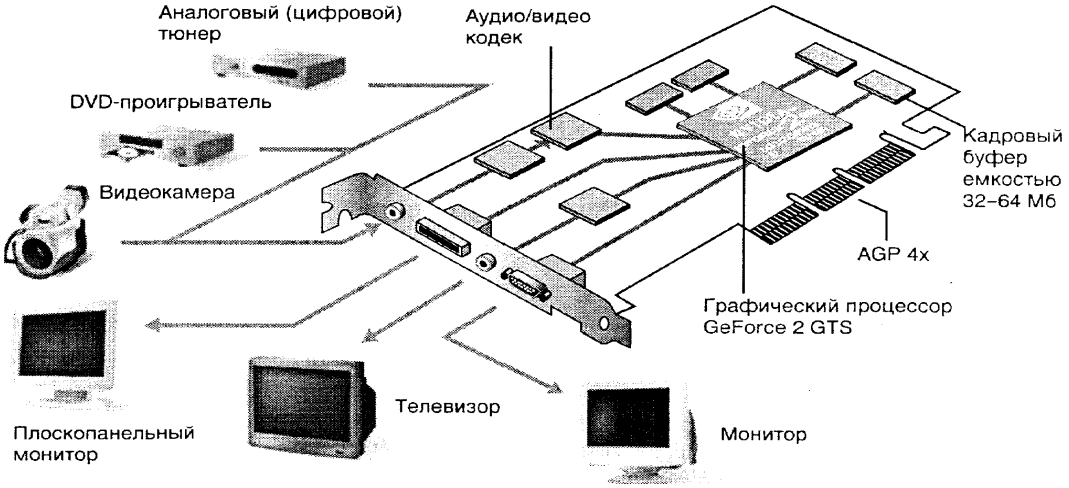


Рис. 3.44. Схема подключения внешних устройств к видеоадаптеру на основе Chipset GeForce 2 GTS (NV15)

Пример видеоадаптера на выполненного на основе Chipset GeForce2 GTS - видеоадаптер WinFast GeForce2 GTS фирмы Leadtek.

3.3.4. Chipset фирмы Matrox

Продукцией фирмы Matrox традиционно являются высококачественные Chipset и платы для ускорения 2D-графики (например, видеоадаптеры Matrox Mystique и Matrox Millennium считаются эталоном качества). В настоящее время фирма активно осваивает рынок 3D-акселераторов. Первой, и довольно неудачной попыткой был 2D/3D-акселератор Matrox Productiva G100. Имея прекрасные показатели в 2D-графике и непривычно низкую для изделий Matrox цену, эта карта не обеспечивала приемлемого качества 3D-изображений. Следующие два изделия фирмы — видеоадаптеры Matrox Millennium G200 и Matrox Mystique G200 — выполнены на основе Chipset Matrox MGA G200 и уже имеют встроенный акселератор трехмерной графики. Одной из последних разработок фирмы является Chipset MGA G400.

3.3.4.1. Matrox MGA G200

Chipset Matrox MGA G200 является совмещенным 128-разрядным 2D/3D-акселератором с интерфейсом AGP (поддерживаются режимы «2x» и DiME). На плате акселератора на основе Chipset Matrox MCjA G200 обычно устанавливается 8 Мбайт видеопамяти типа SGRAM или SDRAM; возможно увеличение ее объема

до 16 Мбайт. Отличительной особенностью Chipset является высокая разрядность z-буфера — 32 бита.

Данный Chipset, как и RivaTNT, также называют «киллером» знаменитого Voodoo^2. Это объясняется тем, что Chipset Matrox MGA G200 может работать в режиме True Color с высоким разрешением: при наличии 16 Мбайт локальной памяти он обеспечивает разрешение 1280×1024. Если объем установленной локальной памяти 8 Мбайт, то максимальное разрешение в режиме True Color составит 1024×768. Однако в этом режиме, в отличие от 16-битного режима High Color, скорость работы акселератора падает на 30-40%. Отметим, что работать в режиме True Color совсем не обязательно. Изделия фирмы Matrox отличаются очень качественной реализацией дизеринга на основе технологии VCQ (*Vibrant Color Quality*— качество «вибрирующих» цветов) собственной разработки. Так что изменение качества 3D-изображения при работе в 16-битном режиме High Color будет практически незаметным.

Chipset MGA G200 реализует полный набор аппарат ускоряемых 3D-функций, включая фирменную особенность изделий фирмы Matrox — аппаратную поддержку рельефного текстурирования с использованием карт окружающей среды. Однако качество реализации этих функций (за исключением последней) невысокое.

Специфический недостаток акселераторов на основе Chipset MGA G200 — наличие «желе» на движущемся изображении, т. е. расплывчатость и подвижность элементов поверхности медленно перемещающегося объекта. Плохо реализована также аппаратная поддержка антиалиасинга.

Chipset содержит RAMDAC, работающий на тактовой частоте 230 или 250 МГц (в зависимости от модели). Традиционно видеоадаптеры на основе этого Chipset имеют разъем для подключения дочерней платы, позволяющей реализовать аппаратную поддержку дополнительных функций по обработке видеосигнала, хотя некоторые модели обеспечивают это за счет встроенных средств.

К недостаткам Chipset стоит отнести отсутствие поддержки API OpenGL, благодаря чему невозможна, например, полноценная игра в Quake II. Однако имеется эмулирующая программа — wrapper (фирмы Matrox, которая частично решает данную проблему.

Видеоадаптер на основе MGA G200 не имеет столь очевидной зависимости производительности от быстродействия центрального процессора и шины, как, например, платы на основе Voodoo^2 или Riva TNT. что говорит о сравнительно низкой скорости выполнения конвейера рендеринга. Это значит, что Chipset MGA G200 целесообразнее использовать на компьютерах с процессором Celeron. *Если* же в перспективе вы планируете заменить свой процессор на Pentium 11, то для работы с 3D-приложениями стоит подумать о более быстродействующей плате.

В целом Chipset ориентирован самой фирмой Matrox па работу в составе PC нижнего и среднего уровня. Его преимуществом является очень высокое быстродействие и качество 2D-акселерации. Однако по качеству воспроизведения 3D-изображений, особенно по скорости, он заметно отстает от своих ближайших конкурентов — Voodoo^2 и Riva TNT.

Существует несколько модификаций плат на основе Chipset MGA G200. Наиболее распространена Matrox Millennium G200, имеющая достаточно высокие характеристики (RAMDAC с тактовой частотой 250 МГц). Плата Matrox Mystique G200 больше ориентирована на игровые приложения (RAMDAC с тактовой частотой 230 МГц) и позволяет осуществлять ввод/вывод видеосигнала (эта функция отсутствует у платы Matrox Mystique G200/SD). И, наконец, для любителей мультимедиа и домашнего видео фирма Matrox выпустила плату Matrox Marvel G200. На ней имеется RAMDAC, работающий на частоте 350 МГц, встроенный TV-тюнер, реализована аппаратная поддержка ввода/вывода видеосигналов и декодирования видеосюжетов в форматах MPEG 1 и MPEG 2.

3.3.4.2. Matrox MGA G400

Невысокая, по сравнению с изделиями nVidia и 3Dfx, скорость работы Chipset MGA G200 по обработке 3D-изображений заставила специалистов Matrox улучшить 3D-ядро своего Chipset. В результате появился новый Chipset — Matrox MGA G400 (рис. 3.48), на основе которого в настоящее время выпускаются три типа видеоадаптеров:

- Matrox Millennium G400 16Mb AGP;
- Matrox Millennium G400 32Mb AGP;
- Matrox Millennium G400 MAX 32Mb AGP.

Модели G400 и G400 MAX различаются, соответственно, тактовыми частотами ядра (125 и 150 МГц), памяти (166 и 200 МГц) и RAMDAC (300 и 360 МГц), а также производительностью конвейера рендеринга (250 и 300 млн. пикселов в секунду). Пропускная способность (Triangle Throughput) обоих моделей примерно одинакова и составляет около 8 млн. треугольников в секунду. Chipset Matrox MGA G400 имеет более высокую производительность, чем Riva TNT, но уступает моделям на основе Riva TNT2 Ultra (модель G400 MAX практически не отстает от последней).

Стоимость видеоадаптеров на основе Chipset Matrox G400 и Riva TNT2 Ultra также примерно одинакова (первые чуть дороже). Набор аппаратно ускоряемых 3D-функиий для 3D-акселераторов данного уровня также стандартный, при этом отдельного упоминания заслуживает качественная аппаратная реализация рельефного текстурирования с использованием карт окружающей среды — фирменная особенность 3D-ядра

фирмы Matrox. Впрочем, в данном режиме производительность видеосистемы снижается примерно на 30—35%.

В отличие от других моделей, улучшена аппаратная поддержка декодирования видеопотоков в формате MPEG-2: при использовании Matrox G400 загрузка центрального процессора составляет всего 50-65% (при использовании видеоадаптеров других моделей значение этого параметра составляет 85—100%).

Хотя производители отнесли Chipset Matrox G400 к семейству 256-разрядных устройств, это не совсем правильно: 3D-ядро очень похоже на ядро Chipset MGA G200, т. е. является 128-разрядным, однако в MGA G400 использована двойная 256-разрядная шина памяти. Эта шина представляет собой совокупность двух однонаправленных 128-разрядных шин, по которым одновременно и независимо друг от друга передаются данные из видеопамяти в 3D-ядро и обратно.

Chipset компании Matrox традиционно характеризуются исключительно высоким качеством формируемого двумерного изображения, и MGA G400 не исключение. Он является безусловным лидером среди всех рассмотренных выше Chipset. Особенно это заметно при использовании высококачественных мониторов с экраном большого размера (не менее 17").

Пожалуй, наиболее характерным отличием видеоадаптеров на основе Chipset Matrox MGA G400 является наличие у некоторых моделей (например, Matrox Millennium MGA G400) двух разъемов для одновременного подключения либо двух мониторов, либо монитора и телевизора — эта функция называется *Dual Head.* В последнем случае, т. е. для подключения телевизора или аналогового видеомагнитофона, используется специальный переходник VGA—S-Video или VGA-RCA.

Возможность подключения телевизора или видеомагнитофона (функция TV-out) к видеоадаптеру обеспечивается благодаря наличию на плате специальных микросхем, выполняющих цифро-аналоговое преобразование видеосигнала. Хотя эта функция и является стандартной для многих видеоадаптеров, в Matrox Millennium MGA G400 она реализована особенно хорошо — изображение на экране телевизора получается очень четким и при этом не ухудшается качество изображения на экране монитора.

Но лучше всего вы сможете оценить преимущества видеоадаптера с функцией Dual Head, если одновременно подключите два монитора. В этом случае можно либо наблюдать одинаковое изображение на двух экранах, либо растянуть одно изображение на оба экрана. Такая функция особенно полезна при работе в графических приложениях, настольных издательских системах и т. и. Например, в Adobe Photoshop рабочее поле, как правило, загромождено множеством открытых инструментальных окон. При использовании видеоадаптера с функцией Dual Head все открытые окна можно разместить на экране одного монитора и оставить свободным для работы поле на экране другого.

Для реализации функции Dual Head необходимы два независимых контроллера ЭЛТ. Напомним, что контроллер ЭЛТ является одним из главных компонентов видеоадаптера (2D-ядра) и предназначен для управления процессом сканирования видеопамяти и формированием синхросигналов. Наличие двух независимых контроллеров ЭЛТ позволяет устанавливать различные разрешения, частоту кадров и глубину цвета для каждого монитора. Для настройки мониторов не требуется какого-либо специального программного обеспечения — для этого используется стандартное диалоговое окно Свойства: Экран (Display Properties), в котором после установки драйвера видеоадаптера Matrox Millennium MGA G400 появятся дополнительные закладки.

Все перечисленные особенности делают видеоадаптер Matrox Millennium MGA G400 весьма удачным выбором среди изделий такого класса.

Традиционным недостатком 3D-акселераторов производства фирмы Matrox является низкая производительность в приложениях, ориентированных на API OpenGL. Этот недостаток присущ и Chipset Matrox MGA G400, хотя в настоящее время фирма усиленно работает над созданием эффективных драйверов уровня ICD.

3.3.5. Chipset фирмы ATI

Канадская фирма ATI хорошо известна на рынке 2D-акселераторов. Также как и у фирмы Matrox, первый дебют ATI на рынке 3D-акселераторов с Chipset ATI 3D Rage PRO был неудачным: при достаточно высокой производительности в 2D-графике он показал низкую производительность и некорректно реализованные функции в 3D-графике.

Первой удачной разработкой фирмы явился 128-разрядный Chipset ATI Rage 128, в котором фирма ATI постаралась реализовать множество технологических новшеств (например, текстурную и пиксельную кэш-память). Интересно отметить, что при его создании были использованы наработки двух других компаний — Tseng Labs и OAK Technologies. Первая была полностью куплена фирмой ATI, а у второй была приобретена технология WARP (*Windows Accelerator and Rendering Processor*— акселератор для Windows и процессор рендеринга). Благодаря этому ожидалось, что Chipset ATI Rage 128 обеспечит высокую производительность и качество изображения и станет достойным конкурентом лидерам рынка того периода — Chipset Riva TNT и Voodoo^2.

Отчасти так и получилось, однако ошибки в реализации отдельных 3D-функций, недостаточная отлаженность драйверов (особенно драйверов уровня ICD для OpenGL), а также традиционно более высокая цена изделий, по сравнению с Chipset фирмы nVidia, не способствовали успеху ATI Rage 128. Тем более, что к моменту его выпуска на рынке появились Chipset следующего поколения — Riva TNT2 Ultra и Voodoo^3.

Исправив допущенные в Chipset ATI Rage 128 ошибки и улучшив его характеристики, фирма ATI выпустила новый Chipset Rage 128 Pro, который по производительности приблизился к Riva TNT2 Ultra. На основе этого Chipset фирма ATI выпустила несколько моделей видеоадаптеров семейства ATI Rage Fury Pro, различающихся набором мультимедийных функций и объемом установленной видеопамяти. Появились также видеоадаптеры ATI Rage Fury Pro MAXX, у которых на одной плате установлено два Chipset Rage 128 Pro.

Относительно новым изделием фирмы ATI является 3D-акселератор нового поколения — Radeon 256, оснащенный геометрическим процессором Charisma Engine. Он относится к новому поколению ускорителей фотореалистичной графики и составляет конкуренцию Chipset GeForce фирмы nVidia и Savage 2000 фирмы S3.

3.3.5.1. ATI Rage 128

Так же как и все современные изделия для ускорения 3D-графики, Chipset ATI Rage 128 является мультимедийным 2D/3D-акселератором с интерфейсом AGP (поддерживает режим «2x»). Chipset имеет 128-разрядную архитектуру, ядро и видеопамять работают на тактовой частоте 103 МГц. На плате может быть установлено 16 или 32 Мбайт видеопамяти типа SGRAM. Встроенный RAMDAC работает на частоте 250 МГц, благодаря чему обеспечивается максимальное разрешение 1920×1200. 3D-ядро содержит два конвейера рендеринга и выполняет аналогичный Chipset Riva TNT набор ускоряемых 3D-функций, за исключением анизотропной фильтрации (реализация билинейной фильтрации выполнена также с ошибками). Пропускная способность (Triangle Throughput) Chipset составляет 4 млн треугольников в секунду, скорость закраски (Fill Rate) — 200 млн пикселов в секунду. При обработке 3D-изображений используется 32-битный z-буфер, поддерживаются текстуры больших размеров (1024×1024) и 32-битный цвет.

Новинкой в традиционной архитектуре 3D-ядра стало наличие текстурной и пиксельной кэш-памяти, ускоряющих процесс рендеринга. Традиционно большинство видеоадаптеров фирмы ATI, в том числе и на основе Chipset Rage 128, характеризуются оптимизированным 128-разрядным 2D-ядром, обеспечивающим высокую скорость работы и отличное качество формируемых 2D-изображений.

Помимо стандартного набора аппаратно ускоряемых 2D-функций, Chipset обеспечивает частичное аппаратное декодирование видеопотоков форматов MPEG-2 и DVD, что позволяет снизить загрузку центрального процессора примерно на 45%.

Кроме этого возможна установка дополнительных микросхем, обеспечивающих реализацию функции TV-out.

Пример видеоадаптера на основе Chipset Rage 128: ATI XPERT 128.

3.3.5.2. ATI Rage 128 Pro

Недостатки ATI Rage 128 и поставляемого с ним программного обеспечения серьезно затруднили продвижение этого Chipset на рынке. Поэтому компания ATI вскоре выпустила новую, усовершенствованную версию Chipset — Rage 128 Pro. Этот Chipset отличается от предыдущих изделий фирмы следующими характеристиками:

- выполнен по технологии 0,25 мкм;
- тактовые частоты ядра и памяти увеличены до 140 и 150 МГц соответственно;
- тактовая частота RAMDAC увеличена до 300 МГц, что позволило повысить максимальное разрешение до 2048×1536;
- исправлены ошибки реализации билинейной фильтрации, введена аппаратная поддержка анизотропной фильтрации;
- усовершенствован механизм Setup Engine, благодаря чему пропускная способность (Triangle Throughput) Chipset увеличилась до 8 млн треугольников в секунду;
- реализована поддержка режима «4x» интерфейса AGP.

Благодаря этим изменениям, Chipset ATI Ragel28 Pro no производительности вплотную приблизился к Chipset Riva TNT2 Ultra, а по отдельным параметрам даже превзошел его.

В качестве дополнительного стимула для приобретения видеоадаптера на основе данного Chipset фирма ATI аппаратно реализовала в нем некоторые мультимедийные функции, в частности, модель ATI Rage Fury Pro VIVO обеспечивает качественный ввод/вывод аналогового видеосигнала (функция Video In/Out).

3.3.5.3. ATI Rage Fury MAXX

Дополнительным средством в конкурентной борьбе компании ATI против nVidia, S3 и 3Dfx является новая технология MAXX, позволяющая устанавливать на одной плате видеоадаптера несколько Chipset, параллельно выполняющих обработку 3D-графики (при работе с обычной, двумерной графикой в этом нет необходимости). Такое решение можно считать аналогом технологии SLI, которая давно и успешно применяется в изделиях компании 3Dfx.

В результате был выпущен видеоадаптер ATI Rage Fury MAXX (рис. 3.45), у которого на одной плате размещено два Chipset ATI Rage 128 Pro и видеопамять типа SDRAM емкостью 64 Мбайт (по 32 Мбайт для

каждого Chipset).

В обычной работе (в 2D) задействуется только один Chipset. Наличие второго потенциально обеспечивает работу с двумя мониторами (на плате даже имеется место для второго разъема), однако пока эта функция не поддерживается. При работе с 3D-приложениями производительность ATI Rage Fury MAXX приближается к производительности Chipset GeForce 256.

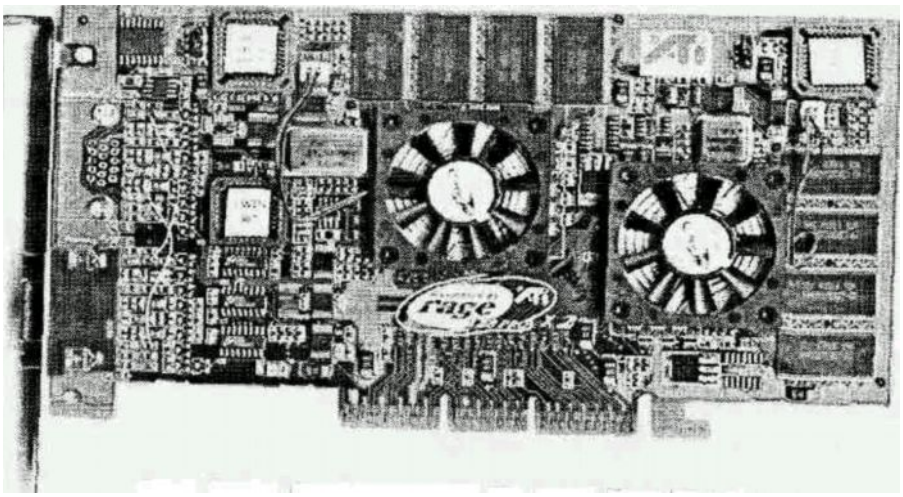


Рис. 3.45. Видеоадаптер ATI Rage Fury MAXX

3.3.5.4. ATI Radeon 256

Принципиально новым изделием ATI является 3D-акселератор нового поколения — Chipset Radeon 256.

Особенности архитектуры и основные характеристики ATI Radeon 256 как одного из ускорителей фотореалистичной графики были рассмотрены в разд. 3.2.3. Здесь же отметим лишь следующие его особенности:

- 256-разрядное графическое ядро состоит из двух, работающих параллельно, 128-разрядных блоков;
- помимо традиционной для современных видеоадаптеров памяти типа SDRAM/SGRAM, ATI Radeon 256 поддерживает работу с перспективной памятью типа DDR SDRAM;
- помимо графического процессора (2D/3D-ядра), на плате видеоадаптера установлена микросхема Rage Theater, позволяющая выполнять ввод/вывод аналогового видеосигнала и его аппаратное сжатие;
- Chipset ATI Radeon 256 поддерживает технологию MAXX, поэтому в скором времени возможно появление видеоадаптеров, содержащих несколько Chipset.

Заметим, что с появлением данного Chipset, имеющего не менее высокие характеристики, чем GeForce 256, компания ATI впервые на рынке 3D-устройств класса High-End составила серьезную конкуренцию фирме nVidia.

3.3.6. Chipset фирмы S3, Inc.

Фирма S3, Inc. имеет репутацию изготовителя недорогих, но имеющих достаточно высокое быстродействие Chipset для видеоадаптеров, благодаря чему ее продукция составляет значительную долю рынка 2D-устройств. Первые попытки S3 завоевать рынок 3D- устройств были неудачными. Выпущенный в конце 1995 г. видеоадаптер на основе Chipset S3 Virge GX/DX обладал хорошей производительностью в 2D-приложениях, но не обеспечивал значительного ускорения операций с трехмерной графикой, даже несмотря на попытки S3 анонсировать данное изделие как 2D/3D-акселератор. Это стало особенно очевидно после появления акселератора Voodoo Graphics.

Следующая попытка — Chipset S3 Trio3D — имела еще меньший успех, поскольку аппаратное ускорение основных 3D-функций в этом Chipset также отсутствовало (в этой связи не могут не вызывать улыбку имеющиеся в продаже AGP-модификации видеоадаптеров на основе Chipset Virge GX/DX и Trio3D, поскольку использование в этих моделях интерфейса AGP совершенно бесполезно).

Взяв на вооружение последние достижения в области ускорения функций трехмерной графики, фирма S3 создала совершенно новый продукт — Chipset Savage 3D. Как и любое первое изделие для ускорения 3D-графики, Chipset получился довольно «сырым»: его производительность явно не дотягивала до уровня Riva TNT, зато по сбоям и неустойчивости работы он был в первых рядах. Однако в новой версии Chipset — Savage 4 — уже были исправлены многие недостатки Savage 3D. Новый Chipset, благодаря высокой производительности (на уровне Riva TNT), устойчивой работе и сравнительно невысокой цене, получил довольно широкое распространение, хотя к изделиям класса High-End его отнести все же нельзя.

Зато одну из последних разработок фирмы — Savage 2000 — вполне можно охарактеризовать как

высококлассный 3D-акселератор нового поколения, содержащий геометрический процессор (любопытно, что фирма S3 всего на один день, но все же опередила своего главного конкурента — фирму nVidia, анонсировав Savage 2000 как первый 3D-акселератор с геометрическим ускорителем).

3.3.6.1. S3 Savage 3D

S3 Savage 3D имеет 128-разрядное 2D/3D-ядро (два блока по 64 разряда), изготовленное по технологии 0,25 мкм, а также ставший уже традиционным стандартный набор средств по обработке видеоданных. Наиболее существенным отличием данного Chipset от его конкурентов является использование в нем технологии сжатия текстур S3TC собственной разработки фирмы. Эта технология позволяет решить две задачи:

- значительно (в несколько раз) снизить требования к объему видеопамяти, необходимой для хранения текстур, либо увеличить разрешение текстур, хранящихся в видеопамяти;
- увеличить скорость передачи текстур в акселератор из системной памяти PC за счет того, что текстуры пересылаются в сжатом виде и скорость информационного потока значительно снижается.

Благодаря этим достоинствам, технологию S3TC лицензировала фирма Microsoft, включив ее поддержку в API DirectX 6.0. Однако в настоящее время лишь немногие приложения поддерживают данную технологию (к числу таковых относится игра Unreal).

Сделав ставку на использование сжатых текстур, S3 реализовала в Savage 3D поддержку всего 8 Мбайт видеопамяти. Chipset поддерживает работу с видеопамятью двух типов: работающей на частоте 125 МГц SGRAM и более дешевой SDRAM. Как известно, такой подход себя не оправдал: даже при поддержке режима «4x» производители видеоадаптеров стремятся наращивать объем локальной памяти (что в дальнейшем сделала и S3 в своей новой модели Savage 4).

Будучи ориентированным на интерфейс AGP, Chipset Savage3D поддерживает и интерфейс PCI (спецификация PCI 2.1), включая управление питанием шины.

Аппаратные особенности Chipset Savage3D:

- 128-разрядное (2×64) 3D-ядро с тактовой частотой 125 МГц;
- реализация однопроходного рендеринга при аппаратно поддерживаемой трилинейной фильтрации (при этом используются числа с плавающей точкой, т. е. корректно учитывается дробная часть числа);
- довольно высокая для 3D-акселераторов данного класса пропускная способность (Triangle Throughput) — 5 млн треугольников в секунду, обусловленная наличием модифицированного механизма обработки треугольников Triangle Setup Engine (обеспечивает оптимизированный режим передачи данных о смежных вершинах элементарных треугольников — *triangle strip and fans);*
- невысокая скорость закраски (Fill Rate) — 125 млн пикселов в секунду (при включенной трилинейной фильтрации);
- краевой антиалиасинг, отсутствие аппаратной поддержки полного антиалиасинга;
- отсутствие аппаратной поддержки анизотропной фильтрации.

Что же касается качества реализации 2D-функций, то оно находится на традиционно высоком для S3 уровне.

В настоящее время стоимость видеоадаптеров на основе S3 Savage 3D значительно снизилась, однако низкое, по сравнению с Riva TNT, быстродействие, а также явное несовершенство аппаратного и программного обеспечения могут создать массу проблем в работе и настройке адаптера.

3.3.6.2. S3 Savage 4

Спустя 7 месяцев после выпуска Savage 3D фирма S3 изготовила новый Chipset — Savage 4, в котором были устранены многие недостатки предыдущего Chipset. Также как и Savage 3D, Savage 4 имеет 128-разрядное 2D/3D-ядро, выполненное по технологии 0,25 мкм. Этот Chipset выпускается в трех модификациях, различающихся тактовыми частотами ядра/видеопамяти, а также ее размером и быстродействием:

- Savage 4 GT(110/125 МГц, до 16 Мбайт локальной памяти типа SDRAM);
- Savage 4 Pro (110/125 МГц, до 32 Мбайт локальной памяти типа SDRAM или SGRAM);
- Savage 4 Pro+ (125/143 МГц, до 32 Мбайт локальной памяти типа SDRAM или SGRAM).

В настоящее время на плате видеоадаптера на основе Chipset семейства Savage 4, как правило, устанавливается только память типа SDRAM с временем доступа 10 нс для Chipset Savage 4 GT и 8 нс — для Savage 4 Pro. Опыт показывает, что тактовую частоту без проблем можно повышать до 135— 150 МГц (в зависимости от времени доступа установленных микросхем памяти).

Основные отличия Savage 4 от Savage 3D:

- увеличена до 300 МГц тактовая частота RAMDAC;
- улучшен механизм преобразования формы представления координат вершин граней (Triangle Setup Engine), благодаря чему пропускная способность (Triangle Throughput) Chipset возросла с 5 до 8 млн

треугольников в секунду;
- добавлена аппаратная поддержка анизотропной фильтрации;
- усовершенствован конвейер рендеринга: мультитекстурирование и трилинейная фильтрация выполняются за один такт, скорость закраски (Fill Rate) увеличилась до 140 млн пикселов в секунду;
- реализована аппаратная поддержка рельефного текстурирования по методу вдавливания (Embossing);
- реализована аппаратная поддержка полного антиалиасинга (FSAA);
- обеспечена поддержка режима «4x» интерфейса AGP (у Chipset Savage 4 Pro);
- добавлен 8-разрядный буфер шаблонов, позволяющий динамически изменять тени от объектов;
- поддерживается пиксельный и полигонный туман;
- реализованы различные алгоритмы преобразования текстур, включая текстурирование с использованием программно-изменяемых рельефных карт;
- реализована поддержка текстур большого размера (до 2048×2048 пикселов).

Благодаря этим доработкам, Chipset Savage 4 по функциональным возможностям и производительности практически сравнялся с Riva TNT. Отличительная черта Chipset Savage 4 — прекрасное качество прорисовки трехмерной сцены, недоступное для Chipset большинства других фирм. Однако проблемы с драйверами частично остались, что особенно заметно при работе с приложениями, ориентированными на OpenGL.

Не помогло решению проблемы и приобретение компанией S3 фирмы Diamond Multimedia, традиционно отличавшейся высококачественными драйверами. Имеет место также плохая совместимость Chipset Savage 4 с некоторыми моделями материнских плат: по причине некорректной реализации функции Plug&Play (система неправильно распознает подключенный к видеоадаптеру на основе Chipset Savage 4 монитор).

Несмотря на то что Chipset содержит RAMDAC, работающий на достаточно высоко частоте, и видеопамять большого объема, поставляемые драйверы (даже разработанные Diamond Multimedia) не позволяют работать в режиме True Color с разрешением выше 1152×854. Однако главным недостатком данного Chipset все же являются некачественные драйверы, особенно уровня ICD для OpenGL. Если вы любите играть в Quake или работаете с приложениями, ориентированными на API OpenGL (например, 3D Studio Max), то видеоадаптер на основе Savage 4 вам вряд ли подойдет: скорость работы будет очень низкой. Правда, для Chipset Savage 3D и Savage 4 фирма S3 разработала свой API — Metal3D, при использовании которого видеоадаптеры на основе этих Chipset демонстрируют великолепную производительность. Однако приложений, ориентированных на данный API, очень мало. Наиболее известным из них является игра Unreal.

В целом видеоадаптер на основе Savage 4, оснащенный видеопамятью емкостью не менее 16 Мбайт, является недорогим, недостаточно удачным выбором для домашнего компьютера, благодаря чему составляет конкуренцию видеоадаптерам на основе RivaTNT, а модели на основе Savage 4 Pro+ (например. Diamond Stealth III S540) сопоставимы по быстродействию с видеоадаптерами на основе RivaTNT2 M64. Преимущество этих адаптеров над их конкурентами еще более заметно при использовании CPU фирмы AMD (не хуже K6-11 с тактовой частотой 350 МГц). Посему выходит, что графические процессоры фирмы S3 поддерживают набор команд 3Dnow!.

3.3.6.3. S3 Savage 2000

Хотя выпуск Chipset Savage 4 и считается успехом фирмы S3, но не позволил ей стать лидером и занять свое место в ряду производителей 3D-акселераторов класса High End. Скорее, наоборот: нынешнее положение фирмы на рынке 3D-устройств стало сильно напоминать многолетнюю ситуацию на рынке обычных (2D) видеоадаптеров, когда S3 характеризовалась как производитель не слишком дорогих, но и не самых лучших изделий. Стремясь исправить это положение и догнать лидера (фирму nVidia), S3 выпустила новый, 128-разрядный Chipset, получивший название Savage 2000.

Основное отличие этого Chipset от его предшественников — наличие геометрического процессора *S3TL (S3 Transform and Lighting).* Он обеспечивает аппаратное ускорение операции геометрической стадии 3D-конвейера, а именно, расчет преобразования координат и освещенности (до 8 источников света). Наличие геометрического процессора в составе Savage 2000 позволяет поставить его в один ряд с такими моделями, как GeForce 256 и Radeon 256, однако производительность S3TL примерно вдвое ниже, чем у этих моделей. Остальные особенности архитектуры Savage 2000 делают его похожим на 128-разрядные 3D-акселераторы типа Riva TNT2 и Savage 4.

Несмотря на то что ядро Savage 2000 выполнено по передовой технологии 0,18 мкм, его номинальная тактовая частота сравнительно невелика — всего 125 МГц, максимальная — 150 МГц. Chipset поддерживает до 64 Мбайт видеопамяти типа SDRAM или SGRAM, работающей на частоте от 143 до 166 МГц. Имеется встроенный RAMDAC, работающий на частоте 350 МГц. Набор аппаратно реализуемых функций практически такой же, как у Savage 4. за исключением одной — *Single-pass Quad-Texture engine,* позволяющей накладывать четыре текстуры за один проход. Пропускная способность (Triangle Throughput) Chipset Savage 2000 составляет 10—15 млн треугольников в секунду, скорость закраски (Fill Rate) — 300 млн

пикселов в секунду.

Анонсирована модель Savage 2000+. у которой максимальные тактовые частоты ядра и памяти увеличены до 200 МГц. У этого Chipset значение параметра Fill Rate достигает 400 млн пикселов в секунду.

По сравнению с Savage 4, в новом Chipset значительно улучшена поддержка API OpenGL, однако недостаточно проработаны драйверы, ориентированные на Direct3D. Как и следовало ожидать, наивысшую производительность этот Chipset демонстрирует при использовании API Metal3D.

В целом Savage 2000 по быстродействию превосходит Chipset Riva TNT2 и Rage 128, однако явно уступает GeForce 256 (в основном из-за менее мощного геометрического процессора и более узкой шины видеопамяти). Единственное преимущество этого Chipset, по сравнению с GeForce 256, — более низкая цена (примерно на 30%).

3.3.7. Chipset фирмы 3Dlabs

Фирма 3Dlabs выпускает профессиональные графические акселераторы, работающие главным образом под управлением API OpenGL. Ее продукция — Chipset Permedia и Permedia 2 — предназначена для использования в рабочих станциях нижнего уровня, на которых выполняются задачи 3D-моделирования или САПР. Производительность этих изделий в играх очень мала, поэтому они не годятся для этой цели.

Новой разработкой фирмы является высокоскоростной 128-разрядный Chipset Permedia 3. В Chipset аппаратно реализованы практически все упоминавшиеся ранее 2D/3D-функции. Пропускная способность (Triangle Throughput) Chipset составляет 8 млн треугольников в секунду, а скорость закраски (Fill Rate) — 250 млн пикселов в секунду. Такие значения параметров обусловлены наличием встроенного геометрического процессора Delta. Chipset поддерживает работу с локальной памятью типа SGRAM или SDRAM емкостью от 4 до 16 Мбайт, работающей на частоте 200 МГц. Тактовая частота RAMDAC составляет 270 МГц.

Поскольку Chipset поддерживает систему команд обоих API (OpenGL и DirectX), видеоадаптеры на его основе могут использоваться не только профессионалами, но и любителями компьютерных игр. К сожалению, цена на эти изделия намного выше, чем у традиционных игровых 3D-акселераторов.

Другим примером 3D акселератора фирмы 3Dlabs является профессиональный видеоадаптер семейства Oxygen на основе Chipset GLint R3.

Для наглядности и удобства сравнения основные характеристики рассмотренных Cliipset сведены в табл. 3.2 (состояние на 1999г.).

3.3.8. Современные графические процессоры.

Последние несколько лет графические процессоры (ГП) для ПК улучшались очень быстро, нисколько не уступая в этом отношении центральным процессорам. Ядро ГП NVID1A RIVA TNT2, запущенного в производство в марте 1999 г., состояло из 10,5 млн. транзисторов, NVID1A Ge-Force2 GTS - 25 млн. (апрель 2000 г.), NVID1A Ge-Force3 — 57 млн. (февраль 2001 г.), а ГП модельного ряда NVID1A GeForce4 Ti — уже 63 млн. (февраль 2002 г.). В то же время ЦП Pentium III на ядре Katmai без встроенного кэша L2, выпущенный в феврале 1999 г., содержал 9,5 млн. транзисторов, на ядре Coppermine с 256-Кбайт кэшем L2 — 28,1 млн. (октябрь 1999 г.), на Tualatin с 512-Кбайт кэшем L2 — 44 млн. (июнь 2001 г.), ЦП Pentium 4 на ядре Willamette с 256-Кбайт кэшем L2 — 42 млн. (ноябрь 2000 г.), а на ядре Northwood с 512-Кбайт кэшем L2 — 55 млн. (январь 2002 г.). Как видно из приведенных данных, по темпам развития и сложности ГП обошли процессоры и, судя по всему, дальше разрыв будет увеличиваться, поскольку компьютерные игры — главный «двигатель прогресса» этого сектора рынка — усложняются все быстрее и способны легко «затормозить» даже самый мощный ГП.

Еще одна важная тенденция развития рынка графических ускорителей, особенно сильно проявившаяся за последний год, — деление его на множество узких, но пересекающихся потребительских ниш, так что выбор графического адаптера превратился в настоящую головоломку для рядовых пользователей. Сегодня на рынке активно продаются графические платы на базе более десятка современных ГП, не считая устройства двух-трехлетней давности, которые тем не менее до сих пор довольно часто используются для построения недорогих ПК.

Появление в феврале 2001 г. ГП GeForce3 фирмы NVIDIA вызвало немалый интерес у поклонников 3D-игр — это был первый ГП NVIDIA с программируемым блоком обработки вершин (Vertex Shader). Вместе с блоками обработки пикселов (Pixel Shader), четырехканальным «кроссовым» контроллером видеопамяти (Crossbar Memory Controller), состоит из четырех отдельных 64-бит субконтроллеров, (см. рис.3.46), который позволяет повысить производительность благодаря параллельной работе с несколькими микросхемами памяти, и сжатием данных Z-буфера, это обещало поднять скорость обработки трехмерной графики и реалистичность изображения на качественно новый уровень.

Наименование Chipset		Технология изготовления, мкм	Размер локальной памяти	Тип памяти и тактовая частота шины памяти, МГц	Интерфейс	GPU	Разрядность 3D-ядра	Тактовая частота ядра, МГц	Разрядность шины памяти	Максимальная тактовая частота RAMDAC, МГц	Пропускная способность	Скорость закраски	API			Максимальное разрешение в 3D с z-буферизацией	Глубина цвета в 3D	Тип ускорителя
													Разрядность z-буфера	DirectX	OpenGL			
3Dfx	Voodoo Graphics	0,35	4–8	EDO, 48—65	PCI	O	64	60	64	135	500 тыс.	50 млн	16	●	MCD	800×600	16	3D
	Voodoo Rush	0,35	4–8	EDO, 48—65	PCI	O	64	60	64	135	500 тыс.	50 млн	16	●	MCD	800×600	16	2D/3D
	Voodoo2	0,35	6–12	EDO, 80—110	AGP/PCI	O	64	100	64	250	2,5 млн	90 млн	16	●	Miniport	800×600	16	3D
	Voodoo Banshee	0,35	4, 8, 16	SGRAM, SDRAM, 100	AGP/PCI	O	64	100	128	250	2,5 млн	90 млн	16	●	ICD	1600×1200	16	2D/3D
	Voodoo3	0,25	4–16	SGRAM, SDRAM, 100	AGP/PCI	O	64	143–183	128	300–350	6–8 млн	183 млн	16	●	Miniport (wrapper)	1600×1200	16	2D/3D
Intel	i740	0,35	2–16	SGRAM, SDRAM, 66—100	AGP	O	64	66–100	64	220	500 тыс	55 млн	16	●	ICD	1280×1024	8/16	2D/3D
nVidia	Riva 128	0,35	4	SDRAM, SGRAM, 90	AGP/PCI	O	128	100	128	230	5 млн	100 млн	16	●	ICD	960×712	24	2D/3D
	Riva 128ZX	0,35	8	SGRAM, 90	AGP/PCI	O	128	100	128	250	5 млн	100 млн	16	●	ICD	1200×1024	24	2D/3D
	Riva TNT	0,35	8, 16	SGRAM 125/SDRAM 110	AGP/PCI	O	128	90	128	250	8 млн	250 млн	24	●	ICD	1600×1200	32	2D/3D
	Riva TNT2	0,25	16, 32	SGRAM, SDRAM, 150	AGP	O	128	125	128	300	8 млн	300 млн	24	●	ICD	2048×1536	32	2D/3D
	Riva TNT2-A	0,22	16, 32	SGRAM, SDRAM, 150	AGP	O	128	143	128	300	8 млн	300 млн	24	●	ICD	2048×1536	32	2D/3D
	Riva TNT2 Ultra	0,22	32	SGRAM, SDRAM, 183	AGP	O	128	150	128	300	9 млн	300 млн	24	●	ICD	2048×1536	32	2D/3D
	Riva TNT2 Vanta	0,25	8, 16, 32	SDRAM, 100	AGP	O	128	125	64	250	8 млн	250 млн	24	●	ICD	2048×1536	32	2D/3D
	Riva TNT2 M64	0,25	8, 16, 32	SDRAM, 125 (140)	AGP	O	128	150 (183)	64	300	8 млн	300 млн	24	●	ICD	2048×1536	32	2D/3D
	GeForce 256 (NV10)	0,22	8–128	SGRAM, SDRAM, DDR SDRAM, 200	AGP	T&L	256	120	256	350	15 млн	480 млн	16/24/32	●	ICD	2048×1536	32	2D/3D
	GeForce 2 GTS (NV11)	0,18	8–128	SGRAM, SDRAM, DDR SDRAM, 200	AGP	TCL	256	200	256	350	25 млн	400 млн	16/24/32	●	ICD	2048×1536	32	2D/3D
	GeForce 2 GTS (NV15)	0,18	8–128	SGRAM, SDRAM, DDR SDRAM, 200	AGP	TCL	256	200	128	350	25 млн	800 млн	16/24/32	●	ICD	2048×1536	32	2D/3D
Matrox	MGA G200	0,25	4–32	SGRAM, SDRAM	AGP/PCI	O	128	200/250	64	250–350	1,5 млн	100 млн	32	●	ICD	1280×1024	32	2D/3D
	MGA G400	0,25	16, 32	SGRAM, SDRAM, 166	AGP/PCI	O	128	125	256	300	8 млн	250 млн	32	●	ICD	2048×1536	32	2D/3D
	MGA G400 MAX	0,25	16, 32	SGRAM, 200	AGP/PCI	O	128	150	256	360	8 млн	300 млн	32	●	ICD	2048×1536	32	2D/3D
ATI	Rage 128	0,25	16, 32	SGRAM, 103	AGP	O	128	103	128	250	4 млн	200 млн	16/24/32	●	ICD	1920×1200	32	2D/3D
	Rage 128 Pro	0,25	32	SGRAM, SDRAM, 150	AGP	O	128	140	128	300	8 млн	250 млн	32	●	ICD	2048×1536	32	2D/3D
	Radeon 256	0,18	8–128	SDRAM/SGRAM, DDR SDRAM, 183	AGP	Charisma Engine	256	200–400	256		30 млн	400 млн	32	●	ICD	2048×1536	32	2D/3D
S3	Savage3D	0,25	2–8	SGRAM, 125	AGP/PCI	O	128	125	64	250	5 млн	125 млн	16/24	●	ICD	1024×768	24	2D/3D
	Savage 4 GT	0,25	2–16	SDRAM, 125	AGP/PCI	O	128	110	128	300	8 млн	140 млн	16/24/32	●	ICD	1920×1440	32	2D/3D
	Savage 4 Pro	0,25	2–32	SDRAM, SGRAM, 125	AGP/PCI	O	128	110	128	300	8 млн	140 млн	16/24/32	●	ICD	1920×1440	32	2D/3D
	Savage 4 Pro+	0,25	2–32	SDRAM, SGRAM,143	AGP/PCI	O	128	125	128	300	8 млн	140 млн	16/24/32	●	ICD	1920×1440	32	2D/3D
	Savage 2000	0,18	2–64	SDRAM 143, SGRAM 166	AGP/PCI	S3TL	128	125	128	350	15 млн	300 млн	16/24/32	●	ICD	2048×1536	32	2D/3D
	Savage 2000+	0,18	2–64	SGRAM 200	AGP/PCI	S3TL	128	200	128	350	15 млн	400 млн	16/24/32	●	ICD	2048×1536	32	2D/3D
3Dlabs	Permedia 2	0,35	4–8	SGRAM, SDRAM, 70—125	AGP/PCI	●	64	70–125	64	230	1 млн	83 млн	16	●	ICD	1024×768	24 (Z 16)	2D/3D
	Permedia 3	0,25	4–16	SGRAM, SDRAM,200	AGP/PCI	Delta	128	140	128	270	8 млн	250 млн	32	●	ICD	1920×1200	32	2D/3D

Примечание. В таблице указаны максимальные значения разрядности z-буфера, разрешение и глубина цвета в 3D

Таблица 3.2. Основные характеристики наиболее распространенных Chipset для 3D-видеоплат (состояние на 1999г.).

GeForce3 действительно удался: построенная на его основе ASUS AGP-V8200/Pure значительно обгоняла плату предыдущего поколения Gigabyte GA-GF2000 на GeForce2 GTS — на 15-40% в 16-бит и на 30-60% в 32-бит цвете.

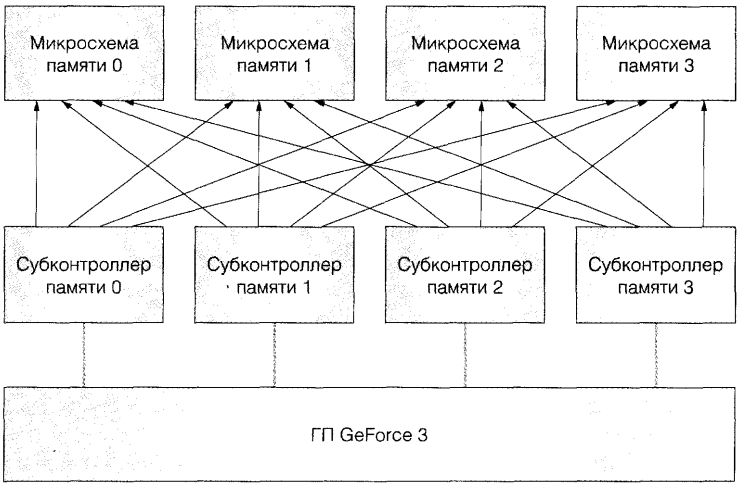


Рис. 3.46. Блок-схема «кроссового» контроллера видеопамяти ГП GeForce3 и GeForce4.

Позднее NVIDIA выпустила «разогнанную» модификацию этого ГП, GeForce3 Ti 500, и «оригинальный» GeForce3 утратил лидирующие позиции, а с появлением ГП модельного ряда GeForce4 Ti и вовсе перешел в разряд изделий для игровых плат среднего класса. (Блок-схема ГП GeForce4 Ti представлена на рис. 3.47.) Так, на игровых тестах без использования ФС (фильтрация сглаживания - antialiasing) и АФ (анизотропная фильтрация) уже ASUS AGP-V8200/Pure проигрывала в скорости плате на базе GeForce4 Ti 4400 до 65%, при включении ФС — до 80%, а по интегральному показателю 3DMark Score отставала примерно на 30% (при разрешениях 1280×1024 и 1600×1200).

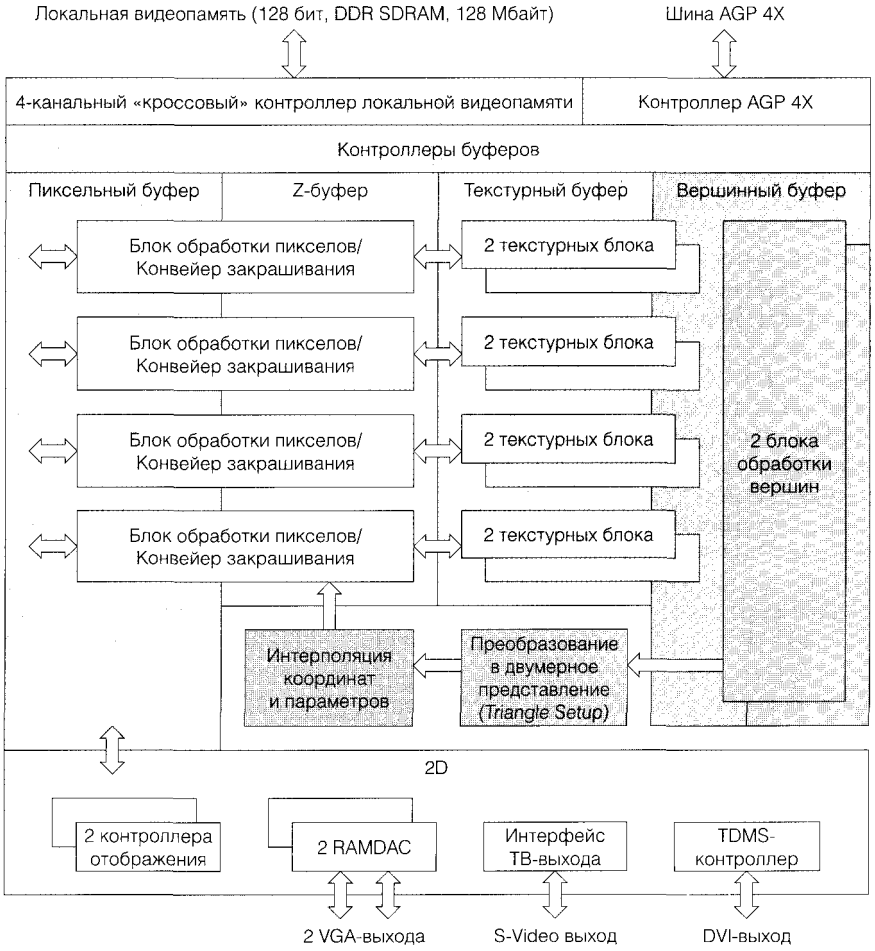


Рис. 3.47. Блок-схема ГП GeForce4 Ti

Одним из новейших по состоянию на 2002 года конкурентом ГП GeForce4 Ti является графический процессор RADEON 8500 фирмы ATI (см. рис 3.48). Среди основных новшеств этого ГП можно отметить: аппаратную реализацию новой технологии разбиения гладких поверхностей на треугольники, известную под фирменным названием Truform (с помощью так называемых N-патчей), адаптивное сглаживание (Adaptive Antialiasing) и улучшенную технологию сжатия данных Z-буфера (HyperZ II). Кроме того в ГП предусмотрены: наличие блоков обработки вершин и пикселов, наличие встроенного интерфейса ТВ-выхода и возможность использования второго внешнего ЦАП для подключения аналогового монитора и второго внешнего TDMS-контроллера для подключения монитора с DVI-интерфейсом.

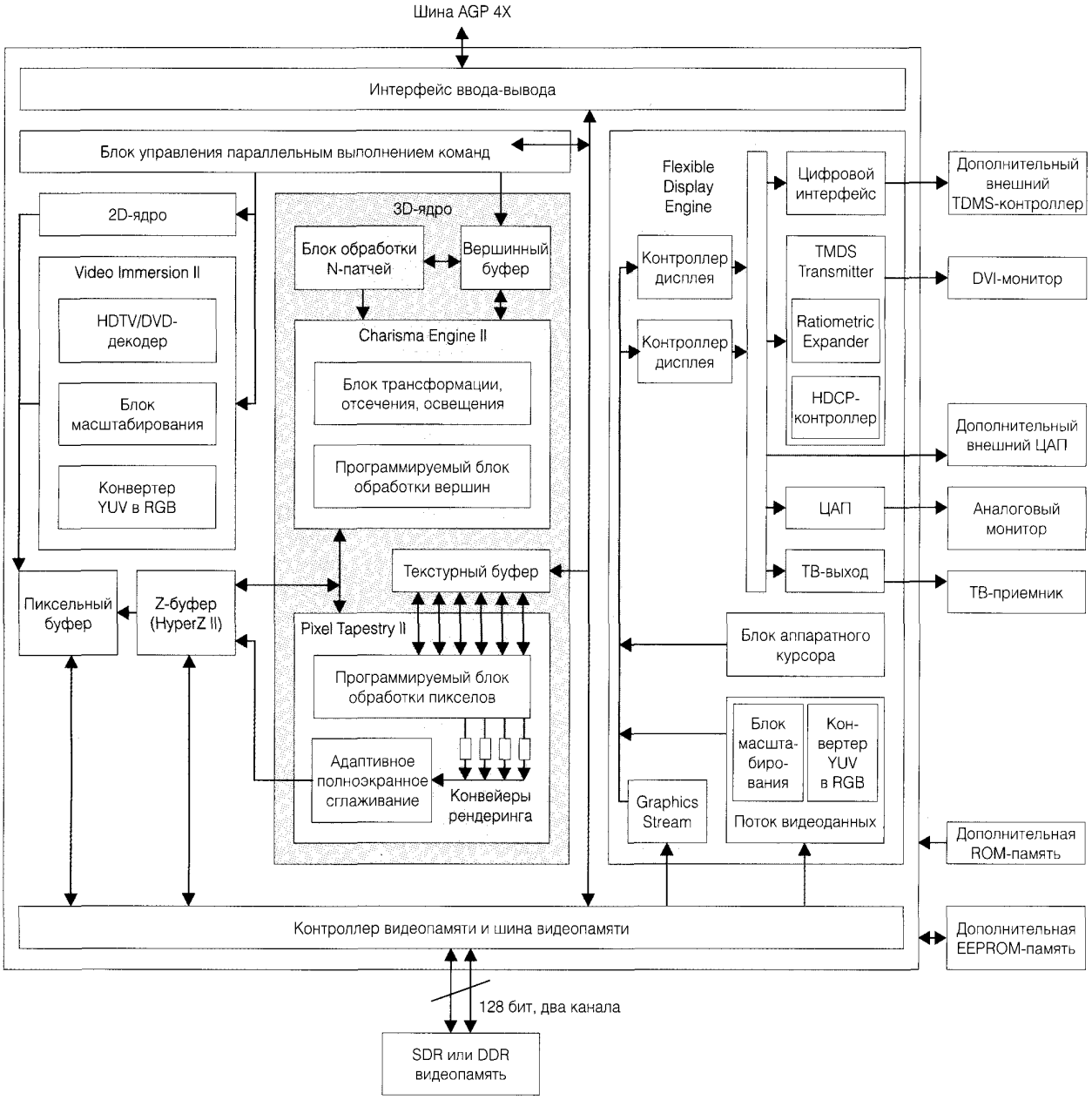


Рис. 3.48. Структурная схема ГП RADEON 8500

Как можно видеть из сравнения структурной схемы ГП Radion 256 (рис. 3.39) и структурной схемы ГП Radion 8500 (рис.3.x2x) за несколько лет развития технологий архитектура этих ГП не претерпела каких-либо значительных изменений. Реализация основных выше перечисленных новшеств проводилась на базе 0.15мкм технологии и потребовала довести число транзисторов в чипе до 60 миллионов, что в несколько раз больше , чем у чипа ГП Radion 256.

В таблице 3.3 приведены некоторые параметры графических процессоров (Chipset), которые в настоящее время широко используются в мультимедийных видеокартах.

Таблица 3.3. Графические процессоры (состояние на 2002 год).

Параметры	ATI RADEON 7500LE	ATI RADEON 8500	Matrox G550	NVIDIA GeForce2 GTS	NVIDIA GeForce3 Ti 200
Технология изготовления, мкм	0,15	0,15	0,18	018	0,15
Число транзисторов, млн	30	60	Н/Д	25	57
Тактовая частота ядра, МГц	230-255	275	133	200	175
Физическая (эффективная) тактовая частота видеопамяти, МГц	166-175	275 (550)	166 (333)	166 (333)	200 (400)
Видеопамять (ширина шины, тип, максимальный объем)	64 или 128 бит, SDR или DDR SDRAM, 64 Мбайт	128 бит, SDR или DDR SDRAM, 128 Мбайт	64 бит, SDR или DDR SDRAM, 32 Мбайт	128 бит, SDR или DDR SDRAM или SGRAM, 128 Мбайт	128 бит, DDR SDRAM или SGRAM, 128 Мбайт
Тактовая частота RAMOAC, МГц	2x350	400+внеш-ний 240	360 и 230	350 МГц	350
Число конвейеров рендеринга	2	4	2	4	4
Число текстурных блоков на каждом конвейере рендеринга	3	2	2	2	2
Число блоков трансформации и расчета освещенности (T&L)	1	1	1	1	1
Блок обработки вершин	Нет	Версия 1 1	Есть	Нет	Версия 1.1
Блок обработки пикселов	Нет	Версия 1 4	Нет	Есть	Версия 1.1

Н/Д — нет данных

Таблица 3.3. Графические процессоры (состояние на 2002 год) (продолжение).

Параметры	NVIDIA GeForce3	NVIDIA GeForce3 T1 500	NVIDIA GeForce4 MX 420	NVIDIA GeForce4 MX 440	NVIDIA GeForce4 MX 460
Технология изготовления, мкм	0,15	0,15	0,15	0,15	0,15
Число транзисторов, млн	57	57	Н/Д	Н/Д	Н/Д
Тактовая частота ядра, МГц	200	240	250	270	300
Физическая (эффективная) тактовая частота видеопамяти, МГц	230 (460)	250 (500)	166	200 (400)	275 (550)
Видеопамять (ширина шины, тип, максимальный объем)	128 бит, DDR SDRAM или SGRAM, 128 Мбайт	128 бит, DDR SDRAM или SGRAM, 128 Мбайт	128 бит, SDR SDRAM, 64 Мбайт	128 бит, DDR SDRAM, 64 Мбайт	128 бит, DDR SDRAM, 64 Мбайт
Тактовая частота RAMOAC, МГц	350	350	2x350	2x350	2x350
Число конвейеров рендеринга	4	4	2	2	2
Число текстурных блоков на каждом конвейере рендеринга	2	2	2	2	2
Число блоков трансформации и расчета освещенности (T&L)	1	1	1	1	1
Блок обработки вершин	Версия 1.1	Версия 1,1	Версия 1,1	Версия 1.1	Версия 1.1
Блок обработки пикселов	Версия 1.1	Версия 1.1	Нет	Нет	Нет

Н/Д — нет данных

Таблица 3.3. Графические процессоры (состояние на 2002 год) (продолжение).

Параметры	NVIDIA GeForce4 Ti 4400	NVIDIA GeForce4 Ti 4600	STMicroelectronics/ Imagination Technologies KYRO II
Технология изготовления, мкм	0,15	0,15	0,18
Число транзисторов, млн	63	63	15
Тактовая частота ядра, МГц	275	300	175
Физическая (эффективная) тактовая частота видеопамяти, МГц	275 (550)	325 (650)	175
Видеопамять (ширина шины, тип, максимальный объем)	128 бит, DDR SDRAM, 128 Мбайт	128 бит, DDR SDRAM, 128 Мбайт	128 бит, SDR SDRAM, 64 Мбайт
Тактовая частота RAMOAC, МГц	2x350	2x350	300
Число конвейеров рендеринга	4	4	2
Число текстурных блоков на каждом конвейере рендеринга	2	2	1
Число блоков трансформации и расчета освещенности (T&L)	2	2	0
Блок обработки вершин	Версия 1.1	Версия 1.1	Нет
Блок обработки пикселов	Версия 1.3	Версия 1.3	Нет

Контрольные вопросы

1. Перечислите чипсеты фирмы 3Dfx.
2. Охарактеризуйте параметры чипсета Voodoo Graphics.
3. Охарактеризуйте чипсет Voodoo Ruch.
4. Какие микросхемы входят в состав чипсета Voodoo^2 и с какой конфигурацией памяти они могут работать?
5. Какие возможности дает применение второй микросхемы?
6. Какой процессор оптимален для Voodoo^2?
7. Каковы основные характеристики Voodoo^2?
8. Какова идея создания чипсета Voodoo Banshee и что входит в его состав?
9. Почему Voodoo Banshee не поддерживает мультитекстурирование и трилинейную фильтрацию?
10. На какую память рассчитан Voodoo Banshee?
11. Чем Voodoo ^{3} отличается от своих предшественников и сколько модификаций имеет?
12. Какими дополнительными возможностями обладает чипсет Voodoo ^{3}?
13. Каковы основные недостатки чипсета Voodoo ^{3}?
14. Дайте краткую характеристику чипсету i740 фирмы Intel.
15. С какой памятью, какого объема и на какой частоте работает i740?
16. Какой RAMDAC содержит i740?
17. Какие 3D-функции реализованы в i740?
18. Перечислите чипсеты фирмы nVidia.
19. Дайте краткую характеристику видеоадаптеров, выполненных на чипсете Riva 128.
20. Перечислите характеристики чипсета Riva 128, определяющие его быстродействие.
21. На какой API сориентированы поставляемые с чипсетом Riva 128 драйверы?
22. Каковы основные отличительные особенности чипсета Riva 128ZX?
23. Дайте краткую характеристику чипсету Riva 128TNT.
24. Что, наряду с функциями 3D-акселерации обеспечивает Riva 128TNT?
25. Какими показателями характеризуется высокая производительность Riva TNT?
26. Какие дополнительные возможности имеет Riva TNT по обработке видеоданных?
27. Каковы главные отличительные особенности чипсета Riva TNT2?
28. Как изменились частоты ядра и памяти у Riva TNT2?
29. Какие чипсеты, помимо базового, входят в семейство Riva TNT2?
30. Какова отличительная особенность Riva TNT2-A?
31. Какова отличительная особенность Riva TNT2-Ultra?
32. Охарактеризуйте отличия Riva TNT2-Vanta.
33. Охарактеризуйте отличия Riva TNT2-M64.
34. В чем основная особенность чипсетов семейства GeForce?
35. Каковы другие отличительные особенности чипсетов семейства GeForce?

36. Охарактеризуйте параметры чипсета GeForce 256 (NV10).
37. Охарактеризуйте особенности чипсета GeForce 2.GTS (NV15 и NV11).
38. Чем отличается вариант NV15 от NV11?
39. Перечислите основные чипсеты фирмы Matrox.
40. Охарактеризуйте общие параметры чипсета MGA G200.
41. Какими недостатками обладает чипсет MGA G200?
42. Охарактеризуйте видеоадаптеры на основе чипсета MGA G200.
43. Какие разновидности видеоадаптеров выпускаются на базе чипсета MGA G400 и в чем заключаются основные отличия G400 и G400 MAX?
44. Какой видеоадаптер, выполненный на базе MGA G400, поддерживает подключение двух мониторов или монитора и телевизора?
45. Какой чипсет является первой удачной разработкой фирмы ATI, и каковы его особенности?
46. Какие чипсеты фирмы ATI относятся к более поздним ее разработкам?
47. Охарактеризуйте параметры чипсета ATI Rage 128.
48. Чем чипсет ATI Rage 128 Pro отличается от предыдущего?
49. Каковы особенности видеоадаптера ATI Rage Fury MAXX?
50. Каковы особенности чипсета ATI Radion 256?
51. В чем особенность технологии MAXX?
52. Какие чипсеты для 3D-устройств выпускала и выпускает фирма S3?
53. Какие задачи позволяет решать технология сжатия текстур S3TC?
54. Охарактеризуйте аппаратные особенности чипсета Savage 3D фирмы S3.
55. В каких трех модификациях выпускался чипсет S3 Savage 4?
56. Охарактеризуйте основные отличия Savage 4 от Savage 3D.
57. Охарактеризуйте недостатки и достоинства чипсета S3 Savage 4.
58. Охарактеризуйте чипсет S3 Savage 2000.
59. Охарактеризуйте продукцию фирмы 3Dlabs.
60. Охарактеризуйте графические процессоры рядов GeForse 3 и GeForse 4 фирмы NVIDIA.
61. Охарактеризуйте графический процессор Radion 8500 фирмы ATI.

В данном разделе использована информация из [1, 8]. Более свежую и полную информацию о современных графических процессорах и о выпускаемых на их основе видеокартах можно найти в соответствующих периодических изданиях (Компьютер Пресс, PC Magazine Russian edition и др.), а также на серверах фирм изготовителей (см. Приложение 1).

Приложение 1

Список WWW-адресов производителей видеоадаптеров

3Dfx Interactive	http://www.3dfx.com
3Dlabs	http://www.3dlabs.com
AccelGraphics	http://www.accelgraphics.com
Alliance Semiconductor	http://www.alsc.com
Ark Logic	http://www.arklogic.com
Artist Graphics	http://www.bayriver.com
ASKA (Silver Star)	http://www.silverstar.com.hk
Asus	http://www.asus.com
ATl Tech.	http://www.atitech.com
A-Trend	http://www.a-trend.com
Avance Logic	http://www.avance.com
BioStar	http://www.biostar.com.tw
Boca Research	http://www.bocaresearch.com
Britek	http://www.britek.com.tw
Brooktree	http://www.brooktree.com
California Graphics	http ://www. calgraph .corn
Canopus Corporation	http://www.canopuscorp.com
Cardex	http://www.gainward.com
Chromatic Research	http://www.chromatic.com
Chips&Technologies	http://www.chips.com
Cirrus Logic	http://www.cirrus.com
Colorgraphic	http://www.colorgfx.com
Compaq	http://www.compaq.com
Cornerstone	http://www.corimage.com
Creative Labs	http://www.creativelabs.com
Crystal Semiconductor	http://www.crystal.com
Data Expert	http://www.dataexpert.com
Datapath Technologies	http://www.datapath.co.uk
Deltron Technology (Real Vision)	http://www.deltrontech.com
Densan	http://www.densan.com
Dynamic Picture	http://www.dypic.com
DFI	http://www.dfiusa.com
DTK Computer	http://www.dtk.com.tw
Diamond Multimedia Systems, Inc.	http://www.diamondmm.com
D System	http://www.dgroup.de
Dynamic Pictures	http://www.dypic.com
Elecede Technologies (E4)	http://www.e4.com
ELSA Inc.	http://www.elsa.com
Eontronics	http://www.eontronics.com
EPS Technologies	http://www.epstech.com
Evans & Sutherland	http://www.es.com
ExpertColor Multimedia	http://www.expertcolor.com
Gainward (Cardex)	http://www.gainward.com
Gateway 2000	http://www.gw2k.com
Genoa	http://www.genoasys.com
Guillemot (GI)	http://www.guillemot.com/uk
Hercules	http://www.hercules.com
Intergraph Computer Systems	http://www.intergraph.com
IGS Technologies, Inc.	http://www.igst.com
IXMICRO	http://www.ixmicro.com
InnoVISION	http://www.ivmm.com
Intel (i740)	http://www.intel.com
I/O Magic	http://www.iomagic.com
Jaton	http://www.jaton.com

Jazz Multimedia	http://www.jazzmm.com
Joytech	http://www.joytech.com
Leadtek	http://www.leadtek.com
Legend	http://www.legend.com.au
Macronix (MXIC)	http://www.macronix.com
Matrox Group	http ://www. matrox. corn
MaxTech	http://www.maxcorp.com
MaxVision	http://www.maxvision.com
MediaVision	http://svtus.com
Metabyte	http://www.metabyte.com
Microstep	http://www.microstep.com
MIRAGE Video Solution's	http://www.mirage-mmc.com
miro Computer Products AG	http://www.miro.com
MiroMEDIA	http://www.miro.de
Mitsubishi	http://www.3dpro2mp.com
Multimedia Access Corporation	http://www.osprey.mmac.com
NEC	http://www.nec.co.jp
NeoMagic Corporation	http://www.neomagic.com
NetPower	http://www.netpower.com
Newer Technology	http://www.newertech.com
NVidia Corporation	http://www.nvidia.com
Number Nine	http://www.nine.com
OAK tech.	http://www.oaktech.com
Omnicorp	http://www.omnicorp.com
OPTi	http://www.opti.com
Orchid Technology	http://www.orchid.com
Packard Bell	http://www.packardbell.com
Paradise Graphics (Philips Semiconductors)	http://www.paradisemmp.com
Philips Semiconductor	http://www.semiconductors. philips.corn
Pixel Fusion	http://www.pixelfusion.com
PowerColor	http://www.powercolor.com.tw
PowerVR (Nec совместно с VideoLogic)	http://www.powervr.com
Quanta	http://www.quantacorp.com
Quantum 3D	http://www.quantum3d.com
Radius	http://www.radius.com
Real 3D	http://www.real3d.com
RealTek	http://www.realtek.com.tw
RealVision	http://www.deltrontech.com
Rendition, Inc.	http://www.rendition.com
S3,Inc.	http://www.s3.com
SGS-Thomson	http://www.st.com
Sierra Online	http://www.sierra.com
Sigma Design	http://www.sigmadesigns.com
Silicon Reality	http://www.sireal.com
SiS	http://www.sis.com.tw
Skywell	http://www.skywell.com.tw
S-MOS	http://www.smos.com
STB Systems, Inc.	http://www.stb.com
Symmetric	http://www.symmetric.com
Tekram	http://www.tekram.com
Trident Microsystems	http://www.tridentmicro.com
TriTech	http://www.tritechmicro.com
Tseng Labs	http://www.tseng.com
VideoLogic	http://www.videologic.com
VideoLogic	http://www.videologic.com
Vogue 3D	http://www.vogue3d.com
Xionics, Inc.	http://www.xionics.com

Приложение 2

Список терминов и сокращений

Английские

α-буфер - см. альфа-буфер.

2D-графика - двумерная графика (2-*Dimentional* — двумерный).

3D RAM - память для ускорителей трехмерной графики. Разработана фирмой Mitsubishi для совместного использования с Chipset 3Dpro ее же производства. Этот Chipset основан на использовании передовой технологии REALImage фирмы Evans&Sutherland и предназначен для работы с трехмерной графикой. Память 3D RAM является двухпортовой и имеет разветвленную архитектуру, позволяющую осуществлять конвейерную (параллельную) обработку данных. Кроме того, предусмотрено наличие встроенного процессора и скоростных видеобуферов.

3D-графика - трехмерная графика *(3-Dimentional* — трехмерный).

3D-конвейер - процесс расчета трехмерного изображения объекта (точнее, его двумерной проекции). Он выполняется в несколько последовательных этапов. В общем виде можно выделить 13 этапов 3D-конвейера: Этапы 1 — 6 образуют так называемую *геометрическую стадию* 3D-конвейера. На этой стадии выполняются интенсивные тригонометрические вычисления, в большинстве случаев центральным процессором PC, так как он содержит математический сопроцессор с аппаратной поддержкой необходимых функций: вычисление значений тригонометрических функций, обработка чисел с плавающей точкой и др. Лишь профессиональные модели 3D-акселераторов содержат специальный *геометрический процессор,* частично или полностью выполняющий все операции по расчету геометрии объекта. Этапы 7—13 образуют *стадию прорисовки объекта,* или *стадию рендеринга (rendering*— рисование, визуализация). На этой стадии все действия выполняются уже с растровыми объектами, состоящими из отдельных дискретных элементов — пикселов и текселов.

Access time - временем доступа. Время, необходимое для чтения (записи) данных, хранящихся по случайному адресу памяти.

Alpha pixel blending (alpha blending) - см. альфа-смешение.

ANSI (American National Standardization Institute) - код, разработанной американским национальным институтом стандартизации для представления символьной информации в среде MS Windows.

APA (All Point Addressable - все точки адресуемы) - графический режим работы видеоадаптера.

API (Application Program Interface - интерфейс прикладного программирования) - специальный пакет служебных программ, занимающий промежуточное положение между высокоуровневыми прикладными программами и низкоуровневой программой — драйвером акселератора, который имеет собственную систему команд. Использование API избавляет разработчика прикладной программы от необходимости работать с низкоуровневыми командами, облегчая процесс создания программ и делая их более универсальными.

ASCII (American Standard Code for Information Interpretation) - американский стандартный код; используется в операционной системе MS DOS.

AVI (Audio-Video Interleaved) - формат видеосигнала, в котором сжатие видео- и аудиоинформации практически отсутствует.

BEDO DRAM (Burst Extended Data Output DRAM) - пакетная память EDO DRAM. В микросхему BEDO DRAM добавлен специальный счетчик слов, благодаря чему считывание из памяти четырех соседних слов осуществляется автоматически, даже тогда, когда необходимо считать лишь одно из них. На частоте системной шины PC 66 МГц временная диаграмма BEDO DRAM имеет вид 7-1-1-1, в то время как для FPM — 7-3-3-3, а для EDO— 7-2-2-2.

Bit Blt (Bit Block transferring) - перемещение блока бит из одной области видеопамяти в другую.

CAS (Column Address Strobe) - строб адреса столбца DRUM.

CDRAM (Cashed DRAM) - кэшированная динамическая память; также является разработкой фирмы Mitsubishi. Представляет собой комбинацию динамической памяти и небольшого скоростного буфера (с временем доступа 10-15 нс), выполненного на элементах статической памяти, традиционно используемой в качестве кэш-памяти. В частности, имеются модули памяти CDRAM, содержащие 4 или 16 Мбит динамической памяти и 16 Кбит статической. Подобно 3D RAM, CDRAM предназначена для использования в 3D-ускорителях (в качестве памяти для текстур). Кроме того, CDRAM используется в качестве кэш-памяти в скоростных винчестерах с интерфейсом Fibre Channel. CDRAM имеет 128-разрядную внутреннюю шину данных и 16-разрядную внешнюю, благодаря чему данные считываются из памяти со скоростью около 800 Мбайт/с.

CGA (Color Graphics Adapter) - цветной графический адаптер. Он обеспечивал цветное изображение (от 4 до 16 цветов) и несколько графических режимов; работал с цветным TTL-монитором (цифровым цветным монитором).

Clip volume - объем отсечения. Представляет собой параллелепипед, задающий границы трехмерной сцены.

Color Dithering (дизеринг) - метод интерполяция цветов, используемый тогда, когда текущий режим

работы видеоадаптера не позволяет отобразить требуемое количество оттенков цветов (имеет недостаточный объем видеопамяти). Метод заключается в получении недостающих цветов за счет смешения имеющихся. Суть метода подобна идее, положенной в основу работы цветного монитора: для создания цвета пиксела экрана используется триада зерен люминофора основных цветов. При дизеринге цвета соседних пикселов изображения выбираются из доступной палитры таким образом, чтобы в результате их смешения создавалась иллюзия появления нового цвета, которого нет в исходной палитре. Очевидно, что при этом немного ухудшается разрешение изображения. Этот метод широко распространен и используется не только в трехмерной графике, но и во многих программах обработки растровых изображений.

Conventional Memory *(обычная* или *стандартная память)* - часть адресного пространства IBM PC в диапазоне 00000— 9FFFFh (первые 10 сегментов10×2^{16} = 640 К 8-разрядных ячеек памяти) отведеная для обращения к оперативной памяти: процессором.

CPU (***Central Processing Unit)*** - центральный процессор.

CRT (***Cathode Ray Tube*** - лампа с катодными лучами) - английский вариант названия электронно-лучевой трубки (ЭЛТ).

CRTC (Cathode Ray Tube Controller) - контроллер ЭЛТ.

Culling - отбрасывание задних граней из проекции трехмерного объекта.

Culling - отбрасывание задних, невидимых граней при проецировании трехмерного объекта на плоскость.

DDC (Display Data Channel) - канал данных монитора по которому происходит обмен данными между монитором и видеоадаптером.

DDR SDRAM (Double Data Rate SDRAM) - память SDRAM с удвоенной скоростью передачи данных. Другое название этого типа памяти — SDRAM II. Разработка компании Samsung. Основное отличие этой памяти от ее прототипа SDRAM заключается в том, что чтение данных осуществляется как по фронту, так и по спаду тактового сигнала, что позволяет выполнять два обращения к памяти за время одного цикла обращения к «стандартной» SDRAM.

Direct RDRAM (Direct Rambus DRAM) - динамическая память с прямым доступом, производства американской фирмы Rambus, на сегодня — одна из самых быстродействующих. В настоящее время память этого типа выпускают несколько компаний, в том числе NEC, Fujitsu и Toshiba. В микросхемах Direct RDRAM сохранились старые принципы записи/чтения данных. Изменения затронули лишь организацию доступа к ячейкам памяти. Технология Rambus предусматривает наличие 16-разрядной высокоскоростной шины Rambus Channel, тактовая частота которой составляет 400 МГц, а скорость передачи данных — 1,6 Гбайт/с.

DirectX - API, разработанный фирмой Microsoft для области игровых приложений, ориентированных на работу под управлением операционной системы Windows 95/98. Включает в себя несколько специализированных API: DirectDraw; обеспечивает использование аппаратных средств ускорения обычной, двумерной графики; Direct3D; отвечает за работу графической системы в режиме создания трехмерных изображений; DirectInput; обеспечивает аппаратно независимый ввод информации в PC с помощью клавиатуры, мыши и джойстика; DirectPlay; используется при совместной игре на нескольких компьютерах, объединенных в сеть или соединенных непосредственно — через параллельный или последовательный порт; DirectSound; управляет использованием ресурсов звуковой системы PC.

Dithering - смотри *"Дизеринг".*

Dot matrix - точечная матрица, задающая изображение символа в пределах каждого знакоместа.

Dot rate или ***pixel rate*** - скорость вывода пикселов на экран. Это частота, с которой происходит опрос ячеек кадрового буфера, содержащих коды цвета пикселов, определяет скорость прорисовки изображения на экране монитора.

DRAM (Dynamic Random Access Memory) - динамическая память с произвольным доступом. Ячейка динамической памяти построена на основе конденсатора, хранящего электрический заряд. Если конденсатор заряжен, то считается, что в ячейку записана логическая единица, если разряжен — логический ноль. Поскольку емкость конденсатора очень мала (доли пикофарады) и имеется ток утечки, информация в такой ячейке может храниться только в течение короткого промежутка времени, а при считывании данных конденсатор разряжается чуть ли не мгновенно. Для длительного хранения информации в такой ячейке необходимо периодически, а при считывании — каждый раз проводить *регенерацию* ее содержимого. Регенерация осуществляется путем выполнения (примерно через каждые 10 мс) операций считывания/записи всех хранящихся в памяти данных.

EDO DRAM (Extended Data Output DRAM) - память DRAM с увеличенным временем удержания данных на линиях ввода/вывода. В отличие от FPM DRAM, в которой линии ввода/вывода отключаются от шины данных при задании адреса столбца следующей ячейки, в EDO DRAM данные доступны для считывания во время передачи нового адреса. Таким образом, при работе с памятью EDO DRAM возможно одновременное считывание данных и задание адреса следующих, что позволяет повысить быстродействие при чтении на 10—15% по сравнению с быстродействием FPM DRAM и на 30% — по сравнению с DRAM. Также как и в случае использования FPM DRAM, выигрыш в быстродействии имеет место лишь при обращении к данным, расположенным последовательно.

EGA (Enhanced Graphics Adapter) - улучшенный графический адаптер. Максимальное разрешение EGA составило 640×350, размер видеопамяти (64 - 128) Кбайт, поддержка 16 цветов из палитры 64 цвета. Для управления цветом изображения на мониторе используется 6 цифровых линий.

ESDRAM (Enhanced SDRAM) - улучшенная память SDRAM. Разработка фирмы Enhanced Memory Systems. Более высокого быстродействия элементов памяти ESDRAM, по сравнению с SDRAM и даже DDR SDRAM, удалось добиться благодаря использованию более совершенной архитектуры и встроенной статической (SRAM) кэш-памяти (подобно тому, как это реализовано в элементах памяти CDRAM). Кроме того, за счет реализации алгоритма «упреждающего чтения» из ячеек SDRAM в более быстрые регистры SRAM, по аналогии с технологией кэширования жестких дисков, значительно повышена скорость считывания информации.

Extended Memory (расширенная память) - область памяти с адресами выше FFFFFh (1Мбайт) не доступная процессору i80286 и старше, работающему в режиме реальной адресации.

Fill Rate - максимальная скорость закраски, измеряемая количеством пикселов в секунду.

FPM DRAM (Fast Page Mode DRAM) - память DRAM с быстрым страничным обменом. Память FPM DRAM разбита на участки, именуемые страницами. В пределах страницы полный адрес ячейки памяти состоит из двух частей: адреса строки и адреса столбца. Размер страницы может составлять от 512 байт до нескольких килобайт. При считывании данных, расположенных последовательно в пределах одной строки, отпадает необходимость в повторном формировании сигнала RAS, благодаря чему удается значительно ускорить чтение данных.

FPS (frame per second) - частота смены кадров.

Game Engine - «игровой движок». Игровой драйвер, разработанный для конкретного 3D-ускорителя и ориентированный на достижение максимальной производительности за счет непосредственного использования низкоуровневых команд акселератора в обход API (разновидность минипорта).

GUI (Graphic User Interface) - стандартный графический интерфейс пользователя в среде Windows.

Hardware character generator (аппаратный знакогенератор) - это ПЗУ, размещенное на плате видеоадаптера, в котором хранится графическое изображение символов в виде последовательности двоичных чисел; в качестве адреса ячейки знакогенератора используется порядковый номер символа.

High Color - видеорежим с прямым кодированием цвета пиксела (16 бит на пиксел, или 65 536 оттенков цветов).

HSR (Hidden Surface Removal) - смотри "*Удаление скрытых поверхностей".*

ICD (Installable Client Driver) — драйвер приложения-клиента. Полная версия OpenGL-совместимого драйвера.

IHV (Independent Hardware Vendor) - независимый производитель аппаратного обеспечения.

ISA (Industrial Standard Architecture - стандартная промышленная архитектура) - стандартная шина ввода/вывода персональных компьютеров. Эта 16-разрядная шина работает на тактовой частоте 8 МГц и впервые была реализована на материнских платах с процессором Intel 80286 (напомним, что разрядность шины данных этого процессора равна 16, а тактовая частота для его первых моделей — 8 МГц). Теоретическая пропускная способность шины ISA составляет 16×8000000 = 122 Мбит/с = 15,2 Мбайт/с. Однако, поскольку далеко не каждый такт сопровождается передачей данных, реальная пропускная способность шины ISA не превышает 3—8 Мбайт/с.

MCD (Mini Client Driver) - усеченная версия OpenGL-совместимого драйвера.

MCGA (Multi-Color Graphics Adapter) - многоцветный видеоадаптер. Разработан IBM для младших моделей IBM/PS. По возможностям аналогичен VGA.

MDA (Monochrome Display Adapter) - адаптер монохромного дисплея. Он работал только в текстовом режиме 80×25 символов. Размер символьной матрицы (знакоместа) составлял 9×14 пикселов, поэтому разрешение, поддерживаемое монитором MDA, составляло 720×350 пикселов. Размер самого символа составлял 7×9 пикселов. Емкость видеопамяти видеоадаптера MDA была минимальной и достаточной для размещения только одной видеостраницы размером 4 Кбайт.

MDRAM (Multibank DRAM) - мультибанковая динамическая память, разработанная фирмой MoSys Inc. и предназначенная для использования исключительно в системах обработки графики и видео изображений. Главной особенностью памяти MDRAM является ее необычная организация. MDRAM представляет собой набор независимых банков обычной динамической памяти; емкость каждого составляет 32 Кбайт. MDRAM можно организовать таким образом, что обращение к каждому банку будет независимым. А это — путь к несомненному повышению производительности видеоадаптера в целом, по сравнению с моделями, использующими двухпортовую память—VRAM или WRAM. Память типа MDRAM используется пока только в видеоадаптерах фирмы Tseng Labs (ET6000, ET6100).

MIP — Multum In Parvo - много в одном.

MIP-mapping - смотри *"Мипмеппинг".*

MIP-каскад - совокупность всех вариантов одной и той же текстуры.

MIP-текстурирование - смотри *"Мипмеппинг".*

MPEG (Motion Picture Experts Group) - экспертная группа по кинематографии. Группа, разработавшая одноименный алгоритм сжатия подвижного изображения. Согласно алгоритму сначала упаковывается неподвижное изображение, затем пересылается только его изменения. Коэффициент сжатия до 200:1.

NTSC (National Television System Committee) - североамериканский цветной телевизионный формат. Использует 525 линий разложения изображения, частоту смены полей 60 Гц, полосу пропускания канала 4 МГц, частоту строк 15,75 КГц, длительность кадра 1/30 с, частоту цветовой поднесущей 3,58 МГц. Особенностью телевизионной системы NTSC является то, что оба цветоразностных сигнала передаются одновременно в каждой строке развертки без расширения полосы частот сигнала цветности в спектре ТВ сигнала. Это достигается применением квадратурной модуляции, при которой результирующий цветовой сигнал модулируется по амплитуде и манипулируется по фазе. Амплитуда сигнала характеризует насыщенность цвета, а фаза - цветовой тон.

OpenGL - API, используемый в основном в сфере профессиональных приложений (САПР, системы трехмерного моделирования, симуляторы и т. п.), работающих под управлением операционной системы Windows NT. Разработан общепризнанным лидером в области мощных графических станций — компанией Silicon Graphics, Inc. (SGI).

OpenGL ARB (Architecture Review Board) — совет по пересмотру архитектуры OpenGL. В состав совета входят компании DEC, Evans & Sutherland (E&S), IBM, Intel, Intergraph, Microsoft, Silicon Graphic, Inc.

PAL (***Phase Alternation Line*** - система с построчным изменением фазы) - система с одновременной квадратурной модуляцией цветовой поднесущей. Ее основное отличие от NTSC - изменение от строки к строке на 180° фазы цветоразностных сигналов. При приеме ТВ-сигнала системы PAL на выходе модуля цветности TV-тюнера формируются сигналы *Y, U* и *V*. Используется в Германии и ряде других стран. Частоа строк, кадров и полей такая же, как и в SECAM. Частота поднесущей - 4,436 МГц.

PC (***Personal Computer)*** - персональный компьютер.

PCI (Peripheral Component Interconnect - соединение периферийных устройств) - стандартная шина ввода/вывода PC, выполненных на базе CPU Pentium и др. На современных материнских платах тактовая частота шины PCI (спецификация PCI 1.0) задается как 1/2 тактовой частоты системной шины, т. е. при тактовой частоте системной шины 66 МГц тактовая частота шины PCI будет 33 МГц. Поэтому ее пропускная способность составила 132 Мбайт/с (столько же, сколько у шины VLB). Однако с появлением новых версий спецификации PCI (PCI 2.0) тактовая частота была увеличена до 66 МГц, т. е. пропускная способность шины увеличилась вдвое. В настоящее время завершается работа над новой версией спецификации PCI, в соответствии с которой тактовая частота шины составит 100 МГц. Важной особенностью шины PCI является реализация принципа Bus Mastering — возможность управления шиной (или пересылкой данных) внешним устройством без участия центрального процессора. Во время передачи информации устройство, поддерживающее технологию Bus Mastering, захватывает шину и становится главным. Такой подход позволяет освободить центральный процессор от управления передачей данных для выполнения других задач. Еще одним свойством шины PCI является ее интеллектуальность, т. е. она в состоянии распознавать аппаратные средства и осуществлять анализ конфигурации системы в соответствии с технологией Plug&Play, разработанной корпорацией Intel.

PDF (***Packed Display Format)*** - формат пакетного (компактного) хранения содержимого кадрового буфера. Согласно формату PDF, номер цвета пиксела целиком хранится в одной 8-разрядной ячейке кадрового буфера.

Pixel Engine - механизм обработки итогового кадра, являющийся элементом конвейера рендеринга (наряду с *Texel Engine).*

RAM (Random Access Memory) — память с произвольным доступом

RAMDAC *(Random Access Memory Digital-to-Analog Converter)* - цифро-аналоговый преобразователь — ЦАП — данных, хранимых в ОЗУ. RAMDAC представлял собой быстродействующий трехканальный ЦАП с 256 регистрами, образующими его собственное маленькое ОЗУ — RAM. RAMDAC предназначен для преобразования двоичных чисел, содержащихся в ячейках видеопамяти, в три непрерывных RGB-сигнала, уровень которых пропорционален яркости каждого из трех основных цветов.

RAS (Row Address Strobe) - строб адреса строки DRAM.

Rendering - *смотри "Рендеринг".*

Rendering Engine - механизм прорисовки.

RGB - красный *(Red),* зеленый *(Green)*, синий *(Blue).* Разновидность кодирования цветного изображения.

ROM BIOS (Read Only Memory Base Input/Output System) - ПЗУ, содержащее *базовую систему ввода/вывода.*

S3TC — S3 Texture Compression - технология сжатия текстур, разработанная фирмой 3S.

SDRAM (Synchronous DRAM) - синхронная память. Принцип работы SDRAM такой же, как и DRAM, за исключением того, что для работы SDRAM необходим дополнительный тактирующий сигнал, частота которого кратна тактовой частоте процессора. Этим сигналом фиксируются моменты обновления информации на входах и выходах элементов памяти, т. е. процессор в зависимости от типа используемой синхронной памяти будет точно «знать», в какой момент времени данные появятся на ее выходных линиях. В этом случае процессор (видеоконтроллер) и память работают синхронно, без циклов ожидания, т. е. все операции по управлению памятью синхронизированы с тактовой частотой процессора. Теперь вместо циклов ожидания процессор может выполнять другие операции. Синхронная работа процессора и памяти позволила исключить ситуацию, когда команды чтения/записи совпадают по времени с процедурой регенерации памяти, а также уменьшить время выборки данных за счет реализации трехступенчатой конвейерной

адресации, позволяющей осуществить доступ к запрошенным данным до завершения обработки предыдущих. Благодаря этому удалось уменьшить длительность рабочего цикла до 7-10 нс и поднять частоту тактового сигнала до 125 МГц и выше.

SECAM (Sequentiel Couler a Memoire (фр.) - последовательные цвета и память) - последовательная, цветная, совместимая с черно-белой телевизионная система. Используется в России. Франции и ряде других стран. Ее отличительной особенностью является поочередная передача через строку двух цветоразностных сигналов при непрерывной передаче сигнала яркости. Частота смены полей - 50 Гц, частота кадров - 25 Гц. Частота строк 15625 Гц, частота поднесущей: для строк B - 4,425 МГц; для строк R - 4,40625 МГц.

SGRAM (Synchronous Graphics RAM) - синхронная графическая память имеет много общего с памятью типа SDRAM. Память SGRAM может работать на частотах 66 МГц и более. По сравнению с DRAM, быстродействие SGRAM возросло более чем в четыре раза. Память SGRAM предназначена для хранения изображений, т. е. для использования в видеоадаптерах.

Software character generator (программный знакогенератор) - специально отведенная область видеопамяти в которую копируется содержимое знакогенератора; содержимое этой памяти можно корректировать, изменяя тем самым начертание символов. На этом принципе основана технология, позволяющая загружать шрифты национального алфавита.

SRAM - статическая память, ячейка которой представляет собой триггер. Быстродействие такой памяти оказывается в несколько раз выше (среднее время доступа 15-20 нс), чем DRAM. Однако и стоимость ее изготовления также значительно выше. Поэтому статическую память обычно используют в качестве скоростного буфера небольшой емкости — *кэш-памяти,* — в котором хранятся наиболее часто используемые данные.

Tesselation - *смотри " тесселяция".*

Texel— (TEXture Element) - смотри *"Тексел".*

Texel Engine - механизм обработки элементов текстур, являющийся элементом конвейера рендеринга (наряду с *Pixel Engine).*

Triangle Setup - смотри "*Обработка (настройка) координат проекций вершин элементарных треугольников."*

Triangle Throughput - пропускная способность 3D-ускорителя, измеряемая количеством обрабатываемых элементарных треугольников в секунду.

True Color - видеорежим с прямым кодированием цвета пиксела (24 бит на пиксел, или 16 777 216 оттенков цветов).

TV-тюнер - устройство, которым достаточно снабдить компьютер для приема телевизионных программ. TV-тюнер включает в себя приемник телевизионного радиосигнала, декодер (модуль цветности), а также устройство преобразования типа развертки.

UMA (Upper Memory Area - верхней памятью) - часть адресного пространства IBM PC в диапазоне A0000—FFFFFh (верхние шесть сегментов, т. е. 6×2^{16}= 384 К) зарезервированные для обращения к внешним устройствам, в том числе и к видеоадаптеру.

VBE(VESA BIOS Extension - VESA-расширение BIOS) - программы, дополняющие Video BIOS видеоадаптера SVGA и обеспечивающие поддержку спецификации VESA. Подобно стандартному Video BIOS VBE содержит программы, реализующие *дополнительные графические функции,* необходимые в видеорежимах VESA. Помимо управления изображением в этих режимах, функции VBE используются для поддержки канала данных (DDC) монитора, управления энергопотреблением, сопряжения видеоадаптера с плоскопанельными мониторами и др.

VESA (Video Electronics Standards Association) - Ассоциация Стандартов по Видеоэлектронике; название ассоциации инженеров по видеоэлектронике. Известна разработкой стандарта на локальную шину VESA и рекомендациями, которые позволили разработать *стандарт SVGA.*

VESA-режимы - двухбайтная нумерация расширенных видеорежимов. Явилась для различных производителей видеоадаптеров SVGA удобным компромиссом: с одной стороны, все они сохранили свою оригинальную нумерацию SVGA-режимов, используя для них однобайтные номера в пределах 14h - 7Fh, с другой — каждый SVGA-режим получил второй, единый для всех видеоадаптеров номер, состоящий из двух байт. Двухбайтные номера VESA-режимов начинаются с номера 100h. Видеоадаптер может устанавливать требуемый видеорежим при указании одного из двух номеров — собственного однобайтного или стандартного двухбайтного.

VGA (Video Graphics Adapter) - графический адаптер, формирующий видеосигнал (т. е. аналоговый сигнал). Видеоадаптер VGA имел 256 Кбайт видеопамяти. Это обеспечило поддержку графических режимов 640×480/16 (640×480 при 16 цветах) и 320×200/256. Максимальный размер палитры определялся разрядностью регистров RAMDAC. В стандартном видеоадаптере VGA каждый из трех каналов RAMDAC являлся шестиразрядным, что обеспечивало палитру из $2^{6\times3}$ = 262144 оттенка цветов. Такая палитра соответствует возможностям органа зрения человека по цветоощущению. Однако из-за ограниченного объема видеопамяти для кодирования цвета каждого пиксела изображения может быть отведено не более 8 бит, поэтому видеоадаптер мог отображать лишь 2^8 = 256 оттенков из указанной палитры.

Video BIOS - это набор подпрограмм, написанных в кодах команд центрального процессора и предназначенных для реализации основных функций видеосистемы. Набор этих функций принято называть

видеосервисом. Кроме того, Video BIOS содержит множество данных (констант), необходимых для работы в различных видеорежимах, а также сведения о производителе, модели и возможностях видеоадаптера. Video BIOS хранится в специальном ПЗУ (ROM), размещенном на плате видеоадаптера VGA. Его емкость составляет 32 Кбайт.

VLB (VESA Local Bus) - скоростная локальная шина ввода/вывода (133 Мбайт/с). Создана Ассоциацией стандартов по видеоэлектронике (VESA). В свое время оказалась наиболее удачным и недорогим решением проблемы повышения пропускной способности шины ввода/вывода. Локальная шина VESA представляет собой не новое устройство на материнской плате, а скорее расширение шины ISA для обмена видеоданными. Обмен информацией осуществляется под управлением контроллеров, расположенных на картах, устанавливаемых в слот VLB, напрямую с CPU, в обход стандартной шины ввода/вывода. Шина VLB является 32-разрядной и работает на тактовой частоте процессора. Кроме того, передача данных по этой шине невозможна без использования линий шины ISA, по которым передаются сигналы адресов и управления. Согласно спецификации VESA, тактовая частота локальной шины не должна превышать 40 МГц. Однако для большинства материнских плат, имеющих процессор с тактовой частотой 50 МГц, проблем обычно не возникало.

VRAM (Video RAM) - это динамическая память с двумя портами для одновременного выполнения операций чтения и записи. Для работы с такой памятью на плате видеоадаптера устанавливается специальный контроллер. Преимущества VRAM особенно заметны при работе в режимах с высоким разрешением (1280×1024 и выше) и 24-битным представлением цвета. Используются в основном в профессиональных графических системах.

Wireframe - *смотри "Каркасная модель объекта".*

WRAM (Window RAM) - это память, архитектура которой очень похожа на архитектуру VRAM, но стоимость изготовления WRAM ниже. Эта память может работать на частоте до 50 МГц, благодаря чему быстродействие повышается на 50% по сравнению с VRAM. Технология изготовления и схемотехника памяти типа WRAM ориентированы на конкретное применение в видеоадаптерах. В этой памяти имеется режим быстрой записи двухцветных блоков для ускорения пересылки текста и закраски областей пиксельными шаблонами *(Dual-Color Block Write),* а также реализована технология двойной буферизации *FastBit.* В соответствии с технологией FastBit, для двух смежных кадров предусмотрены два кадровых буфера. Пока первый кадр отображается (считывается в RAMDAC), CPU формирует следующий. В итоге обеспечивается быстрая смена кадров, что особенно важно при обработке видео и работе с трехмерной графикой.

Wrapper (Раппер) - минипорт, использующий перевода инструкций OpenGL в инструкции Direct3D. Из всех типов драйверов раппер обеспечивает самую низкую скорость работы 3D- акселератора.

YUV («формат YUV», «цветовое пространство YUV») - формат представления цветного изображения. Подразумевает замену сигналов *R, G, B* сигналом яркости *Y* и двумя цветоразностными сигналами *U и V.* Поскольку сигналы *U и V* имеют гораздо более узкий спектр по сравнению с сигналом яркости Y и сигналами *R, G, B,* преобразование изображения в цифровую форму в пространстве YUV обеспечивает значительное сжатие информации.

Z-алиасинг (z-aliasing) - искажение глубины отдельных элементов рисунка.

Z-буфер - совокупность данных о глубине каждой из вершин элементов поверхности трехмерного объекта (о расстоянии от плоскости, на которую он проецируется). Это расстояние выступает как третья координата проекции вершины.

Z-буферизация - При z-буферизации выполняется анализ всех z-координат элементов объекта в процессе рендеринга. Для этого, собственно, и нужен z-буфер. Из всех точек объекта с одинаковыми координатами *(x, y}* проецироваться будет только одна, у которой координата *z* имеет минимальное значение. В результате будут корректно отображаться объекты, образованные пересекающимися поверхностями.

Z-сортировка (z-sorting) — наиболее простой метод проецирования трехмерного объекта на плоскость, при котором целенаправленное удаление скрытых поверхностей специально не производится. Программа выполняет рендеринг всех элементов трехмерной сцены, но в порядке убывания координаты *z.* При этом происходит наложение элементов сцены, причем более близкий к зрителю элемент прорисовывается поверх более удаленного. Естественно, что видимым оказывается последний из элементов, т. е. ближний. Z-сортировка не требует хранения (запоминания) координаты *z* (z-буфер не используется), однако нерационально загружается CPU: прорисовывается весь объект, независимо от видимости его элементов. Кроме того, при прорисовке близко расположенных или пересекающихся объектов данный метод часто дает ошибочный результат.

Русские

Авто-мип-мэппинга (Auto MIP-mapping) - создание 3D-акселератором MIP-каскада, когда прикладная программа вместо полного MIP-каскада содержит только один вариант текстуры. При этом имеющейся текстуре присваивается максимальный уровень детализации, а затем выполняется ее кратное уменьшение. Масштабирование выполняется один раз, в начале работы, и в дальнейшем используется обычная техника мип-мэппинга.

Активная страница (используются также синонимы *текущая* и *визуальная)* - видеостраница, содержимое которой в данный момент выводится на экран.

Алиасинг (aliasing) - рваные края объектов (лестничный эффект); дефект особенно заметен на наклонных линиях.

Альфа-буфер - часть кадрового буфера, в котором хранятся атрибуты α каждого пиксела, используемые для реализации смешения цветов (альфа-смешения).

Альфа-смешение (alpha-blending) - метод коррекции цвета пиксела, использующий смешение цветов пикселов с учетом коэффициента α. Технология, позволяющая управлять насыщенностью цвета и прозрачностью каждого пиксела при наложении объектов.

Антиалиасинг (antialiasing) - коррекция дефектов картинки, вызванных угловатостью линий на границах (краях) растровых объектов.

Аппаратные средства ускорения трехмерной графики - средства PC, к которым относятся: собственно 3D-акселераторы; специализированные высокоскоростные шины ввода/вывода, предназначенные для обеспечения работы 3D-акселератора (шины AGP и AGP Pro).

Банк памяти - совокупность микросхем памяти, суммарная разрядность которых равна разрядности шины данных. Например, для организации памяти из 8-разрядных микросхем при 32-разрядной шине данных каждый банк должен содержать 4 микросхемы памяти одинаковой емкости. Обычно используются два банка памяти, чтобы реализовать режим чередования адресов и циклов регенерации.
ввод в PC отдельных кадров или целых видеофрагментов; формирование стандартного телевизионного сигнала (функция TV-out); воспроизведение Video-CD и DVD в полноэкранном режиме и др.

Вертекс, от англ. *vertex* - совокупность трехмерных координат вершин элементарных треугольников, определяющая направление вектора нормали к этой грани.

Взвешенное усреднение цветов - метод преобразования цветного изображения в черно-белое с градацией серого. Он заключается в том, что по 6-разрядным значениям трех основных цветов *(R, G* и *B,,* образующим цветовой оттенок пиксела, вычисляется *одно* 6-разрядное значение яркости *(Y)* пиксела черно-белого изображения. Это значение заносится во все три 6-разрядных элемента выходного регистра цвета ЦАП. В этом случае на трех входах каналов ЦАП оказывается одно и то же число К. Очевидно, что сформированные на выходах ЦАП аналоговые сигналы *R, G* и *B* также будут одинаковы. В результате три зерна люминофора в пределах триады будут иметь одинаковую яркость свечения и ЭЛТ цветного монитора будет формировать оттенки только серого цвета. Человеческий глаз по-разному воспринимает объекты, окрашенные в различные цвета. Эта зависимость описывается *кривой относительной спектральной чувствительности глаза,* имеющей максимум в области зеленого цвета (длина волны 550 нм). В соответствии с данной кривой, при вычислении средневзвешенного значения яркости используются следующие весовые коэффициенты: $Y = 0{,}3R + 0{,}59G + 0{,}11B.$

Видеопамять - это специализированное ОЗУ, размещенное на плате видеоадаптера и предназначенное для хранения цифрового образа формируемого изображения. Синонимами данного понятия являются термины *видеобуфер* и *Video RAM.* Отметим, что термин Video RAM (или VRAM) используется также для обозначения конкретного типа высокоскоростной двухпортовой видеопамяти, используемой в дорогих профессиональных видеоадаптерах. Видеопамять видеоадаптеров, в которых для построения изображения используется не только своя собственная, но и системная память PC (например, видеоадаптеры с интерфейсом AGP), часто называют *локальной,* подчеркивая тем самым место ее физического размещения.

Видеорежимом - совокупность параметров, характеризующих режим работы видеоадаптера (разрешение, палитра, частоты строчной и кадровой разверток, способ адресации участков экрана и др.).

Видеосистема персонального компьютера - совокупность устройств, выполняющих следующие функции: формирование и обработку цифрового изображения; преобразование цифрового изображения в видеосигналы, поступающие на устройство отображения; формирование изображения.

Видеостраница (video page) - область видеопамяти в которой хранятся коды символов (и их атрибутов), отображаемых на экране монитора Видеостраница является аналогом кадрового буфера в графическом режиме, но имеет значительно меньший объем. В наиболее распространенном текстовом режиме 80×25 символов размер видеостраницы составляет 4000 байт, в режиме 40×25 — 2000 байт.

Гамма-коррекция - коррекция.нелинейности модуляционной характеристики ЭЛТ.

Геометрическая стадия 3D-конвейера - первые шесть этапов конвейера. Смотри "*3D-конвейер*"

Глубина пиксела - число разрядов видеопамяти, используемых для кодирования цвета или яркости пиксела.

Графические примитивы - линии, прямоугольники, окружностей и т.п.

Графический контроллер- элемент видеоадаптера VGA, который предназначен для управления обменом данными между центральным процессором и видеопамятью, а также для выполнения элементарных преобразований этих данных. Графический контроллер помогает центральному процессору выполнять следующие элементарные операции: запись пиксела (группы пикселов) по заданному адресу; считывание значения пиксела (группы пикселов) по заданному адресу; модификация цвета пиксела (группы пикселов) путем выполнения таких элементарных логических операций, как «И», «ИЛИ», «Исключающее ИЛИ», циклический сдвиг бит в пределах байта и др.; считывание из кадрового буфера кода пиксела (считывание со

сравнением цвета; фактически реализуется поиск пиксела с заданным цветом) и др.

Графический процессор - современный вариант *графического сопроцессора.*

Графический сопроцессор - это интеллектуальное устройство, основу которого составляет арифметико-логическое устройство (АЛУ). АЛУ фактически представляет собой специализированный микропроцессор, работающий по программам, содержащимся в собственном ОЗУ микроинструкций. Основное отличие графического сопроцессора от графического акселератора заключается в том, что сопроцессор можно запрограммировать на выполнение различных задач, тогда как ускоритель работает по жесткому, неизменному алгоритму. Помимо этого графический сопроцессор, в отличие от графического акселератора, является активным устройством, сфера деятельности которого не ограничивается только видеопамятью. В процессе выполнения своих функций он может наравне с центральным процессором обращаться к системной оперативной памяти и брать на себя управление шиной ввода/вывода.

Графический ускоритель (акселератор) - специализированное устройство видеоадаптера SVGA, предназначенное для выполнения только графических функций. Повышение быстродействия видеосистемы и PC в целом при использовании графического ускорителя достигается за счет: аппаратной реализации заданного набора графических функций, выполнение которых осуществляется всего лишь за несколько тактов работы акселератора; использования акселератором системы команд высокого уровня, что разгружает шину ввода/вывода — поток данных значительно уменьшается; освобождения центрального процессора от необходимости выполнения множества элементарных операций с содержимым кадрового буфера.

Двойная буферизация - используется для ускорения процесса создания изображения. Механизм ускорения заключающийся в том, что в видеопамяти выделяется область для хранения одновременно двух смежных кадров (по сути, два кадровых буфера): построение следующего кадра начинается еще до того, как закончится отображение (передача в RAMDAC) текущего. В результате удается избежать ненужного «простоя» акселератора и обеспечить более высокую частоту смены кадров.

Двумерная адресация. - Применяется при разбиение кадрового буфера на цветовые плоскости для ускорения его заполнения в 16-цветных графических режимах (в частности, в режиме 12h видеоадаптера VGA). При этом адресация пиксела фактически становится двумерной. Положение пиксела в кадровом буфере характеризуется уже двумя координатами: адресом пиксела в пределах цветовой плоскости (смещением ячейки относительно начала плоскости и номером бита в ячейке); номером цветовой плоскости (когда речь идет о методе цветовых плоскостей, термины «цветовая плоскость», «битовая плоскость», «банк» и «видеостраница» являются синонимами). Преобразование двумерного адреса в одномерный и обратно осуществляет графический контроллер.

Двумерное маскирование - маскирование бит в пределах байта и цветовой плоскости в пределах группы из четырех плоскостей. Маскирование пикселов в пределах группы из 8 пикселов осуществляется путем ввода соответствующего значения в регистр битовой маски и используется при записи, а маскирование цветовых плоскостей — путем ввода соответствующего значения в четыре младших разрядах регистра безразличного цвета. Двумерное маскирование реализуется графическим контроллером в процессе выполнения операций чтения/записи кадрового буфера.

Двухпортовая видеопамять - позволяет осуществлять одновременное выполнение операций чтения и записи данных, т. е. к ней одновременно могут обращаться два устройства, например графический процессор и RAMDAC. Обеспечивает значительное ускорение работы видеоадаптера, особенно в режимах с высокими разрешением и глубиной цвета. В настоящее время используются две разновидности двухпортовой памяти — VRAM и WRAM.

Декодер - специальный блок, используемый для выделения из *полного цветового телевизионного сигнала* сигналов цветности.

Дизеринг (dithering) - интерполяция недостающих цветов. Один из этапов 3D-конвейера. Применяется в том случае, когда в текущем видеорежиме 3D-акселератора для кодирования цвета пиксела используется менее 24 бит (например, в режиме High Color). Это приводит к незначительному ухудшению разрешения (резкости), но позволяет эффективно имитировать отсутствующие в палитре цвета.

Дополнительные функции VBE - функции, которые позволяют наиболее полно использовать возможности видеосистемы современного мультимедийного PC. При помощи функции 4F10h операционная система реализует энергосберегающие функции монитора согласно стандарту DPMS, также разработанному VESA. Напомним, что управление состоянием монитора, согласно данному стандарту, осуществляется путем отключения сигналов синхронизации, что и позволяет сделать функция 4F10h. С помощью функции 4F13h можно управлять встроенной акустической системой мультимедийного монитора. При помощи функции 4F15h реализуется поддержка мониторов Plug&Play, использующих для обмена данными специальный канал DDC.

Закон освещенности - освещенность точки поверхности прямо пропорциональна силе света и косинусу угла α между направлением падения света и нормалью к поверхности и обратно пропорциональна квадрату расстояния до источника света.

Затенение по методу Гуро - метод Гуро, изобретенный еще в 1971 г. и названный так по имени автора, был предназначен для устранения «угловатости», характерной для плоского затенения, и создания иллюзии кривизны плоского объекта. Затенение по методу Гуро выполняется на основе результатов, полученных при плоском затенении. Расчет освещенности по методу Гуро выполняется в два этапа. Сначала вычисляются

значения освещенности на ребрах грани (как среднее арифметическое освещенности для смежных граней). На втором этапе выполняется линейная интерполяция освещенности вдоль параллельных линий от одного ребра к другому внутри грани. Интерполяции может подвергаться не только освещенность, но и цвет. Метод Гуро применим только для небольших граней, расположенных на значительном расстоянии от источника света. В игровых 3D-акселераторах затенение Гуро является наиболее распространенным приемом, позволяющим быстро и легко устранить дефекты, полученные в результате тесселяции поверхности объекта.

Затенение по методу Фонга - назван в честь автора: Ву Тонг Фонга (Wu Tong Phong). Данный метод, как и плоский и Гуро, предполагает постоянство расстояния от источника света до каждой из вершин грани, но учитывает изменение угла падения света для различных точек грани (в идеале — для всех точек, как минимум — для вершин грани). Это позволяет более корректно рассчитать освещенность грани, особенно в том случае, когда источник света расположен на небольшом расстоянии от нее. Другой важной особенностью затенения по методу Фонга является то, что для каждой точки грани используется своя «нормаль», направление которой может отличаться от направления истинной нормали к грани. Метод Фонга требует значительных вычислительных затрат и редко используется в реальном масштабе времени.

Затуманивание (fogging) - метод коррекции цвета пиксела.

Интерполяция недостающих цветов (Color Dithering (дизеринг)) - метод интерполяция цветов, заключающийся в получении недостающих цветов за счет смешения имеющихся. Применяется тогда, когда текущий режим работы видеоадаптера не позволяет отобразить требуемое количество оттенков цветов (имеет недостаточный объем видеопамяти). Этот метод широко распространен и используется не только в трехмерной графике, но и во многих программах обработки растровых изображений.

Интерфейс (Interface) - совокупность конструктивных особенностей шины и протокола обмена.

Информационные функции VBE - функции с номерами 4F00h и 4F01h, которых не было в стандартном Video BIOS. Они позволяют получить подробную информацию обо всех видеорежимах, поддерживаемых данным видеоадаптером, что облегчает прикладной программе их корректное использование.

Кадровый буфер или ***фрейм-буфер*** (от *англ. frame buffer* - кадровый буфер) - участок видеопамяти, отведенный для хранения цифрового образа текущего изображения (кадра).

Каркасная модель объекта, или *wireframe.* Получается на первой стадии 3D-конвейера при построение геометрической модели поверхности объекта путем задания трехмерных координат опорных (ключевых) точек, а также уравнений соединяющих их линий.

Карта затенения - представляет собой изображение плоской поверхности, освещаемой точечным источником света, расположенным над ее центром. Освещенность каждой точки этой карты рассчитывается на основе закона освещенности. Заметим, что точки карты, лежащие на концентрических окружностях, имеют одинаковую освещенность, поскольку для них угол между нормалью к поверхности и направлением на источник света одинаков. Могут использоваться при выполнении затенения Фонга Карта затенения используется в качестве шаблона, задающего распределение освещенности в пределах грани. Достаточно наложить соответствующим образом контур грани на карту - и освещенность каждой точки грани станет известной. Этот алгоритм предложил Майкл Абраш.

Кинескоп (от греч. *Kinesis* - движение, *Skopeo* - смотрю, наблюдаю) - применяемые в телевизорах ЭЛТ.

Код атрибутов символа - байт, несущий информацию о способе представления символа на экране монитора: цвет и яркость символа и фона, подчеркивание или мерцание символа. Обычно информация, содержащаяся в первых (младших) четырех разрядах байта атрибутов, описывает собственно символ, т. е. передний план символьной матрицы *(foreground),* а в четырех старших разрядах — задний план, т. е. параметры фона.

Код символа - порядковый номер символа в кодовой таблице.

Кодовая страница - вторая половина символьного набора (с номерами 128—255) отводимая под буквы национального алфавита и являющаяся уникальной для каждой страны. Каждой кодовой странице фирма Microsoft присвоила уникальный номер. Например, кодовая страница России имеет номер 866 для DOS (ASCII) и 1251 для Windows (ANSI).

Компонентный видеосигнал - предусматривает раздельную подачу сигнала яркости Y и сигналов цветности C и обозначается *Y/C* или S-Video. Раздельная подача сигналов Y и C позволяет повысить качество изображения за счет устранения типичных для композитного видеосигнала перекрестных искажений. Такой сигнал используется в полупрофессиональной видеозаписывающей аппаратуре формата S-VHS.

Контроллер атрибутов видеоадаптера VGA - предназначен для управления цветом выводимого на экран монитора изображения. В текстовом режиме работы видеоадаптера он, на основании содержимого байта атрибутов выводимого символа, задает цвет пикселов в пределах символьной матрицы. Кроме того, в этом режиме контроллер атрибутов позволяет создавать такие эффекты, как мигание, инверсия цвета или повышенная яркость символа. В графическом 16-цветном режиме контроллер атрибутов преобразует условный 4-разрядный номер цвета пиксела, хранящийся в видеопамяти, в 8-разрядный номер регистра RAMDAC, содержащего 18-разрядный код отображаемого цвета. Другими словами, в графическом режиме контроллер атрибутов управляет *16-цветной палитрой* — специальной таблицей перекодировки, содержащей 16 8-битных чисел. Эта таблица задает 16 одновременно отображаемых оттенков цветов из теоретически доступной 256-цветной палитры. Отметим, что графический режим с 16-цветной палитрой унаследован видеоадаптером VGA от его предшественника — видеоадаптера EGA, поэтому механизм

кодирования цвета пиксела в этом режиме у обоих видеоадаптеров имеет много общего. Помимо управления 16-цветной палитрой, контроллер атрибутов позволяет управлять цветом окаймления экрана (по умолчанию цвет окаймления всегда черный).

Контроллер ЭЛТ - элемент видеоадаптера, который формирует сигналы горизонтальной и вертикальной синхронизации, сигналы инкремента (увеличения) счетчика адреса ячеек видеопамяти, в которых хранится цифровой образ изображения, а также стробирующие сигналы чтения/записи видеопамяти. Эти сигналы формируются таким образом, что движение луча по экрану ЭЛТ осуществляется синхронно с процессом сканирования ячеек видеопамяти, причем цвет пиксела на экране соответствует коду, содержащемуся в соответствующей ячейке кадрового буфера. Контроллер ЭЛТ решает также ряд дополнительных задач: выбор формы курсора в текстовом режиме; выбор выводимой на экран активной видеостраницы; установка параметров аппаратного подчеркивания символов.

Максимальное (физическое) разрешение - определяется величиной зерен люминофора и их количеством на экране ЭЛТ.

Методы затенения - методы интерполяции значений освещенности элементарных треугольников поверхности, позволяющие плавно изменить освещенность каждой грани и скрыть резкие переходы между ними. Наиболее распространенными из этих методов являются *метод Гуро (Gouraud Shading)* и *метод Фонга (Phong Shading).*

Минипорт **-** специализированные OpenGL-совместимые драйверы, специально созданные для работы с какой-либо конкретной программой (игрой).

Мип-мэппинг (MIP-mapping) или ***MIP-текстурирование*** - использование текстур с различным разрешением: технология прорисовки ближних и дальних объектов. Суть ее состоит в том, что в локальной памяти акселератора хранятся несколько вариантов одной и той же текстуры, но с разным разрешением, или уровнем детализации (LOD — *Level Of Detalization*}, например 64×64,128×128 и т. д. Размер каждой текстуры следующего уровня в 4 раза больше размера предыдущей. Совокупность всех вариантов одной и той же текстуры называют *MIP-каскадом*. При рендеринге ближних поверхностей используются более крупные текстуры, а при прорисовке дальних — более мелкие. Это значительно ослабляет эффект пикселизации. Название этой технологии образовано аббревиатурой MIP — *Multum In Parvo* — «много в одном», что отражает основную идею описанных выше действий.

Моделирование эффектов прозрачности и полупрозрачности - этап 3D-конвейера, на котором на основе информации о взаимной прозрачности объектов и среды выполняется коррекция цвета пикселов — так называемое *альфа-смешение (alpha-blending)* и *затуманивание (fogging).*

Мультимедиа (multimedia) - совокупность множества сфер (сред) информационного взаимодействия человека с компьютером. Несколько витиеватое понятие «среда информационного взаимодействия» — не что иное, как материальная среда, воздействующая на органы чувств человека.

Мультимедийные функции - функции видеоадаптера, к которым относятся следующие: прием телевизионных программ и просмотр их на экране монитора; прием радиопрограмм;

Мультитекстурирование **-** текстурирование, выполняющееся с помощью двух или более блоков обработки и наложения текстур.

Обработка (настройка) координат проекций вершин элементарных треугольников (Triangle Setup) - этап 3D-конвейера, на котором выполняется преобразование формы представления координат вершин граней: числа с плавающей точкой (вещественные) преобразуются в целые числа. Кроме того, производится сортировка вершин, отбрасывание задних граней *(culling)* и так называемая *субпиксельная коррекция*

ОЗУ - оперативное запоминающее устройство (см. RAM).

Пакетный режим - режим, в котором процессор считывает данные из памяти не побайтно, а в виде пакетов *(burst),* или слов, состоящих, в зависимости от разрядности шины данных, из 32 или 64 бит. Время считывания каждого слова измеряется в тактах процессора и записывается в виде *временной диаграммы.* Например, временная диаграмма 7-3-3-3 означает, что на считывание первого слова необходимо 7 тактов процессора, а на считывание каждого последующего — только 3.

Пиксел (Pixel— Picture Element) - минимальный элемент изображения на экране видеомонитора.

Пикселизация **и** ***алиасинг*** (зазубренности наклонных линий) - характерные дефекты масштабирования растровых изображений.

Плоское (простое) затенение **-** это наиболее простой и очевидный метод, согласно которому освещенность всех точек грани считается одинаковой и определяется только ориентацией грани относительно падающего света, т. е. косинусом угла α. В этом случае яркость граней, расположенных на изгибах поверхности, будет резко отличаться от яркости соседних граней, в результате чего поверхность объекта будет иметь ярко выраженную «граненую» структуру. Достоинством плоского затенения является простота и, как следствие, высокая скорость его реализации.

Полный антиалиасинг, или ***субпиксельный антиалиасинг.*** - метод, используемый для полного устранения всех дефектов алиасинга. Основная идея данного метода состоит в следующем. Каждому пикселу итогового изображения ставится в соответствие совокупность *виртуальных субпикселов,* при этом так называемое *виртуальное разрешение* изображения оказывается выше. Антиалиасингу подвергаются именно субпикселы. После выполнения коррекции, когда цвета всех субпикселов определены, субпикселы объединяются в пикселы путем интерполяции, в результате чего восстанавливается исходное разрешение.

Полный цветовой телевизионный сигнал (*композитный*, совмещенный) - телевизионный сигнал, который помимо видеосигналов цветности содержит синхросигналы строчной и кадровой разверток.

Полосы MIP-текстурирования (Banding of MIP-mapping) - хорошо различимые границы между текстурами с разным разрешением.

Пост-обработка (Post-processing) - этап 3D-конвейера, который используется в том случае, когда требуется реализовать какие-либо двумерные эффекты над подготовленным кадром как единым целым.

Преобразователь развертки - блок TV-тюнера в котором телевизионные поля (полукадры) чересстрочной развертки преобразуются в полные кадры, необходимые для реализации построчной развертки на экране монитора. Кроме того, частоты кадров и строк преобразованного сигнала выбираются с учетом текущего режима работы видеосистемы PC. Следует отметить, что корректность выполнения этого преобразования очень сильно влияет на качество получаемого изображения. В дорогостоящих моделях TV-тюнеров на этом этапе используется так называемый *фильтр мерцаний (Flicker Filler),* позволяющий устранить характерные для телевизионного изображения дефекты, вызванные чересстрочной разверткой.

Проецирование трехмерного объекта на плоскость экрана - этап 3D-конвейера, на котором происходит первое, предварительное преобразование трехмерного объекта в совокупность двумерных координат вершин его граней, при этом запоминается информация о расстоянии каждой из вершин грани до плоскости проецирования (координата z). Это позволяет на следующих этапах корректно определить, какие части объекта окажутся видимыми, а какие — нет. Совокупность данных о глубине каждой из вершин, или значения ее третьей координаты *z,* называют *z-буфером.*

Пространственное смешение основных цветов - эффект, согласно которому любой оттенок цвета можно получить из комбинации трех основных (базовых) цветов: красного *(Red),* зеленого *(Green)* или синего *(Blue).* В цветных телевизионных трубках, поскольку размер зерна люминофора весьма мал и соответствует разрешающей способности глаза, близкорасположенные зерна триады RGB воспринимаются как одна точка, цвет которой определяется *этим эффектом (законом).*

Протокол - набор правил, или алгоритм обмена.

ПЦТС (полный цветной телевизионный сигнал) - композитный сигнал, формируемый специальным кодирующим устройством видеоадаптера CGA из четырех двоичных сигналов *I, R, G, B* и сигналов синхронизации.

Размер адресного пространства - общее количество адресов, которое может сформировать CPU. Размер адресного пространства определяется разрядностью шины адреса, например, 20-разрядная шина обеспечивает адресное пространство 2^{20} = 1048576 = 1 М различных адресов.

Регистр-защелка - специальный буферный регистр графического контроллера использующийся для управления чтением/записью информации в каждую битовую плоскость.

Режим реальной адресации - режим работы процессоров i80286 и старше, в котором используется механизм получения 20-разрядного физического адреса из двух 16-разрядных слов, хранящихся в одном из базовых 16 -разрядных регистров процессора и 16-разрядном регистре смещения (такой механизм использовался в процессорах i8086 и i8088).

Рендеринг (rendering - рисование, визуализация) - стадия 3D-конвейера отвечающая за прорисовку объекта. Смотри "*3D-конвейер*".

Секвенсер (указатель последовательности) - элемент видеоадаптера VGA предназначнный для генерации сигналов, необходимых для сканирования видеопамяти. Другими словами, секвенсер обеспечивает последовательную адресацию и считывание содержимого ячеек видеопамяти и передачу их содержимого в контроллер атрибутов и далее — в RAMDAC. Работа секвенсора синхронизируется стробирующими сигналами, формируемыми контроллером ЭЛТ. Кроме того, секвенсер позволяет выполнять переключения между 8 различными таблицами национальных шрифтов, при этом одновременно могут использоваться два шрифта.

Синтез трехмерного изображения - процесс, включающий в себя следующие этапы: конструирование (расчет) трехмерного объекта на основе исходных данных, например результатов 3D-сканирования или математического описания объекта; расчет движения объекта и трансформации его формы; моделирование поверхности объекта с учетом различных факторов (освещения, отражений, рельефа и раскраски поверхности и т. п.); проецирование рассчитанного объекта на плоскость экрана с учетом всех возможных визуальных эффектов, обусловленных его строением, особенностями освещения, степенью прозрачности среды и т. п.

Синхронизатор - элемент видеоадаптера VGA (и EGA), осуществляющий синхронизацию видеоадаптера и CPU Он управляет доступом CPU к кадровому буферу и разрешает его обновление только во время действия сигналов гашения. В результате устраняется основная причина конфликтов при работе с видеопамятью — асинхронность обращений к ней CPU и внутренних схем видеоадаптера.

Системы виртуальной реальности - системы, предназначенные для решения основной задачи мультимедиа: создание у человека иллюзии реальности искусственного, виртуального мира. Очевидно, что достичь этой цели можно лишь в том случае, если формировать виртуальные образы с учетом особенностей органов чувств человека, т. е. визуальные, звуковые и тактильные эффекты должны восприниматься человеком как реальные явления.

Смешение RGB- и ТВ-сигналов - функция TV-тюнера по смешению сигнала, соответствующего принятому телевизионному изображению, с выходным сигналом видеоадаптера. Это смешение может выполняться либо в аналоговом, либо в цифровом виде, с предварительным аналого-цифровым преобразованием выходного сигнала радиоприемника.

Состав видеосистемы мультимедийного компьютера - в ее состав входят ускорители двумерной и трехмерной графики, устройства цифровой обработки и воспроизведения видеоданных, разнообразные по конструкции и возможностям устройства формирования трехмерных изображений, мультимедийные проекторы, устройства приема телевизионных сигналов (TV-тюнеры) и т. п.

Спрайт (Sprite — эльф, гном) - небольшие перемещаемые прямоугольные фрагменты изображения.

Субпиксельная коррекция - метод коррекции цвета по граням изображения с использованием усреднения (интерполяции) цвета суббпикселов.

Субпикселы - совокупность пикселов, получаемая методом *суперсэмплинга,* или *мультисэмплинга* (от англ. *sample* — отсчет). В первом случае субпикселы получаются на стадии рендеринга путем растрирования изображения с заведомо более высоким, виртуальным разрешением, чем требуется. По окончании стадии рендеринга выполняется интерполяция, в результате которой группы субпикселов объединяются в пикселы изображения с итоговым разрешением. Во втором случае субпикселы формируются по окончании рендеринга путем разбиения исходного пиксела на части. Характер разбиения пиксела определяется маской, задающей пространственное положение субпикселов.

Таблица векторов прерываний. - Используется для простой и эффективной локализации процедуры обслуживания прерываний. Формируется, как правило, в первых ячейках оперативной памяти (адреса: 0 - 3FFh). В каждой строке этой таблицы содержится адрес программы обработки прерывания, а порядковый номер строки таблицы соответствует номеру прерывания.

Тактильные ощущения**,** - способность человека осязать объекты;

Тексел - минимальный элемент текстуры. Иными словами, тексел — это пиксел текстуры.

Текстура (texture) — это элемент обшивки объекта, т. е. изображение участка его поверхности. Она хранится в виде квадратной растровой картинки, состоящей из *текселов.*

Текстурирование или *закраска элементарных треугольников* — этап 3D-конвейера, на котором выполняется наложения текстур *(texture mapping)* на поверхность объекта с учетом данных, полученных в результате выполнения предыдущих этапов конвейера. После наложения текстур каркасная модель как бы «обрастает кожей» и становится похожа на реальный объект. Наложение текстур — это первый этап конвейера, выполняемый с растровой графикой. На данном этапе каждый треугольник заменяется частью текстуры. При этом значение каждого пиксела грани вычисляется по значению соответствующего тексела текстуры. Соответствие между пикселами и текселами обеспечивается путем проецирования текстуры на поверхность грани. В зависимости от формы объекта и его отдельных частей (для объектов сложной формы) используются различные системы проекционных координат: прямоугольная, сферическая, цилиндрическая и т. д. Текстурирование — довольно трудоемкий процесс, поскольку обработке подвергается растровая графика. При модификации растрового изображения возникают многочисленные дефекты, поэтому на данном этапе применяются различные приемы коррекции изображения: мипмэппинг (*MIP-mapping*— использование текстур с различным разрешением), коррекция перспективы, фильтрация и др.

Теневая память (Shadow Memory) - специально отведенная область оперативной памяти PC, в которую копируется содержимое ROM (ПЗУ), например, ROM Video BIOS.

Тесселяция (tesselation) - разбиение поверхности полученного объекта на элементарные плоские элементы: прямоугольники или треугольники. Этот, 2-ой этап 3D-конвейера, называется также *триангуляцией.* В результате тесселяции поверхность объекта, обычно имеющая искривленную форму, становится «граненой», т. е. представляет собой совокупность плоских многоугольников (треугольников или четырехугольников). Чем больше таких граней и чем меньше их размер, тем точнее воспроизводится поверхность объекта. Чаще всего поверхность объекта разбивается на треугольники, поскольку тремя точками (вершинами треугольника) однозначно задается плоскость в пространстве. На этапе тесселяции математическая модель объекта представляет собой совокупность трехмерных координат всех вершин *(вертексов,* от англ. *vertex)* элементарных треугольников, записанных в определенном порядке (иными словами, порядок следования координат вершин грани определяет направление вектора нормали к этой грани).

Трансформация (transformation) - это этап 3D-конвейера, на котором моделируется движение объекта: его перемещение, вращение и изменение размеров (формы). Этап сводится к стандартному преобразованию трехмерных координат вершин граней (вертексов) и требует интенсивного использования операций матричной (линейной) алгебры и тригонометрических функций.

Трехмерная графика* или *3D - графика (3-Dimentional — трехмерный) - совокупность приложений и задач, в рамках которых реализуется схема построения трехмерного изображения на экране монитора PC.

Удаление скрытых поверхностей (Hidden Surface Removal* - HSR*) — этап 3D-конвейера, на котором выполняется исключение из проецирования тех элементов поверхности объекта, которые оказываются невидимыми из точки наблюдения. Строго говоря, процесс удаления скрытых поверхностей выполняется последовательно, в несколько этапов: на предыдущем (Triangle Setup), на котором обрабатываются векторные изображения, и на последующих, на которых на основе алгоритмов z-буферизации или z-

сортировки обрабатываются уже растровые изображения

***Управляющие функции* VBE** - функции, предназначены главным образом для решения задач, характерных именно для режимов SVGA, а именно: для реализации универсального способа переключения банков видеопамяти (функция 4F05h); для управления работой видеоадаптера в режимах High Color и True Color (функции 4F08h, 4F09h); для реализации *линейной адресации* кадрового буфера, доступной в защищенном режиме работы CPU.

Ускоритель трехмерной графики или ***3D-акселератор*** - специализированное устройство видеосистемы PC самостоятельно выполняющее основную часть операций по расчету трехмерной сцены.

Фильтрация анизотропная - используется при прорисовке сильно наклоненных к наблюдателю граней, когда текселы перестают быть квадратными. В соответствии с этим методом, тексел описывается эллипсом и интерполяция выполняется по большему количеству текселов (до 32). Анизотропная фильтрация значительно сложнее трилинейной и используется в основном в высококачественных игровых и профессиональных 3D-акселераторах.

Фильтрация билинейная (bi-linear filtering, BLF) - фильтрация, при которой каждый тексел вычисляется как взвешенная сумма значений четырех соседних (слева и справа, сверху и снизу) текселов; это наиболее распространенный метод, однако в результате его применения остаются полосы MIP-текстурирования и изображение немного размывается.

Фильтрация линейная (linear filtering, LF) - фильтрация, при которой для интерполяции тексела используется значение тексела в текстуре ближайшего уровня (LOD).

Фильтрация трилинейная (tri-linear filtering, TLF) - при такой фильтрации каждый тексел текстуры вычисляется как усредненный результат билинейной фильтрации по двум соседним уровням детализации (используются две текстуры); это наиболее точный из используемых методов, при котором полосы MIP-текстурирования не образуются.

Цветовая плоскость - это участок видеопамяти, представляющий собой матрицу бит, каждый из которых соответствует отдельному пикселу на экране ЭЛТ. Адресуемым элементом такой плоскости является не байт, а бит. Количество цветовых плоскостей равно количеству бит, отводимых для кодирования цвета пиксела (2 или 4), а их размеры соответствуют разрешению экрана.

Цветоделительная маска - элемент цветной ЭЛТ (маска), обеспечивающий попадание каждого из трех электронных лучей на зерна люминофора своего цвета

ЭЛТ - это электронно-вакуумный прибор, предназначенный для преобразования электрического сигнала, формируемого видеоадаптером, в изображение (см. CRT).

Алфавитный указатель

α-буфер 236
«Граненая» структура 211
«Кроссовый» контроллер 261
«Узкий» z-буфер 214
«Широкий» z-буфер 214
0,18-микрометровая технология 237
0.15мкм технология 262
16777216 оттенков цветов 105
16-цветная палитра 91
16-цветные режимы 99
2.132 Гбайт/с 239
256 оттенков серого 187
256-цветные режимы 152
256-цветный режим 101
262 144 оттенков цветов 94, 101
2D графика 49
2D-акселератор 49
32 768 и 65 536 оттенков цветов 105
386DX 12
386SX 12
3D RAM 115
3D4h и 3D5h, 77
3Dfx 245
3Dlabs 259
3D-акселератор 11, 49, 203, 235
3D-акселератор 235
3D-графика 49, 242
3D-конвейер 11, 204
3D-моделироване 203
3D-приложения 50
3D-синтез 209
3D-системы 209
3D-сцена 220, 238
3B4h и 3B5h 77
440BX 242
440LX 242

Английский

A

A0000h 154
Access time 112
AD (Address-Data) 240
Adaptive Antialiasing 262
AGP «2x» и «4x» 239
AGP 8x 239
AGP Pro 203, 242
AGP Revision 2.0 239
AGP 13, 117, 190, 215, 238
AGP-текстурирование 242
Aliasing 216
Alpha pixel blending 221
Alpha-blending 206
ANSI 22
APA (All Point Addressable) 21
API — Metal3D 258
API (Application Program Interface) 229
API DirectX 6.0 257
API DirectX 229
API Glide 229
API OpenGL 229
API RenderGL 229
ARB - Architecture Review Board 230
ASCII 22
ASCII-коды 137, 149, 151
Aspect ratio 45
ATI Radeon 256 238
ATI Radeon 256 256
ATI Rage 128 Pro 255
ATI Rage 128 255
ATI Rage Fury MAXX 255
ATI Technologies 237
ATI 254
ATIM64.drv 192
ATIR3.drv 192
Audio 198
Auto MIP-mapping 216
AWARD BIOS 191

B

Back buffer 236
Background 22
Banding of MIP-mapping 216
BCD 149
BEDO DRAM 113
Bi-linear filtering, (BLF) 216
BIOS CGA 28
BIOS extension 66
BIOS MDA 27
BIOS 13
Bit Blt (Bit Block transferring) 107
Bleeding 225
BMP 186
Bump-mapping 219
Burst 112
Bus Mustering 111, 117

C

CAS 112
CD-I 49
CDRAM 115
CD-ROM 10, 49
CGA snow 30, 96
CGA 19, 27
Character boxes 21
Charisma Engine 237
Chipset видеоадаптера 97
Chipset для 3D-акселераторов 245
Chipset для 3D-видеоплат 260
Chipset фирмы 3Dfx Interactive 245
Chipset 48
Chroma Keying 223
CIRRUSMM.drv 193
ClearScreen 133
Clip volume 213
CLUT (Color Look-Up Table) 223
CMOS Setup 66, 242
Color Dithering 226
COM - Common Object Model 231
COM-порт 108

Conventional Memory 57
CPU 12
CRTC 56
Culling 206, 214

D

DAC 44
DDC (Display Data Channel) 45, 120, 127
DDK - 3D Graphics Device Driver Kit 231
DDR SDRAM 115, 237
DiME (Direct Memory Execution) 240, 239,
DiME (DME) 111, 240
DIN 198
Direct Color 126
Direct Model 231
Direct RDRAM 114
Direct3D 229
DirectDraw 152, 154, 229
DirectPlay 229
DirectSound 229
DirectX 6.0 225
DirectX 200, 229
Display Enable 96
DMA (Direct Memory Access) 240
Dot Clock 21, 76, 82, 96
Dot matrix 22
Dot rate 21
DRAM 111
Drawing Commands 107
Dual Head 254
Dual-Color Block Write 114
DVD 197
DVD-диск 49
DVI-интерфейс 262
DXDiag.exe 232
D-образный разъем 44, 56

E

Edge anti-aliasing 225
EDO DRAM 113
EGA 19, 30
EMBM 220
Embossing Bump Mapping 219
Environment Map Bump Mapping 219
ES 151
ESDRAM 115
ET6000.drv 193
Exponential Fog 222

F

Fahrenheit Large Model Visualization 231
Fahrenheit Low Level API 231
Fahrenheit Scene Graph 231
Fahrenheit 231
FastBit 114
FBI (Frame Buffer Interface) 236
FBLD 141
FBSTP 141
Feature Connector 197, 200
Fibre Channel 114
FILD 141
Fill Rate 236, 245
FIST 141
FISTP 141
Flash ROM BIOS 191
Flat Shading 210
FLD 141
FLDLG2 142
Flicker Filler 199
Flip -flop технология обращения 39
Fogging 222
Foreground 22
FPM DRAM 112
Frame buffer 20, 64
Front buffer 236
FS 171
FSAA - Full-Screen Anti-Aliasing 225, 249
FST 141
FSTP 141
FYL2X 142

G

Game Engine 231
GeForce 2 GTS 251
GeForce 256 251
GeForce2 GTS 261
GeForce3 Ti 500 261
GeForce3 259, 261
GeForce4 Ti 261
GeForct4 Ti 50
Geometry Processor 235
GLINT. DRV 193
Gouraud Shading 206
Graphic OpenGL 231
GS 171
GTS (Gyga Texel Shading) 251
GUI — Graphic User Interface 106

H

Hardware character generator 22
Hardware Windowing 107
HiColorl5 (формат 1:5:5:5) 153
HiColorl6 (формат 5:6:5) 153
High Color 47, 95, 105. 152, 163
High-End 256
HIMEM.SYS 58
HSR (Hidden Surface Removal) 206, 213
H-Sync 82
HWT&L (HardWare Transform and Lighting 251
HyperZ II 262

I

I — Intensity 19, 29
i486DX2 117
i740 247
i80286 57
i8087 140
i8087 140
IBM PC AT 30
IBM PC XT 27
IBM PC. 12
IBM PS/2 43
ICD - Installable Client Driver 230
IGA2K.DRV 193
IHV - Independent Hardware Vendor 230
Include-файл 130, 132

Init Display First 191
Int 10h, функция 4Fh, подфункция 00h 120
Int 10h, функция 4Fh, подфункция 01h 120, 154
Int 10h, функция 4Fh, подфункция 02h 123, 154
Int 10h, функция 4Fh, подфункция 03h 123
Int 10h, функция 4Fh, подфункция 04h 123
Int 10h, функция 4Fh, подфункция 05h 124
Int 10h, функция 4Fh, подфункция 06h 124, 154
Int 10h, функция 4Fh, подфункция 07h 125, 154
Int 10h, функция 4Fh, подфункция 08h 126
Int 10h, функция 4Fh, подфункция 09h 126
Int 10h, функция 4Fh, подфункция 0Ah 126
Int 16h 137
INT l0h 26, 47, 66
Int32_to_String 142
INTEGRATED PERIPHERALS 191
Intel 440BX 239
Intel 80286 30
Intel 80486 21
Intel 8088 12
Interlaced Mode 16
Internet 49
Interrupt 66
IRGB 29, 101
IRQ (Interrupt Request Line) 66
IRQ2/9 96
ISA 12, 97, 116, 190

J

JPEG 224

K

Key Frame Interpolation 237

L

Linear filtering, (LF) 216
Linear Fog 222
list1_02.inc 133
list1_03.inc 138
list1_04.inc 139
list2_01.inc 62
list2_05.inc 140, 142
list4_02.inc 158
list4_03.inc 161
list4_09.inc 163
list4_14.inc 171
list4_15.inc 174
list4_16.inc 175
list4_17.inc 176
list4_18.inc 179
list4_19.inc 180
lnt10h, функция 10h, подфункция 12h 178
LOD - Level Of Detalization 215
LOD Morphing 223
LOD-bias 216
LoopBack-кабель 200

M

Matrox MGA G200 252
Matrox MGA G400 253
Matrox 252
MAXX 236
MCA 55, 116
MCD - Mini Client Driver 230
MCGA 43
MCLK 97
MDA 26
MDRAM 114
Memory Clock 96
MIP - *Multum In Parvo* *215*
MIP-mapping 206,215
MIP-каскад 215
MIP-текстурирование 215
Miscellaneous Output Register 96
Miscellaneous Output 34
Mitsubishi 115
MoSys Inc 114
Motorola 6845 28, 76/77
MPEG 108
MPEG-1/2 237
MS Photo Editor файл 188
MSD.EXE 47
MS-DOS 58

N

Narrow z-buffer 214
Nearest MIP-mapping 216
NEC 46, 105
Norfhwood 50
Norton Commander 25
NTSC 198
NV11 251
NV15 251
NVidia 248
NVMINI2.VXD 192

O

Object Transparency 222
OpenGL 229, 230

P

PAL 198
Palletised textures 224
Palletized 224
Panning 108
Pass-Through кабель 203
PC 10
PCI 13, 48, 117, 190, 242
PCX 126, 186
PDF (Packed Display Format) 64
PDP - Parallel Data Processing 248
Pentium 4 50
Pentium 12
Perspective Corrected Texture Mapping 218
Phong Shading 206
Pipelining 239
Pixel Clock 76
Pixel Engine 235
Pixel rate 21
Pixel Tapestry 237
Pixel 17
Plug&Play 117

POST (Power-On Self Test) 66
PowerStrip 84, 97
PPI (Precise Pixel Interpolation) 248
Procedural Texturing 219, 221

Q

QEMM 58

R

R, G, B - сигналы 16
Radeon 256 237
RADEON 8500 262
RAM 12
RAMDAC 43, 93, 105, 235
RAS 112
RCA 198
REALImage 115
Realtek 46
Refresh Rate 16
Rendering Engine 235
Rendering 207
RGBα 223
Riva 128 248
Riva 128TNT (Riva TNT) 249
Riva 128ZX 249
Riva TNT2 Ultra 250
Riva TNT2 Vanta 250
Riva TNT2 M64 251
Riva TNT2 249
Riva TNT2-A 250
RLE - (Run Length Encoding) 187
ROM 13
ROM BIOS 13
ROM Video BIOS 45
RrGgBb 99

S

S3 Savage 2000 258
S3 Savage 3D 257
S3 Savage 4 257
S3 Trio3D 256
S3MM.drv 192
S3TC - S3 Texture Compression 215, 224
S3V.drv 193
Savage 4 GT 257
Savage 4 Pro 257
Savage 4 Pro+ 257
SBA (Side-Band Addressing) 240
SBA Fast Writes 238, 239, 240
Scene Graph 231
Scientific Display Doctor 127
SCSI 13
SDRAM 113
SDRAM/SGRAM 237
SECAM 198
Setup Engine 235
SGI - Silicon Graphics, Inc 229
Shadow Memory 66
Silicon Graphics 48
Single-pass Quad-Texture engine 258
SLI (Scan Lines Interleave) 236, 246
Software character generator 22
Specular Highlights 223
SRAM 112
StretchBI 107
Strips&fans 235
Super VGA или SVGA 46, 105
SuperVGA, 800×600,16 colors 118
S-VHS 198
S-Video 198
Sysinfo 48

T

Table Fog 222
Tative API 231
TDMS-контроллер 262
Tesselation 204
Texas Instruments 48
Texel Engine 235
Texel— TEXture Element 206
Texture Color Keying 223
TIGA 48
TMS34020 48
TMU (Texture Mapping Unit) 236
TNT (TwiN Texel) 218
Translucency 223
Triangle Setup Engine 246
Triangle Setup 206, 235
Triangle strip and fans 257
Triangle Throughput 236, 245
Trident Microsystems 46
TRIDPCI.DRV 193
Tri-linear filtering, (TLF) 216
Trinitron. 15
True Color 47, 95, 105, 152, 163
TrueColor24 153
TrueColor32 153, 163
Truform 262
TTL 19
TTL-монитор 19, 27, 45
TV flicker filtering 249
TV-out функции 30, 197
TV-тюнер 197
TwiN Texel 249

U

UMA - Unified Memory Architecture 239
UMA (Upper Memory Area) 57
UNIX. 231
USB 242
USBSUP.EXE 242

V

VBE (VESA BIOS Extension) 47, 119
VCQ (Vibrant Color Quality) 253
Vertex Fog 222
Vertex Skinning 237
VESA BIOS 2.0 120
VESA версии 1.0 127
VESA версии 2.0 130
VESA 47
VESA-compatible 48
VESA-расширение B1OS 119
VESA-режимы 118
VESA-функции 130

VGA BIOS 65
VGA 18, 43, 55
VGA—TV-конверторы 30
VGA-совместимый адаптер 56
VHS 198
Video BIOS 14, 64, 65
Video CD 49
Video Gate Array 43
Video Graphics Adapter 43
Video Graphics Array 43
Video page 23
Video RAM (или VRAM) 57
Video-CD 197
VIDEORUS.EXE 22
VLB (VESA Local Bus) 48, 116
Voodoo Banshee 246
Voodoo Graphics 245, 246
Voodoo Rush 245
Voodoo 203
Voodoo^2 246
Voodoo^3 246
VRAM 114
VSA 236
V-Sync 96

W

Wait State 113
WARP (Windows Accelerator and Rendering Processor) 254
WAV 201
Web-узел 50
Wide z-buffer 214
Windows 95/98 83
Wireframe 204
WRAM 114

X

XMS (eXtended Memory Specification) 58
XGA 106

Y

Y/C 198
YUV 108

Z

z-aliasing 216
Z-sorting 214
z-алиасинг 216
z-буфер 206, 214, 236
z-буферизация 206, 214, 215
z-сортировка 206, 214

Русский

А

Абраш Майкл 211
Авто мип-мэппинг 216
Адаптеры 13
Адреса видеопамяти и Video BIOS 58
Адреса сегмента 57
Акселератор Windows 109
Акселератор 48, 106
Активная видеостраница 23
Алгоритм Брезенхема 156, 163
Алиасинг 108, 216
АЛУ 84, 110
Алфавитно-цифровой режим 33
Альфа-буфер 222
Альфа-смешение (alpha-blending) 206, 220, 221
Аналоговое смешение 199
Аналоговые (непрерывные) видеосигналы 19
Аналоговый RGB-сигнал 43, 95, 102
Аналоговый монитор 20, 43
Анизотропная фильтрация 216
Анимация двухмерных изображений 168
Антиалиасинг (antialiasing) 206, 225
Апертурная решетка 15
Аппаратная поддержка окон 107
Аппаратное ускорение графических функций 106
Аппаратное ускорение 55
Аппаратные прерывания 66
Аппаратный знакогенератор 22
Аппаратный курсор 108
Архитектура PC 12, 13
Атрибут символа 23
АФ 261
АЧХ 18

Б

Байт атрибутов 91, 101
Байтные операции 87
Банк памяти 64, 114
Билинейная фильтрация 216
Битовая плоскость 87
Бордюр (border) 82
Буфер текстур 235
Быстродействие динамической памяти 112

В

Ведомый видеоадаптер 191
Ведущий видеоадаптер 191
Векторная графика 207
Векторное представление графических примитивов 107
Вертекс 205
Вертикальное прерывание 96
Верхняя память 57
Взвешенное усреднение цветов 101
Видеоадаптер 8514/A 46, 105
Видеоадаптер EGA 30
Видеоадаптер SVGA 47
Видеоадаптер VGA 43
Видеоадаптер с графическим акселератором 109
Видеоадаптер 10, 12, 49, 55
Видеоадаптеры с графическим сопроцессором 109, 110
Видеокарта 12, 55
Видеоконтроллер адаптера EGA 36
Видеоконтроллер 14
Видеопамять 14, 20, 56, 111, 151, 235
Видеоплата 12, 33, 55
Видеорежим 20, 45, 98
Видеорежимы VESA 118
Видеосервис BIOS 58, 66
Видеосервис 66
Видеосигнал 20

Видеосистема PC 11, 12, 55
Видеосистема 10, 12
Видеостраница 23, 87
Видеостраницы 152
Виртуальная реальность 10, 49
Виртуальное разрешение 226
Виртуальные субпикселы 226
Витая пара 45
Внешние регистры 56, 97
Внутренняя шина данных видеоадаптера 65
Воспроизведение видеоинформации на PC 109
Время доступа (Access time) 112
Временная диаграмма 112
Вторичный видеоадаптер 191
Второй кадровый буфер 236
Высокоскоростные шины ввода/вывода 106
Высокоуровневые команды 107
Выходной регистр цвета 93

Г

Геометрическая стадия 3D-конвейера 207
Геометрический процессор 207, 235
Глубина пиксела 21
Глубина трехмерной сцены 214
Глубина цвета 21
ГП 259
Градации серого цвета 101
Градиентный метод 219
Графическая плата 14
Графические примитивы 106, 107
Графические процессоры 263
Графический 16-цветный режим 99
Графический акселератор 12, 107, 109
Графический интерфейс 21
Графический контроллер 13, 31, 84, 109
Графический процессор 11, 13, 49, 50, 109, 197, 203, 245
Графический режим 20, 21, 87, 99
Графический сопроцессор 49, 105, 106, 109, 110
Графический ускоритель 48, 109
Графическое ядро 237
Грубый рельеф 218

Д

Двойная буферизация 207, 236
Двумерная адресация 87
Двумерное маскирование 88
Двухбайтные номера VESA-режимов 118
Двухпортовая видеопамять 57, 114
Двухцветные режимы работы 102
Двухцветный графический режим 06h 102
Двухцветный текстовый режим 07h 102
Декодер 197
Декомпрессия сжатых видеоданных 109
Декомпрессия 49
Детектирование 198
Дизеринг (dithering) 207, 226
Динамическая память 111
Динамическое ОЗУ 96
Дополнительные графические функции 119
Дополнительные функции VBE 127
Дополнительный сегментный регистр данных ES 151
Драйвер видеоадаптера 82

Е

Единичные векторы 213

З

Задний буфер 236
Закон освещенности 209
Заливка (Fill) 107
Заполнения заданным узором (Pattern) 107
Затенение 209
Затуманивание (fogging) 206, 222
Защищенный режим работы 58
Зеркальные блики 223
Зерна люминофора 17
Зерна триады 15
Знакогенератор 33
Зона искажений 169

И

Импульсы гашения 16
Инвар 15
Индексный (адресный) регистр 76
Интерполяция цветов 226
Интерполяция 108
Интерфейс (Interface). 116
Интерфейс между видеоадаптером и материнской платой 115
Интерфейс прикладного программирования 229
Интерфейс с шиной ввода/вывода 56, 97
Информационные функции VBE 120

К

Кадр изображения 16
Кадровая синхронизация 16, 20
Кадровый буфер 20, 55, 64, 235
Каркасная модель объекта 204
Карта высот 219
Карта градиентов 219
Карта расширения 13
Карты затенения 211
Карты окружающей среды 219
Код адреса 57
Код символа 23
Кодирование цвета пиксела 98
Кодовая страница 1251 23
Кодовая страница 866 23
Кодовая страница 23
Команды передачи данных математического сопроцессора 141
Команды прорисовки 107
Компакт-диски 49
Композитный видеосигнал 197
Композитный выход 45
Композитный монитор 45
Компонентный видеосигнал 197
Компрессия 49
Конвейер рендеринга 235
Конвейерная (пакетная) передача данных 239
Конвертер 198
Конденсатор 111

Контроллер CRT 31
Контроллер 6845 27
Контроллер атрибутов 31, 91, 99
Контроллер синхронизации EGA 35
Контроллер синхронизации 88
Контроллер шины 12
Контроллер ЭЛТ 76
Контроллер 13
Коррекция дефектов изображения 209
Коррекция изображения 215
Коррекция перспективы 206
Коррекция цвета пикселов 206
Косвенное кодирование цвета 44
Краевой антиалиасинг 225
Кэш-память 112, 238

Л

Лестничный эффект 166, 215
Линейная адресация видеопамяти 152, 154
Линейная адресация 58, 105
Линейная интерполяция 210
Линейная фильтрация 216
Линейный режим адресации 154
Линейный туман 222
Линия возврата 45
Логическая структура адресного пространства PC 58
Логической длины строки 125
Локальная шина VLB 116
Люминофор 15

М

Макрос DeleteMImage 175
Макрос DrawMImage 175
Макрос 130
Максимальная скорость закраски 236
Максимальная частота спектра видеосигнала 18
Максимальное (физическое) разрешение 17
Маска прозрачности 156
Маски объектов 176
Масштабирование изображений 166
Масштабирование растровых изображений 108
Масштабируемость 3D-акселератора 236
Математический сопроцессор 140
Материнская плата 13
Матрица цветов 224
Матричная (линейная) алгебра 205, 235
Матричная организация 112
Менеджеры расширенной памяти 58
Мерцание экрана 44
Мерцание 151
Метод вдавливания 219
Метод затенения Гуро 206, 209, 210
Метод затенения Фонга 206, 209, 211
Метод масок 156
Метод цветовых плоскостей 64
Метод Родена 60, 186
Методы затенения 206, 209
Методы коррекции изображения 209
Механизм переключения банков памяти 105
Механизм прорисовки 235
Механизм установки 235
Минипорт 230
Мип-мэппинг 206, 215
Модификация цвета пиксела 84
Модулятор 15
Монитор VGA 44
Монитор 12
Монохромные режимы работы 101
Монохромный монитор 15, 101
Морфинг уровней детализации 223
Мультимедиа (multimedia) 10
Мультимедийные функции видеоадаптера 197
Мультимедийные функции 55
Мультимедийный PC 10, 21
Мультимониторная поддержка 192
Мультиплексированная шина 240
Мультисэмплинга 226
Мультитекстурирование 218, 246

Н

Настройка цвета текстуры 223
Недостатки видеоадаптера VGA 102
Неосновные цвета 19, 100
Непрорисовка 169
Новая схема линейной адресации 105
Номер символа 22
Нормаль 211
Нулевой регистр состояния 97

О

Обработка видеосигналов 197
Обратный ход луча 16
Общие регистры VGA 98
Общие регистры 34, 97
Общий модулятор 29
Объемное изображение 203
Однопортовая видеопамять 112
ОЗУ ЦАП 56
ОЗУ 12
Окаймление (overscan) 82
Определение номера регистра цвета ЦАП 10
Оптимизация 3D-конвейера 231
Оптимизированные текстуры 224
Орты 213
Основные графические функции 107
Основные характеристики видеопамяти 115
Основные цвета 15, 100
Отбрасывание задних граней 214
Оттенки цветов 19
Оттенок серого 102

П

Пакетный режим 112
Палетизированные текстуры 224
Палитра по умолчанию 100
Память для видеоадаптеров 111
Панорамирование 108
Пассивное устройство 55
Первичный видеоадаптер 191
Первый регистр состояния 97
Передний буфер 236
Переключение видеостраниц 152
Переключение страниц 154, 171
Переключение экранов 126
Перемещение курсора 152

Перенос блоков изображения из одной части экрана в другую 107
Перспективные типы видеопамяти 114
ПЗУ 13
Пиксел 17
Пикселизация 108
Пиксельные операции 87
Плоское затенение 209
Плоскопанельный ЖК-монитор 237
Плоскости памяти 32
ПО 12
Подпрограмма ExpMaskClear 180
Подпрограмма ShowGameResults 180
Подфункция 69, 100
Позиционирование курсора 152
Полигонный туман 222
Полные кадры 199
Полный антиалиасинг 225
Полный цветной телевизионный сигнал 29
Полоса пропускания видеотракта 44. 96
Полосы MIP-текстурирования 216
Полукадр 16, 17, 199
Порты ввода-вывода CGA 28
Порты ввода-вывода MDA 27
Пост-обработка (Post-processing) 207
Построение трехмерных изображений 109
Построчная развертка 197
Преобразование цветового пространства 108
Преобразователь развертки 199
Программа PlaneAndRocket. 183
Программно-изменяемые рельефные карты 219
Программные прерывания 66
Программный знакогенератор 22
Прокрутка экрана 126
Пропускная способность 3D-акселератора 236
Пропускная способность шины ISA 116
Прорисовка графических примитивов 107
Простые эффекты прозрачности 223
Протокол 116
Процедура Beep 133
Процедура BCD_to_ASCII 142
Процедура ClearScreen 133
Процедура CopyPlaneMask 180
Процедура Deleteimage 178
Процедура Doub1eF1oat_to 142
Процедура Doub1eF1oat_to_String 142
Процедура DoubleFloat_to_ExpForm 142
Процедура DrawMainBackground 180
Процедура DrawMovingImage 178
Процедура DrawStaticImage 178
Процедура Enable_A20 62
Процедура EVGALine 163
Процедура ExplosionFrame 180
Процедура FatalError 133
Процедура GC1earScreen 174
Процедура GclearScreen 161
Процедура GetChar 133
Процедура GetFloat 142
Процедура GetInteger 142
Процедура GrabRusFont 157
Процедура GshowBinDWord 158
Процедура GshowByteBinCode 158
Процедура GshowByteHexCode 157
Процедура GshowDecByte 180
Процедура GshowDecDWord 180
Процедура GShowDecWord 180
Процедура GshowHexDWord 157
Процедура GshowHexWord 157
Процедура GshowString 157
Процедура InitEpisode 180
Процедура Int32_to_String 142
Процедура MathFunctionsTest 149
Процедура MirrorPlaneMask 180
Процедура PutGraChar 161, 174
Процедура SetCursorPosition 132
Процедура SetLAddrModeForGS 61
Процедура SetLAddrModeForGS 62
Процедура SetTrueCo1or32 157
Процедура SetVESAVideoMode 157
Процедура ShowASCIIChar 132
Процедура ShowASCIIField 132
Процедура ShowBackground 178
Процедура ShowBinByte 132
Процедура ShowBinDWord 132
Процедура ShowColorString 132
Процедура ShowColorText 132
Процедура ShowDataString 142
Процедура ShowDecByte 142
Процедура ShowDecDWord 142
Процедура ShowDecWord 142
Процедура ShowEscapedPlanes 178
Процедура ShowHexByte 132
Процедура ShowHexDWord 132
Процедура ShowHexWord 132
Процедура ShowLargeChar 167
Процедура ShowRusFont 166
Процедура ShowString 132
Процедура ShowText 132
Процедура ShowVESAString 130
Процедура String_to_Doub1eF1oat 142
Процедура String_to_Int32 142
Процедура VESA_BIOS_Test 130
Процедура Wait8042BufferEmpty 62
Процедура WaitChar 133
Процедура WaitVSync 157
Прямая запись в видеопамять 133
Прямое кодирование цвета пиксела 47, 95, 105
Прямой ход луча 16
Псевдографика 22
ПЦТС 29

Р

Рабочий цикл памяти 112
Радиоприемник 198
Разгон видеоадаптера 97
Размер адресного пространства 57
Размер палитры 19, 44
Разметка экрана на знакоместа 157
Разрезание картинки 169
Разрешение изображения 16
Разрешение по вертикали 17
Разрешение по горизонтали 17
Разрядность z-буфера 214
Разрядность шины памяти 237
Раппер 231
Распределение видеопамяти 154

Распределенная вычислительная структура 107
Растеризация 207
Растр 15
Растровая графика 206
Растровое изображение 213
Растровые объекты 207
Расчет освещенности 205
Расширенная память 58
Расширенные функции Video BIOS 118
Реальный режим работы 58
Регенерация 112
Регистр AX 66
Регистр безразличного цвета 88, 91, 98
Регистр битовой маски 88
Регистр выбора цвета 93, 100
Регистр маски 95
Регистр смешанного вывода 34
Регистр состояния 57
Регистр-защелка 65
Регистровая модель 117
Регистры RAMDAC 94
Регистры видеоадаптера EGA 34
Регистры графического контроллера EGA 41
Регистры графического контроллера 85
Регистры контроллера CRT 36
Регистры контроллера атрибутов EGA 39
Регистры контроллера атрибутов VGA 91
Регистры контроллера ЭЛТ 76
Регистры общего назначения 66
Регистры палитры ЦАП 97
Регистры палитры 32, 99
Регистры секвенсора 89
Регистры цвета ЦАП 93, 94
Регистры 56
Регистры-защелки 87
Режим 03h 96
Режим 1024×768/256 94
Режим 800×600/256 94
Режим расщепления экрана 31
Режим сегментации видеопамяти 154
Режимом с палитрой 224
Режимы «2x» и «4x» 239
Режимы адресации видеопамяти 154
Рельефная карта 219
Рельефное текстурирование 219, 238
Родена (метод) 60, 186

С

Сегменты памяти 57
Секвенсор 56, 88
Селекторы и порты ввода-вывода 127
Сжатие видеоизображений 49
Сжатие текстур 215, 224
Сигналы RGB 16, 30
Сигналы Y, U и V 198
Сигнатура 'VESA' 130
Символьная матрица 22, 101
Символьные позиции 21
Синхронизатор 31, 56, 96
Синхросмесь 16, 20
Системная шина 12
Системы виртуальной реальности 10
Скалярное произведение 213
Сканирование видеопамяти 20, 89
Сканирование экрана 16
Скорость обновления 16
Слот AGP 241
Слот шины ISA 43
Слоты расширения 191
Смешение RGB- и TB-сигналов 199
Смещение 57
Снег CGA 30, 96
Современные видеоплаты 56
Спектральная чувствительность глаза 101
Специализированное ОЗУ 57
Спецификация VESA 117
Спрайт 108
Стадия прорисовки объекта 207
Стадия рендеринга 207
Стандарт DPMS 127
Стандарт VESA 47, 106
Стандартная память 57
Стандартное адресное пространство 57
Стандартные SVGA-режимы 47
Стандартные видеорежимы 46
Стандартный видеоадаптер VGA 55
Стандартный телевизионный видеосигнал 16
Статическая память 112
Статические изображения 154
Статус видеоадаптеров 191
Стек сопроцессора 141
Степень прозрачности объекта 222
Стереопара 203
Страница памяти 64
Стробирующие сигналы 76
Строка развертки 16
Строчная синхронизация 16, 20,
Структура видеоадаптера с графическим акселератором 110
Структура видеоадаптера с графическим сопроцессором 111
Структура рельефа 218
Структура формирования видеосигнала 33
Структурная схема 3D-акселератора 235
Структурная схема TV-тюнера 197
Структурная схема адаптера EGA 32
Структурная схема видеоадаптера VGA 56
Субпиксел 226
Субпиксельная коррекция 206
Субпиксельный антиалиасинг 225
Суммирование цветов 222
Суперсэмплинг 226
Схема формирования палитры 100
Схема формирования цвета 99
Схемы кодирования цвета 99

Т

Таблица векторов прерываний 66
Таблица доступа к интерфейсу защищенного режима 127
Таблица поиска цветов 223
Таблица портов и областей памяти 127
Табличный туман 222
Тайловая архитектура 226
Тактильные ощущения 10
Тактовые генераторы 56, 96

Тактовый сигнал 96
Тексел (Texel) 206, 216
Текстовый режим 20, 21, 99, 151
Текстура (texture) 206, 117
Текстурирование с коррекцией перспективы 218
Текстурирование 206
Текущая палитра 44, 94
Телевизионные тюнеры 49
Телевизионный радиосигнал 197
Теневая маска 15
Теневая память 66
Тесселяция 204, 210
Технология двойной буферизации 114
Тип развертки 197
Ток утечки 112
Тонкий рельеф 219
Трансформация 205
Трехканальный ЦАП 93
Трехмерная графика 11, 203
Трехмерная сцена 203
Трехступенчатая конвейерная адресации 113
Триангуляция 204
Триггер 112
Трилинейная фильтрация 216, 246
Тюнер 198

У

Удаление скрытых поверхностей 213
Универсальная графическая видеоплата 55
Упакованный десятичный формат 141
Управление курсором 152
Управляющие символы ASCII-кода 138
Управляющие функции VBE 123
Упреждающее чтение 115
Ускоренный графический порт 238
Ускорители 3D-графики 235
Ускорители двумерной и трехмерной графики 10, 109
Ускоритель трехмерной графики 11
Устройство адресации ЦАП 94
Устройство отображения 12
Утилита 82

Ф

Фактическая палитра 100
Фильтр мерцаний 199
Фильтрация 216
Формат BMP 186
Формат PCX 186, 187
Формат PDF 64
Формат YUV 108
Формат заголовка файла PCX 187
Формат информации VESA BIOS 120
Формат информации о видеорежиме 121
Формат кодирования цвета символа 151
Формат строки таблицы палитры 126
Форматы графических файлов 186
Форматы данных математического сопроцессора 141
Фотореалистичная графика 237
Фрейм-буфер 20
ФС 261
Функции 4Fh прерывания int 10h VESA BIOS 130
Функции TV-out 20
Функции Video BIOS 66, 100
Функция 69, 100

Ц

ЦАП 43, 93, 152
Цвет символа 151
Цвет фона знакоместа 151
Цветной монитор 15
Цветовая модель 49
Цветовое пространство 198
Цветовые коды 151
Цветовые плоскости 64, 87, 101
Цветоделительная маска 15
Цветоразностные сигналы U и V 108, 198
Цикл обращения к памяти 112
Цифровая видеостудия 49
Цифровая обработка видеосигнала 48
Цифровая обработка и воспроизведение видеоданных 11
Цифровая обработка изображений 48
Цифровое видео 197
Цифровое смешение 200
Цифровой видеосигнал 19
Цифровой монитор 19
Цифровые видеопотоки 237

Ч

Частота кадров 16
Частота строк 16
Чересстрочная развертка 16, 197
Черно-белое изображение 101
Чипсет (Chipset) 11

Я

Ш

Шина AGP 117
Шина адреса 57
Шина ввода/вывода 12, 49, 115
Шина PCI 117
Шина ISA 116
Шум текстуры 216

Щ

Щелевая маска 15

Э

Экспоненциальный туман 222
Электронная пушка 15
Электронный луч 15
ЭЛТ 12, 15,
Энергосберегающие функции монитора 127
Эффект разрезания изображения 97, 126
Эффекта пикселизации 215
Эффекты полпрозрачности 206
Эффекты прозрачности 206

Я

Яркость (Y) пиксела 101
Ячейка динамического ОЗУ 111

Список литературы

1. Попов С.Н. Аппаратные средства мультимедиа. Видеосистема PC/ Под ред. О.В. Колесниченко, И.В. Шишигина - СПб.: БХВ-Петербург; Арлит. 2000. - 400с.: ил.
2. Павлов В.А. Устройства отображения информации ПК. Учебно-справочные материалы. СарФТИ, Саров. 2003. - 305 с.: ил.
3. ТПП "Сфера" авторский коллектив. Аппаратные средства ПЭВМ. Справочное руководство по IBM PC. Часть 2 - Москва. ПЭМ 1991. - 301с.: ил.
4. PCBOOK - Аппаратные средства PC. - Материалы на электронном носителе в среде MS DOS.
5. Кулаков В. Программирование на аппаратном уровне: Специальный справочник. 2-е изд. СПб.: Питер, 2003. - 848 с.: ил.
6. Рудометов Е., Рудометов В. Устройство мультимедийного компьютера. - СПб.: Питер, 2001. - 512с.: ил.
7. Воронов М.А., Родин А.В., Тюнин Н.А. Ремонт мониторов. Выпуск 12. 2-е издание. Москва: "СОЛОН-Р" 2000. - 299с.: ил.
8. Денисов О., Назаров С. Графические процессоры // PC Magazine/Russian Edition (Компьютер сегодня), 2002, №6. - С. 48-65.
9. Гук М. Аппаратные интерфейсы ПК. Энциклопедия. -СПб.: Питер, 2002. -528 с.; ил.
10. Гук М. Аппаратные средства IBM PC. Энциклопедия, 2-е изд. -СПб.: Питер, 2001. -928 с.; ил.
11. Кулаков В. Программирование на аппаратном уровне: Специальный справочник. СПб.: Питер, 2001. - 496 с.: ил.
12. Павлов В.А. Учебные материалы по курсу "Периферийные устройства ЭВМ", Часть 1. СарФТИ, Саров. 2001. - 231 с.: ил.
13. Абраш М. Таинства программирования графики. - К.: ЕвроСИБ, 1996
14. Кенцл Т. Форматы файлов Internet/ Перевод с англ. - СПб: Питер, 1997. -320 с.: ил.
15. Аппаратные средства ПЭВМ. Справочное руководство по IBM PC. Часть 1 - Москва. ТПП "Сфера" ПЭМ 1991. - 173с.: ил.
16. Богумирский Б. Энциклопедия Windows 98 (второе издание) - СПб: Питер Ком, 2000. - 896 с.: ил.
17. Григорьев В.Л. Видеосистемы ПК фирмы IBM. - М.: Радио и связь, 1993. - 192 с.: ил.

Учебное издание

ПАВЛОВ Виктор Александрович

ВИДЕОАДАПТЕРЫ
ПК

Учебное пособие для вузов