Тема 4. Відеоадаптер. Програмування відеодаптера

4.1. Основні поняття

Відеосистема комп'ютера призначена для формування графічних зображень на екрані монітора. Основна функція відеосистеми – вибір даних із відеобуфера і перетворення їх у сигнали, що створюють зображення на екрані. Центральний процесор не займається управлінням дисплея, тобто не виконує попіксельне прораховування 3D-сцени, а лише посилає спеціальному пристрою – відеоадаптеру просторове положення об'єктів, які необхідно відобразити. Ці об'єкти складаються з величезної кількості многокутників. Каркас многокутників необхідно покрити поверхнею, щоб одержати реалістичне зображення на екрані комп'ютера. Цю роботу і бере на себе відеокарта.

Отже, *відеоадаптер (відеокарта)* – це пристрій (плата), що керує роботою монітора (тобто безпосередньо передає сигнали керування блокам монітора для виведення інформації на екран монітора) та взаємодією з центральним процесором комп'ютера.

Екран дисплея – це прямокутна матриця окремих дискретних точок (пікселів), які формують зображення. Кількість пікселів визначає *роздільну здатність дисплея* і задається парою чисел. Перше число показує кількість пікселів у рядку, а друге – кількість рядків (наприклад, 640 × 480, 1024 × 768, 1280 × 1024). Чим більші ці числа, тим меншими є самі пікселі. Це приводить до того, що вони сприймаються не як окремі точки, а як єдине цільне зображення.

Кожному пікселю в деякій частині загального адресного простору, що називається *відеопам'яттю*, або *буфером кадра*, ставиться у відповідність фіксоване число бітів – *атрибут пікселя* (код кольору). Ці біти і визначають колір пікселя з заданого набору кольорів.

Кількість бітів, що відводиться для опису кольору однієї точки растра, називається *роздільною здатністю бітової глибини*, *кольоровою роздільною здатністю* або просто *глибиною кольору* чи *глибиною буфера кадра.* Глибина кольору може набувати значення 1, 2, 4, 8, 16, 24 і навіть 32 біти. Якщо глибина кольору дорівнює *n* бітів, то графічні зображення на екрані можуть мати на екрані 2^n відтінків.

В дисплейних адаптерах з монохромним монітором значення атрибута керує інтенсивністю одного променя, тобто яскравістю точки, а з кольоровим монітором – інтенсивністю трьох променів, що задають три компоненти кольору пікселя. Кожний кольоровий піксель утворюється трьома меншими ділянками червоного, зеленого і синього кольору, які при змішуванні визначають результуючі кольори пікселів.

Відеопам’ять є складовою частиною відеоадаптера. Фізично відеопам’ять організована у вигляді одновимірного вектора в загальному адресному просторі. Відеопам’ять реалізована у вигляді спеціальних типів мікросхем із довільним доступом (RAM), які дозволяють швидко вивести вміст буфера на екран. Як правило, адреса першого байта відеопам’яті має значення A000:0000. В усіх графічних режимах стартова адреса відеопам’яті відповідає лівому верхньому пікселю на екрані. Відеоадаптер циклічно з частотою кадрів монітора (75–120 разів за секунду) зчитує вміст відеопам’яті й постійно формує зображення на екрані монітора, при цьому колір кожного пікселя визначається поточним значенням його атрибута. Зображення на моніторі повністю відповідає поточному вмісту відеопам’яті. Ємність відеопам’яті сучасних відеокарт досягає 1024 Мб. Така ємність потрібна для зберігання й оброблення великої кількості даних (буфер кадра, буфер глибини, параметри вершин, текстури тощо), що використовуються графічним процесором для побудови тривимірних рухомих зображень на екрані монітора. Ємність графічної пам’яті визначає складність зображень, параметри якості та швидкості відображення.

Програма, що виконується на комп’ютері в графічному режимі, має доступ (читання/запис) до всіх комірок відеопам’яті. Для збереження декількох кадрів зображення в певних режимах у відеопам’яті передбачені окремі області однакової структури, кожна з яких містить атрибути пікселів екрану. Ці області називаються *відеосторінками*. Їх кількість є степенем двійки. У даний момент тільки одна сторінка зображена на екрані (програма в цей момент може працювати з неактивною сторінкою). Наявність сторінок дає можливість програмі миттєво змінювати зображення на екрані і в такий спосіб усувати ефект миготіння. Використання відеосторінок відіграє важливу роль при мультиплікації. Зміна відеосторінок створює ефект руху зображень на екрані.

Для зміни поточної сторінки можна викликати відповідну функцію BIOS, або безпосередньо змінити вмістиме регістра початкової адреси, при цьому процесор записує інформацію безпосередньо у відеопам’ять. Графічна бібліотека дозволяє здійснювати роботу з будь-якою сторінкою. Активна сторінка встановлюється процедурою setactivepage.

Відеоадаптер включає в себе, крім відеопам’яті, в якій зберігається зображення на екрані монітора, ще й постійно запам’ятовуючий пристрій, в якому записані набори шрифтів для текстових та графічних режимів відеоадаптера і функції BIOS для роботи з відеоадаптером. Окрім цього, відеоадаптер містить складні керуючі пристрої, що забезпечують обмін даними та формування зображення.

У сучасних відеоадаптерах більша частина графічних функцій реалізована безпосередньо на апаратному рівні, що вимагає використання потужного спеціалізованого графічного процесора, який за складністю наближається до центрального процесора, причому, можливо, не одного, а декількох, кожен з яких виконує свій набір графічних функцій. Наприклад, графічний процесор ATI Radeon X1800 XT містить 320 млн. транзисторів і декілька десятків спеціалізованих процесорів.

Перші графічні процесори відеоадаптера використовувалися для виконання операцій виведення ліній, полігонів (плоских елементів). Сучасні графічні процесори (GPU, G80) виконують багато базових операцій 3D-графіки, наприклад підтримку *Z*-буфера, накладання текстур, формування ефектів туману, відбитого світла, прозорості тощо. Оскільки відеоадаптери виконують ці операції апаратно, що дозволяє пришвидшити їх у сотні, тисячі та мільйони разів (залежно від складності алгоритму) в порівнянні з програмною реалізацією, то нове покоління відеоадаптерів для роботи з тривимірною графікою називають ще *графічними прискорювачами* або *графічними акселераторами*.

Відеоадаптери можуть відрізнятися не тільки за швидкодією, можливостями роботи з кольором, але й за рівнем реалізації тих чи інших графічних операцій. Наприклад, відеоадаптер Matrox дозволяє створювати якісну двовимірну графіку, Nvidia GeForce – потужний ігровий 3D-акселератор, 3DLabs Wildcat використовується для професійного 3D-моделювання, для створення віртуальної реальності.

Відеоадаптер може бути оформлений у вигляді окремої плати, яка вставляється в слот розширення комп’ютера, або безпосередньо розміщуватися на системній платі.

Для виведення графічних зображень, особливо в режимі анімації, потрібна висока швидкість передачі даних для того, щоб досягти високої частоти кадрів, тому важливою рисою персонального комп’ютера з позицій графіки є те, що сучасні відеоадаптери під’єднується до системної шини за допомогою швидкісної локальної шини AGP (Accelerated Graphics Port) або до більш сучасного послідовного інтерфейсу PCI Express. Це дає можливість швидко вести обмін даними між оперативною пам’яттю та відеопам’яттю і підвищити швидкодію комп’ютера.

Розрядність шини AGP – 64 біти. На частоті 66 МГц вона забезпечує швидкість обміну 528 Мб/с. Це був стандарт AGP 2x. Зараз AGP працює на більш високих частотах. У 1999 р. з’явився режим AGP 4x (швидкість передачі даних – 1,06 Гб/с). В 2002 р. просунуті відеокарти отримали значок AGP 8x. Максимальна пропускна здатність шини PCI Express X16 становить 4000 Мб/с у двох напрямках.

Відеоадаптери можуть працювати в різних текстових і графічних режимах (відеомодах), які відрізняються роздільною здатністю, кількістю кольорів на екрані дисплея, кількістю відеосторінок тощо.

Нині існує велике різноманіття відеоадаптерів, починаючи від монохромних, які не підтримують графічні режими, і закінчуючи сучасними відеоадаптерами, які відтворюють 16,7 млн. кольорів і більше.

Можна виділити декілька типів відеоадаптерів (табл. 4.1). Як видно з табл. 4.1, перший відеоадаптер MDA не міг відображати графіки на моніторі. Крім цього, він працював із монохромним монітором. Більшість відеоадаптерів, описаних на початку табл. 4.1, сьогодні не задовольняють користувачів і не використовуються на практиці.

Із появою операційної системи Windows вимоги до відеосистеми різко зросли. Ні EGA, ні VGA не забезпечували необхідної роздільної здатності та достатньої кількості кольорів. Нині різні фірми випускають адаптери SVGA, XGA, які працюють у режимах High Color та True Color.

Таблиця 4.1

Відеоадаптер	Рік випуску	Обсяг ОЗУ (Кб)	Роздільна здатність	Кількість кольорів
MDA (Monochrome Display Adapter)	1981	4	40 × 25с.	2
CGA (Color Graphics Adapter)	1981	16	320 × 200 640 × 200	4 2
Hercules	1982	32	720 × 350	2
EGA (Enhanced Graphics Adapter)	1984	64	640 × 350	16
VGA (Video Graphics Array)	1987	256	640 × 480 320 × 200	16 256
SVGA (Super VGA)	1990	512	800 × 600 і вище	256
XGA (Extended Graphics Array)	1991	1024	1024 × 768 і вище	256

У режимі High Color відеоадаптер може одночасно відображати на екрані 32768 або 65536 різних кольорів, у режимі True Color – 16777216. Окрім перелічених стандартів, існують ще такі стандарти (адаптери): SXGA (Super XGA) – стандарт для роздільної здатності 1280 × 1024 та UXGA (Ultra XGA) – для роздільної здатності екрана 1600 × 1200 і більше із глибиною кольору 32 біти на піксель. Слід зазначити, що такі відеоадаптери вимагають і відповідних моніторів.

Здатність відеоадаптера відображати велику кількість кольорів із високою роздільною здатністю може бути забезпечена лише відповідним обсягом відеопам'яті. Обсяг відеопам'яті – одна з найважливіших характеристик відеопам'яті. Чим більший обсяг відеопам'яті, тим більше кольорів може відображатися на екрані і тим більшою буде максимальна роздільна здатність екрана. Мінімальний обсяг відеопам'яті (в Кб) при відповідних вимогах можна визначити з табл. 4.2.

4.2. Режими відеоадаптерів

Практично кожен відеоадаптер підтримує декілька режимів роботи (videomodes), які відрізняються один від одного роздільною здатністю і кількістю кольорів. Різні режими навіть одного адаптера мають різну організацію відеопам'яті та способи роботи з нею. Кожен режим має свій номер. У залежності від відеорежиму визначається логічна організація відеопам'яті.

Таблиця 4.2

Кількість кольорів	Роздільна здатність екрана			
	640 × 480	800 × 600	1024 × 768	1280 × 1024
16	150	235	384	640
256	300	469	768	1280
65536	600	938	1536	2560
16777216	900	1407	2304	3840

Більшість адаптерів створюється за принципом сумісності з попередніми. Так, адаптер VGA підтримує всі режими адаптера EGA і т.д.

Установкою режиму роботи керує переривання BIOS 10H, а саме функція 00H може задати будь-який стандартний режим роботи відеоадаптера. Наведемо процедуру установки режиму *v*, використовуючи процедуру intr:

```
procedure mode (v : integer);
var     r : Registers;
begin
  r.ax:=v; {r.ah:=0 – номер функції; r.al:= v – номер режиму}
  intr($10, r); {звернення до переривання}
end; {mode}
```

Список деяких стандартних режимів відеоадаптерів наведений у табл. 4.3.

Таблиця 4.3

Номер режиму	Тип режиму	Відеоадаптер	Роздільна здатність	Кількість кольорів	Коефіцієнт масштабування	Адреса відеопам'яті
00H	текст.	CGA, EGA, VGA, SVGA	40 × 25	16	–	B000:0000H
04H	граф.	– || –	320 × 200	4	1.2	B800:0000H
0DH	граф.	EGA, VGA,	320 × 200	16	1.2	B800:0000H
10H	граф.	EGA, VGA, SVGA	640 × 350	16	1.37	A000:0000H
12H	граф.	VGA, SVGA	640 × 480	16	1.0	A000:0000H
13H	граф.	VGA, SVGA	320 × 200	256	2.4	A000:0000H

Масштабний коефіцієнт визначається відношенням кількості пікселів на дюйм у горизонтальному напрямку до кількості пікселів на дюйм у вертикальному напрямку. Масштабування потрібно проводити для того, щоб зображення виглядали реальніше: інакше квадрат буде прямокутником, а коло – еліпсом.

Відеоадаптери SVGA відрізняються більш потужними графічними контролерами й організацією відеопам'яті. Всі вони підтримують стандарт VGA і можуть працювати також у режимах зі значно кращими характеристиками. Характеристики режимів відрізняються у відеоадаптерів різних фірм, оскільки різні моделі SVGA мають різні набори регістрів. А це створювало труднощі при написанні програм для SVGA.

Асоціація стандартів в області відеоелектроніки VESA (Video Electronic Association) розробила стандарт на відеорежими, що охоплював майже всі можливості сучасних відеоадаптерів, тобто створено стандарт на графічні функції BIOS, який дозволяє керувати відеоадаптерами SVGA. Цей стандарт описує розширення переривання BIOS 10H (Vesa BIOS Extension – VBE), що здійснює управління відеоадаптером. Підтримка VBE включається на заводі у BIOS відеоадаптера.

Широкі можливості для використання SVGA надає операційна система Windows. У ній використовуються спеціальні драйвери, що виконують роботу з програмування відеодаптерів. Тому копітка робота з регістрами адаптера схована від програмістів, які пишуть програми під Windows.

Всі функції VBE доступні через функцію 4FH переривання 10H. Таких функцій у версії VESA 2.0 одинадцять із номерами від 0 до 10. Наприклад, функція 0 повертає інформацію про версію VESA та про виробника відеоплати, функція 1 повертає інформацію про конкретний

відеорежим, функція 2 встановлює відеорежим, функція 3 повертає номер поточного режиму і т.д. Перед викликом функцій VBE необхідно записати в регістр AH значення 4FH, а в регістр AL – номер функції. Якщо ця функція в VBE реалізована, то в регістр AL повертається значення 4FH, інакше – якесь інше значення. У результаті успішного виконання функції в регістр AH повертається нульове значення, якщо в регістр AH записано значення 1H, то функція завершилася з помилкою.

Наведемо текст процедури, яка встановлює VESA-режим vmode.

```
procedure Set_SVGA_mode (vmode : integer);
var r : Registers;
  b : integer;
begin
  r.ah:=4FH;
  r.al:=02H; {номер функції встановлення графічного режиму}
  r.bx:=vmode; {номер режиму}
  intr($10, r);
  if ((r.al = 4FH) and (r.ah = 0)) then b:=0 else b:= –1;
end; { Set_SVGA_mode }
```

У BIOS введено деякий стандартний набір розширених режимів, кожен з яких має власний номер (код) згідно зі стандартом VESA. Номер режиму є 16-бітним числом, де біти 9–15 зарезервовані і дорівнюють нулю, а біт 8 для VESA-режимів дорівнює 1.

Наведемо таблицю деяких стандартних VESA-режимів (табл. 4.4).

Таблиця 4.4

Номер режиму	Тип режиму	К-ть кольорів	Роздільна здатність
100H	граф.	256	640 × 400
101H	граф.	256	640 × 480
102H	граф.	16	800 × 600
103H	граф.	256	800 × 600
104H	граф.	16	1024 × 768
105H	граф.	256	1024 × 768
106H	граф.	16	1280 × 1024
107H	граф.	256	1280 × 1024
108H	текст.	16	80 × 60 (на 1 с.)
⋮			
10DH	граф.	32768	320 × 200
⋮			

110H	граф.	32768	640 × 480
111H	граф.	65536	640 × 480
112H	граф.	16777216	640 × 480
⋮			
116H	граф.	32768	1024 × 768
117H	граф.	65536	1024 × 768
118H	граф.	16777216	1024 × 768
⋮			
11AH	граф.	65536	1280 × 1024
11BH	граф.	16777216	1280 × 1024

4.3. Програмування відеоадаптерів у простіших режимах

Фактично будь-яка графічна операція зводиться до роботи з окремими пікселями, наприклад, вивести піксель заданого кольору. Цю задачу можна розв'язувати безпосереднім доступом до відеопам'яті, але при цьому потрібно знати структуру відеопам'яті у відповідному відеорежимі. *Структура відеопам'яті* – це спосіб упаковки інформації про пікселі в адресному просторі відеопам'яті та відображення цієї інформації на прямокутний растр дисплея.

Розглянемо приклади процедур виведення пікселів у деяких простіших графічних режимах [38].

Режим 4. Це режим 320 × 200 пікселів з чотирма кольорами. Цей режим підтримується відеоадаптерами CGA, EGA, VGA, SVGA. У адаптерів EGA, VGA, SVGA дані містяться в нульовому плані (решта планів не використовуються). Кожному пікселю відповідає 2 біти відеопам'яті. Колір верхнього лівого пікселя зберігається в бітах D7, D6 нульового байта відеопам'яті, наступного – в бітах D5, D4 і т.д. (рис. 4.1).

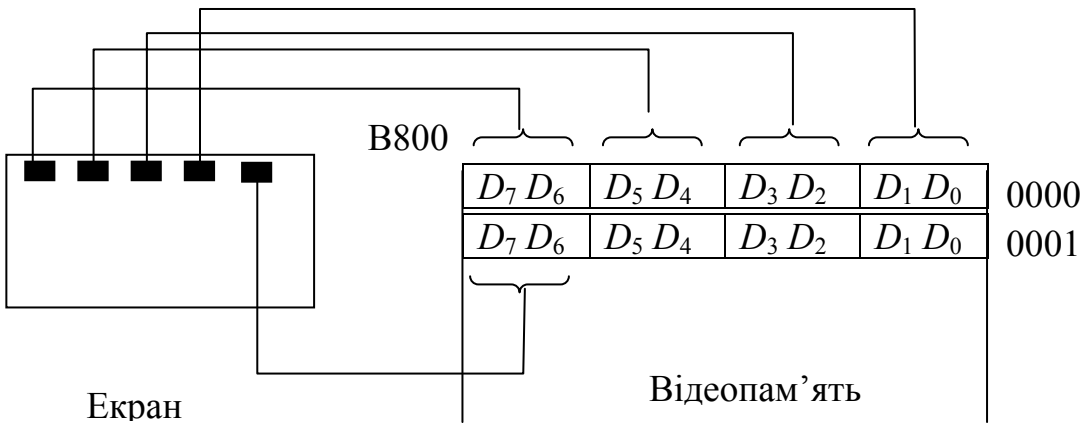

Рис. 4.1. Структура відеопам'яті в четвертому графічному режимі

Зміщення байта, що містить дані про піксель (x, y) можна знайти за формулою

зміщення = 40 * y + ціла частина від ділення x на 4,

оскільки 320 пікселів займає 80 байтів відеопам'яті, але пікселі парних та непарних рядків знаходяться в різних місцях відеопам'яті. Всі непарні рядки записуються, починаючи з адреси B800:0000, а парні – з B800:2000.

Щоб знайти номери бітів у байті пам'яті, використовують формулу:

$$S = (3 - x \bmod 4) * 2.$$

Запишемо процедуру виведення пікселя кольору c з координатами x, y.

```
Procedure  putpixel_4(x, y, c : integer);
  var z, m, s : integer;
begin
  if (0 <= x) and ( x <=319) and (0 <= y) and (y <= 199) then
        begin
              z:=40*y + x shr 2;
              if odd(y) then z:=z+8192;
              s:= (3 – x and 3) shl 1;
              m:=3 shl s;
              mem[$B800:z]:=mem[$B800:z] and not m or (c shl s);
        end;
end;{putpixel_4}
```

Застосовуючи процедуру putpixel_4, досить просто можна намалювати відрізок горизонтальної лінії між точками з абсцисами x_1, x_2, за допомогою оператора (необхідно, щоб $x_1 \le x_2$):

```
for x:=x1 to x2 do putpixel_4(x, y, c);
```

Вертикальний відрізок зображається за допомогою оператора

```
for y:=y1 to y2 do putpixel_4(x, y, c);
```

У циклі виконуються прості операції – приріст на одиницю і перевірка менше або дорівнює, тому операція малювання здійснюється швидко. Однако виникає питання: чи не можна побудувати більш швидкодіючі алгоритми, використовуючи записи відповідних бітів у пам'ять, як це було зроблено в процедурі putpixel_4?

У цьому випадку необхідно враховувати логічну організацію відеопам'яті комп'ютера, в якій зберігаються біти/байти растра. Якщо записувати в четвертому режимі одразу 4 сусідні пікселі (1 байт у пам'яті), то можна істотно пришвидшити виведення горизонталей.

Використовуючи міркування для процедури putpixel_4, можна побудувати швидкий алгоритм виведення горизонтальної лінії. Для цього за значеннями координат x_1 та x_2 потрібно визначити адресу першого та

останнього неповних байтів. Для цих байтів необхідно виробити маску запису кольору. Повні байти будуть заповнені чотирма пікселями, тобто значенням *c c c c*.

Процедура побудови горизонтального відрізка в четвертому графічному режимі з прямим доступом до відеопам'яті набуде такого вигляду:

```
procedure  hor(x1, x2, y, c : integer);
var c1, z1, z2, M1, M2, i : integer;
begin
  if ( 0 <= x1) and (x2 <= 319)  and (x1 <= x2)
                    and ( 0 <= y) and (y <= 199) then
  begin
        z1:=y*40 + x1 shr 2 + 1;    {адреса першого повного байта}
        z2:=y*40 + x2 shr 2 – 1;  {адреса останнього повного байта}
        if odd(y) then
              begin
                 z1:=z1+8192;
                 z2:=z2+8192;
              end;
        c1 := (c shl 6) or (c shl 4) or (c shl 2) or c; {маска повного байта}
        for i:=z1 to z2 do
              mem[$B800:i]:= c1;
        z1:= z1 – 1; {адреса першого неповного байта}
        z2:= z2 + 1; {адреса останнього неповного байта}
        M1 := 255 shr ((x1 and 3) shl 1); {маска першого байта}
        M2 := 255 shl ((3 – x2 and 3) shl 1); {маска останнього байта}
        if (z1 < z2) then  begin
        mem[$B800:z1]:=mem[$B800:z1] and (not M1) or (c1 and M1);
        mem[$B800:z2]:=mem[$B800:z2] and (not M2) or (c1 and M2);
              end
        else
        begin
        M1 := M1 and M2;
        mem[$B800:z1]:=mem[$B800:z1] and (not M1) or (c1 and M1);
        end;
  end;{if}
end;{hor}
```

Для вертикалі бітова маска однакова для всіх пікселів, її слід заповнювати в циклі, але кількість кроків у такому циклі буде значно більшою, тому при однаковій довжині ліній горизонталі зображаються на екрані швидше, ніж вертикалі.

Режим 13H. Цей режим забезпечує 256 кольорів для пікселів та роздільну здатність 320 × 200. Атрибут кожного пікселя визначається 1 байтом. Відеопам'ять організована лінійно.

Зміщення байта, що містить дані про піксель (x, y), від початку сторінки відеопам'яті визначається за формулою $z = 320 * y + x$.

Процедура виведення пікселя в 13H режимі має такий вигляд:

```
procedure  putpixel_13(x, y, c : integer);
   begin
     if (0 <= x) and ( x <=319) and (0 <= y) and (y <= 199) then
     mem[$A000:y*320 + x]:=c;
   end; { putpixel_13 }
```

4.4. Програмування відеоадаптерів у режимах OEH, 10H, 12H

Ці режими працюють з 16 кольорами і різною роздільною здатністю: 0EH – 640×200; 10H – 640×350; 12H – 640×480 та підтримуються адаптерами VGA, SVGA. Ділянка пам'яті A000:0000H не зберігає даних про колір пікселя. Для цього VGA використовує свою внутрішню пам'ять, що не є частиною оперативної пам'яті та безпосередньо не доступна програмісту. Ця внутрішня пам'ять представляється шарами, або планами. Кожному пікселю відповідає один біт із кожного плану.

Чотири біти на піксель дозволяють відображати 16 різних кольорів (табл. 3.1). Плани мають номерацію від 0 до 3 і свою адресацію байтів (рис. 4.2). Існує жорсткий зв'язок між планами і відеопам'яттю. VGA контролює всі операції з відеопам'яттю, перехоплює їх та інтерпретує по-своєму. Наприклад, запис у відеобуфер означає запис у плани (існує запис байтами та точками). Ці всі операції виконуються через численні регістри VGA, які заповнюються через відповідні порти.

Як приклад наведемо групу регістрів, що належить складовій частині відеокарти – графічному контролеру (табл. 4.5).

Таблиця 4.5

Номер	Регістр	Стандартні значення
0	Set/Reset	00
1	Enable Set/Reset	00
2	Color Compare	00
3	Data Rotate	00
4	Read Map Select	00
5	Mode	02
6	Miscellaneous	05
7	Color Don't Care	0F
8	Bit Mask	FF

Якщо виводити інформацію на екран безпосередньо через відеопам'ять, то необхідно вміти визначати біти, що відповідають пікселю з координатами (x, y).

Оскільки в одному рядку є 640 пікселів і на кожний піксель відводиться 1 біт з кожного плану, то всі пікселі одного рядка займають 640 : 8 = 80 байтів, а x пікселів неповного рядка займають [x/8] байтів, тобто зміщення для пікселя з координатами (x, y) дорівнює 80 * y + [x/8].

Номер біта обчислюємо за формулами $s = 7 - x \bmod 8$ або $s = x$ and 7. Для ідентифікації позиції пікселя всередині байта часто використовують не номер біта, а бітову маску – байт, в якому відмінний від нуля тільки той біт, що знаходиться на позиції пікселя. Бітова маска задається виразом 128 shr s.

Зчитування байтами. Щоб прочитати байт із плану, потрібно налаштувати цей план. Це виконує регістр вибору плану читання Read Map Select, що належить графічному контролеру. Усі регістри завантажуються через порти. Але регістри працюють у парі – індексний регістр та регістр даних. Індексний регістр завантажується через порт 3CEH і містить номер функції читання, тобто 4 (табл. 4.5). Регістр даних завантажується через порт 3CFH і містить номер плана N для зчитування (рис. 4.2).

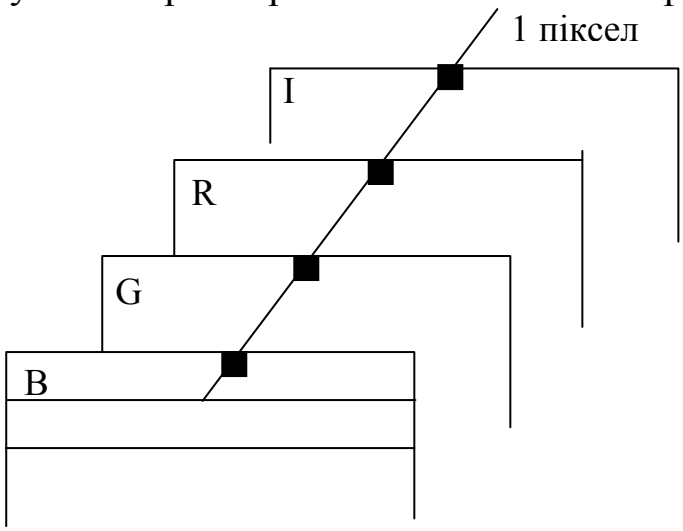

Рис. 4.2. Планарна організація відеопам'яті

Фрагмент програми для зчитування байта з плану N має вигляд

```
port[$3CE] := 4;
port[$3CF] := N;
byte:= mem[$A000:0000];
```

Запис байтами. При записі байтів у плани використовується регістри секвенсера, а саме регістр маски планів MapMask. Індексний регістр секвенсера завантажується через порт 3C4H і містить номер функції запису – число 2. Регістр даних секвенсера заповнюється через порт 3C5H і містить маску планів для запису.

Для утворення маски виставляємо 1 для тих планів, в які хочемо записувати байти, наприклад, якщо Mask = $0101_2 = 5_{16}$, то запис відбувається у нульовий та другий плани.

Отже, фрагмент програми, що записує байт byte в ці плани, такий:

```
port[$3C4]:= 2;
port[$3C5]:= Mask;
mem[A000:0000]:=byte;
```

Процедура налаштування плану N ($0 \le N \le 3$) на читання і запис байтами має вигляд

```
procedure SetPlane(N : integer);
 begin
        port[$3CE]:=4;  port[$3CF]:=N;
        port[$3C4]:=2; port[$3C5]:= 1 shl N;
 end;
```

Наприклад, для зафарбовування екрану в червоний колір потрібно в другий план записати байти з одиниць. Другий план має маску 4 (0100), її записуємо в порт 3C5H, у порт 3C4H заносимо номер функції 2. Отже, маємо такий фрагмент програми:

```
port[$3C4]:= 2;  port[$3C5]:= 4;
for i:= 0 to 28000 do mem[A000:i]:=255; {640:8* 350=28000}
```

Запис та зчитування пікселів. Коли процесор читає байт даних із відеопам'яті, то зразу читаються 4 байти – по одному з кожного плану. При цьому прочитані байти автоматично записуються у спеціальні регістри-затвори (latches-регістри), які не мають прямого доступу.

При операції запису байт, який посилається процесором, накладається на вміст регістрів-затворів за правилами, що визначаються значеннями інших регістрів, і результуючі 4 байти записуються в плани.

Правила, що визначають накладання даних на значення latches-регістрів, при записі визначаються режимом запису. Режим читання задає правила, згідно з якими визначається значення, яке прочитає процесор.

Вибір режиму здійснюємо за допомогою регістра Mode графічного контролера (функція 5), тому в порт 3CEH записуємо 5. Регістр даних (порт 3CFH) заповнюється номером режиму. Номера режиму читання (0, 1) задаються у третьому біті байта Mode, а режими запису (00, 01, 10, 11) – в нульовому та першому бітах цього ж байта (рис. 4.3).

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0		0		

Рис. 4.3. Заповнення регістра Mode

Тому, задаючи режим читання, одночасно визначаємо й нульовий режим запису, і навпаки, задаючи режим запису, задаємо нульовий режим читання.

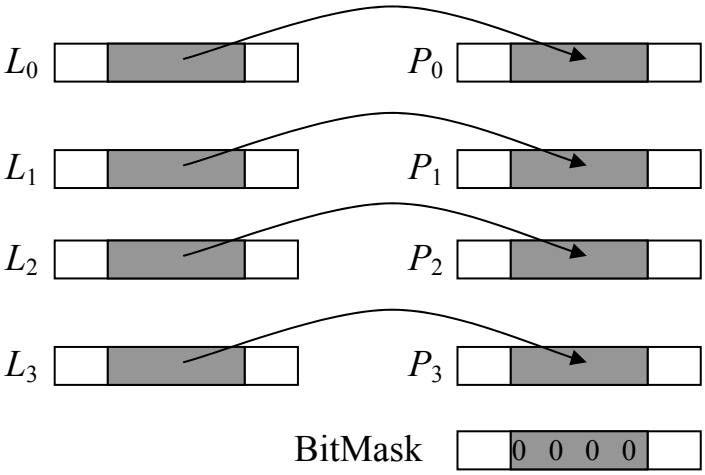

Рис. 4.4. Переміщення вмісту затворів у плани

Розглянемо нульовий режим запису. Це один із найскладніших режимів запису, але використовується він найчастіше. У цьому режимі бере участь група регістрів графічного контролера. Перший із них – регістр BitMask (його номер – 8) дозволяє захистити від зміни певні пікселі. У тих позиціях, де біти в BitMask встановлені в нуль, пікселі не змінюють свого значення, тобто у відповідні біти планів будуть записані біти затворів.

Якщо деякий біт у BitMask встановлений в одиницю, то в дію вступають ще два регістри: Set/Reset (його функція має номер 0) і Enable Set/Reset (функція – 1), який регулює дію першого регістра. У цих регістрах використовуються тільки молодші 4 біти.

Якщо біт у BitMask дорівнює 1, то у всі відповідні позиції планів записується вміст регістра Set/Reset. Але це буде зроблено для тих планів, для яких є дозвіл другого регістра Enable Set/Reset (відповідний біт цього регістра дорівнює 1). У протилежному випадку в план, закритий регістром Enable Set/Reset, буде записаний відповідний біт байта даних (рис. 4.5).

Отже, якщо Enable Set/Reset = 0FH, тоді в позиції планів, що дозволяються регістром BitMask, записується код кольору, заданий у регістрі Set/Reset. Байт, що посилається процесором, жодної ролі не відіграє. Для того, щоб нарисувати один піксель, необхідно завантажити регістр BitMask так, щоб захистити від зміни решту 7 пікселів.

Регістри BitMask і Enable Set/Reset регулюють дію байта даних на відеопам’ять і можуть замінювати окремі біти планів на вміст затворів або регістра Set/Reset.

Отже, процедура запису пікселя color із координатами *x*, *y* має вигляд

```
procedure putpixel_10(x, y, color : integer);
var s, bitmask:integer;
begin
  s:= x and 7;
  bitmask := 128 shr s;    {бітова маска}
  port[$3CE] := 5;   {встановити 0 – режим запису}
  port[$3CF] := 0;
  port[$3CE] := 1; {дозвіл на запис вмісту Set/Reset в усі плани}
  port[$3CF] := $F;
  port[$3CE] := 0; {в регістр Set/Reset завантажити код кольору}
  port[$3CF] := color;
  port[$3CE] := 8; {завантажити регістр bitmask}
  port[$3CF] := bitmask;
  mem[$A000: y*80 + (x shr 3)] := 1; {запис у пам'ять}
        {відновити регістри до стандартного значення}
  port[$3CE] := 1;  port[$3CF] := 0;
  port[$3CE] := 8;  port[$3CF] := $FF;
  port[$3CE] := 5;  port[$3CF] := 10B;
end;{putpixel_10}
```

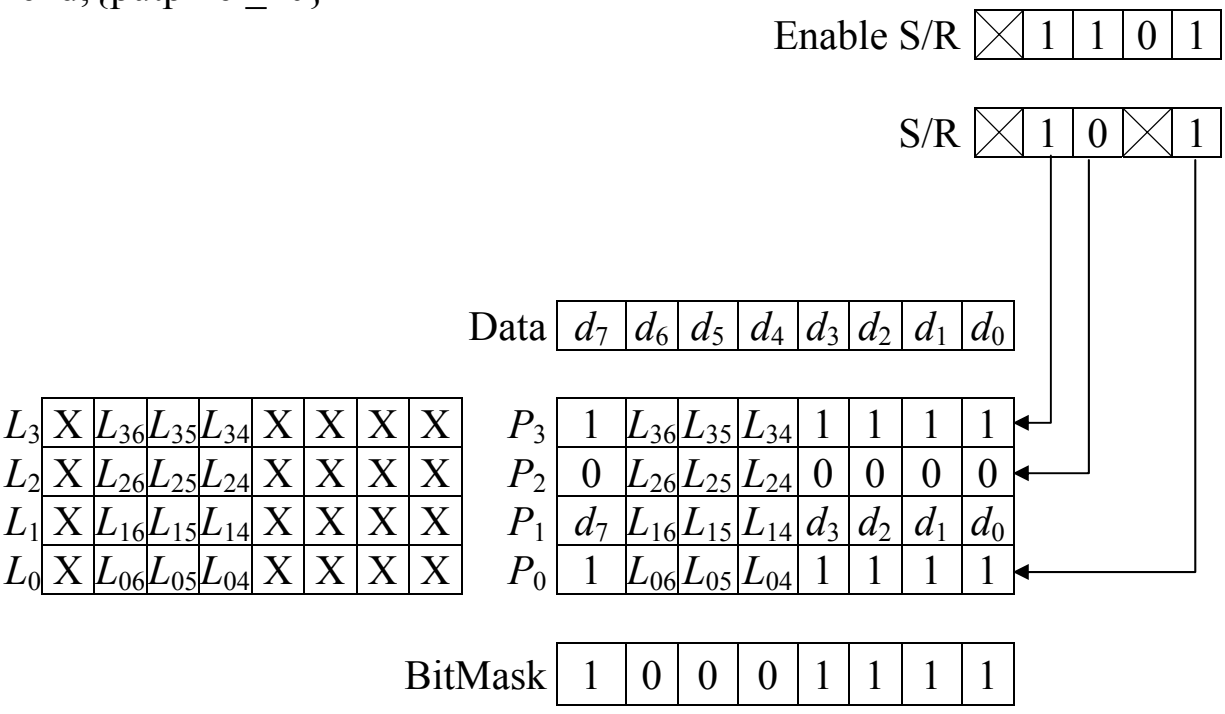

Рис. 4.5. Побітне заповнення планів

При читанні процесором байта даних із відеопам'яті, відповідні байти планів запам'ятовуються в регістрах-затворах (регістри знаходяться на платі відеоадаптера і програмісту недоступні). Якщо потім вико-

нується цикл запису, то над даними, що знаходяться в цих регістрах, і над даними, які надходять від процесора, можуть виконуватися такі логічні операції (без участі процесора):

- запис даних процесора у відеопам'ять без зміни;
- циклічний зсув байта даних, що записується процесором у пам'ять;
- виконання логічних операцій AND, OR, XOR над даними, що записуються процесором у відеопам'ять і величинами регістрів-затворів.

У відеопам'ять записується результат виконання цих операцій. Цими функціями керує регістр Data Rotate (номер 3). У бітах $D_0 – D_2$ регістра даних задаємо, на скільки розрядів потрібно зсунути байт (зсув тільки вправо і є циклічним). Біти D_3, D_4 використовуються для задання логічних операцій

- 00 – запис немодифікованих даних;
- 01 – виконання над даними логічної операції AND, 10 – OR, 11 – XOR.

У режимі зчитування пікселя Read1 в дію вступає регістр Color Compare (порівняння кольору) і VGA порівнює кожну точку в адресованому байті зі значенням, що знаходиться в цьому регістрі. Якщо є збіг цих значень, то в байті ознак у цьому розряді виставляється 1, інакше 0. Якщо тепер циклічно завантажувати регістр Color Compare різними кольорами, то, аналізуючи байт ознак, можна визначити колір пікселя.

Контрольні питання та завдання

1. Назвіть основні пристрої відеоадаптера та їх основні параметри.
2. Дайте класифікацію відеоадаптерів та назвіть їх характеристики.
3. Яке призначення мають графічні процесори? Які задачі вони розв'язують?
4. Охарактеризуйте інтерфейс відеокарти.
5. Опишіть поняття роздільної здатності бітової глибини.
6. Що таке відеосторінки, яке їх призначення?
7. Які ви знаєте стандартні відеорежими, як їх установити?
8. Як визначити обсяг відеопам'яті в різних графічних режимах?
9. Наведіть приклади VESA-режимів. Опишіть процедуру їх установки.
10. Знайдіть зміщення байта, що містить дані про піксель з кординатами (x, y) у режимі 16 кольорів?
11. Як здійснюється вибір режиму запису та зчитування пікелів?
12. Яку роль відіграють регістри-затвори?
13. Назвіть регістри графічного контролера та опишіть їх призначення.
14. Яка будова внутрішньої пам'яті відеоадаптера?

Вправи і задачі для самостійного виконання

1. Записати процедуру зчитування поточного відеорежиму.
2. Графічний режим 640 × 480 × 256. Яка відносна (від початку відеопам'яті) адреса атрибута пікселя з координатами (x,y) на екрані?
3. Графічний режим 640 × 480 × 16. Задана відносна адреса відеопам'яті 2AFH. Для яких точок (x, y) записані атрибути за цією адресою? Скільки бітів містить атрибут пікселя? Який його формат?
4. Графічний режим True Color. Точка на екрані має координати x = 50, y = 20. Яка відносна адреса байта, в який буде записано значення R, G, B для цієї точки?
5. Нехай задана система з монітором 8 × 10 дюймів на якому можна отримати зображення 100 dpi. Якщо пам'ять складається з однобайтових слів, вихідний адрес буфера кадру 0000, а кожному пікселю відповідає один байт пам'яті, то яким буде адрес буфера кадру для пікселя з екранними координатами (x, y)?
6. Скласти програму, яка переміщуватиме прямокутний блок пікселів з одного місця екрана в інше. Режим 13H. Функціями BIOS, BGI не користуватися.
7. Записати процедуру putpixel_10( ), використовуючи функції відеоBIOS.
8. Записати процедуру getpixel_10( ).
9. Написати підпрограму для зчитування поточного VESA-режиму.
10. Роздільна здатність екрана 1280 × 1024, частота кадрів – 72 Гц. Знайти швидкодію пам'яті буфера кадру, тобто визначити скільки часу буде займати зчитування коду кольору одного пікселя?
11. Як довго буде завантажуватися буфер кадру в режимі True Color/ Hight Color з роздільною здатністю 1280×1024, якщо за одну секунду може передаватися 10^7 бітів? До скількох пікселів за секунду може мати доступ контролер дисплея такої системи?
12. Знайти час доступу до одного пікселя в растровій системі з роздільною здатністю 640 × 480 і частотою оновлення 80 кадрів за секунду.