rozdział trzeci

OBSŁUGA PLAYFIELDÓW

Obraz generowany przez Amigę składa się z dwóch podstawowych części – playfieldów, które czasami zwane są tłami, i sprite’ów – obiektów graficznych, które łatwo wprawić w ruch. Ten rozdział opisuje w jaki sposób używać rejestrów sprzętowych do tworzenia i obsługi playfieldów. Rozdział rozpoczyna się od krótkiego przeglądu właściwości playfieldu i obejmuje następujące zagadnienia:

- tworzenie pojedynczego, podstawowego playfieldu, który ma ten sam rozmiar co wyświetlany na ekranie obraz. Ta sekcja obejmuje fundamentalne zasady, które są związane z tworzeniem dowolnego playfieldu.
- tworzenie podwójnego playfieldu, w przypadku którego jeden playfield jest nadrzędnym nad drugim. Ta procedura nieznacznie się różni od tej, dzięki której tworzymy pojedynczy playfield.
- tworzenie playfieldów o dowolnych rozmiarach i wyświetlających jedynie część większego playfieldu.
- poruszanie playfieldów przez przesuwanie ich w pionie i poziomie.
- zaawansowane techniki pozwalające używać playfieldów w specjalnych sytuacjach.

Informacje o ruchomych obiektach sprite zawarte są w rozdziale czwartym o obsłudze sprite’ów. Amiga potrafi także wprawiać w ruch fragmenty playfieldów. Do ich poruszania służy technika zwana animacją playfieldu, która jest opisana w rozdziale szóstym opisującym Blitter.

Informacje o różnicach w obsłudze playfieldów wprowadzonych przez Enhanced Chip Set (ECS), takich jak tryb SuperHires i inne zawarte są w rozdziale „Dodatek C, Enhanced Chip Set”.

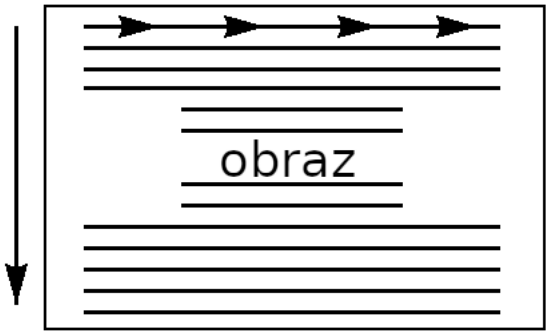
O Amigowych playfieldach

Playfield tworzy podstawowy fundament obrazu wyświetlanego przez Amigę i określa jego podstawowe cechy. Aby utworzyć playfield, trzeba zaprogramować odpowiednie rejestry sprzętowe układów specjalizowanych podstawowymi parametrami obrazu jaki chcesz uzyskać. Definiowanie playfieldu obejmuje: wybór liczby kolorów, ustawienie tablicy kolorów, określenie bitplanów oraz wybór rozdzielczości i trybu wyświetlania.

Aby zrozumieć jak działają Amigowe playfieldy trzeba najpierw poznać jak Amiga generuje obraz.

JAK AMIGA GENERUJE OBRAZ

Amiga tworzy obraz jaki widzimy na ekranie przy użyciu techniki rastrowej. Obraz ten składa się z serii poziomych linii, które są wyświetlane kolejno jedna po drugiej od góry do dołu ekranu. Każda taka linia składa się z wielu malutkich punkcików nazywanych pikselami (ang. *pixel*). To, co widzimy na ekranie, składa się w rzecywistości z jednego lub wiecej bitplanów – tablic w pamięci komutera – zawierających jedynki i zera. Kombinacja tych jedynek i zer określi kolor danego punktu na ekranie.



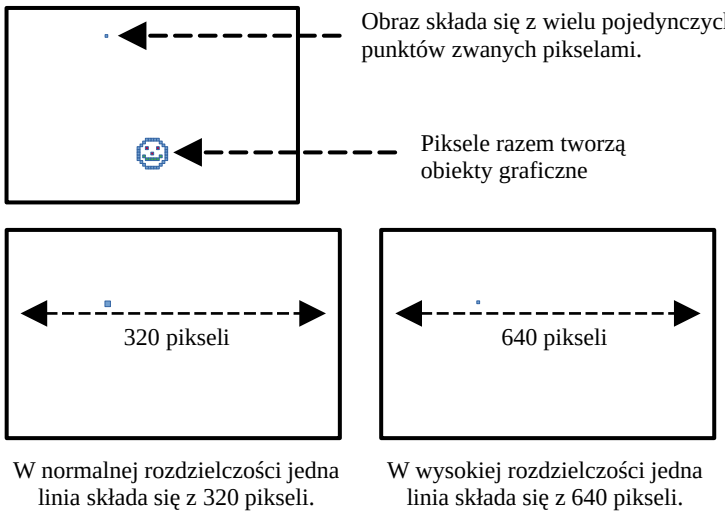
Każda linia obrazuje jedno przejście działa elektronowego które „maluje” obraz.

Promień wizyjny tworzy każdą linię przechodząc od lewej do prawej. Cały obraz to seria takich linii rysowanych od góry ekranu do dołu.

Rys. 3-1: Jak tworzy się obraz wideo

Działko elektronowe tworzy 262 linie obrazu od góry ekranu do jego dołu ale tylko 200 z nich jest widocznych na ekranie w systemie NTSC. W systemie PAL mamy 312 linii, z których 256 jest widocznych. Każdy pełny zestaw linii (262 dla NTSC i 312 dla PAL) jest nazywany klatką (ang. *display field*). Czas, jaki jest potrzebny do stworzenia pełnej klatki obrazu wynosi 1/60 sekundy dla systemu NTSC lub 1/50 sekundy dla obrazu w systemie PAL. Po zakończeniu rysowania jednej klatki obrazu działko elektronowe wraca z dolnego prawego rogu ekranu do wyjściowej pozycji w górnym lewym rogu ekranu aby rozpocząć rysowanie kolejnej klatki obrazu.

Wyświetlany obszar można zdefiniować jako dwuwymiarową siatkę pikseli. Pikesl to pojedynczy, najmniejszy element wyświetlanego obrazu. Obrazki poniżej prezentują jak wygląda piksel i w jaki sposób piksele tworzą obraz.



Rys. 3-2: Co to jest piksel?

Amiga umożliwia wybór zarówno pionowej jak i poziomej rozdzielczości obrazu. Poziomą rozdzielzość można ustawić na niską (normalną) lub wysoką. Rozdzielczosć pionowa może być ustawiona bez przeplotu i z przeplotem (ang. *interlace*).

- W liskiej rozdzielczości standardowy playfield ma szerokość 320 pikseli.
- Wysoka rozdzielczość umożliwia wyświetlenie 640 punktów na tej samej fizycznej szerokości obrazu.
- W trybie bez przeplotu, standardowy playfield NTSC ma wysokość 200 linii. Standardowa wysokość obrazu PAL wynosi 256 linii.
- Trby z przeplotem oferuje 400 linii w NTSC i 512 linii w PAL na tej samej fizycznej wysokości obrazu.

Pozyższe tryby mogą być dowolnie łączone ze sobą. Dla przykładu możemy mieć obraz wysokiej rozdzielczości z przeplotem.

Zauważ, że wymiary określone w poprzednim paragrafie jako standardowe to wartości nominalne i stanową wartości, które powinno się zwykle używać. W rzeczywistości możesz wyświetlać większe playfieldy, maksymalne ich wielkości są podane w sekcji „Bitplany i Playfieldy wszystkich rozmiarów”. Również warto wspomnieć, że wielkość playfieldu w pamięci jest często większa niż playfield wyśiwetlany na ekranie. Musisz wtedy określić która część większego playfieldu ma być aktualnie pokazana definiując rozmiar tzw. okna wyświetlania (ang. *display window*).

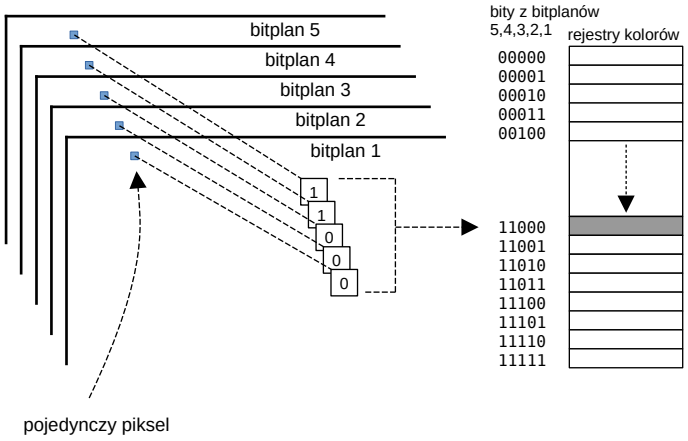
Playfield wyższy niż ekran może być płynnie przesuwany w górę i w dół. Playfield szerszy niż obraz może być płynnie przesuwany w poziomie od prawej do lewej i od lewej do prawej. Przesuwanie jest opisane w sekcji „Przesuwanie (scrollowanie) playfieldów”.

System graficzny Amiga umożliwia użycie do 32 różnych kolorów na jednym playfieldzie przy użyciu normalnych metod wyświetlania. Możesz ustawić kolor każdego punkntu na ekranie ustawiając poszczególne bity składające się na pojedynczy piksel. Obraz tworzony w ten sposób nazywa się obrazem bitmapowym.

Przykładowo, jeśli mówily o obrazni dwukolorowym, kolor każdego piksela jest określany przez to czy bit dla danego punktu jest ustawiony czy nie. Jeśli wartość bitu wynosi 0, punkt jest w jednym kolorze, dla bitu o wartości 1 kolor jest inny niż poprzednio. Czterokolorowy obraz uzyskuje się dzięki dwum bitplanom w pamięci. Kiedy obraz jest wyświetlany, bitplany nakładają się na siebie. Oznacza to, że każdy punkt jest teraz opisany przez dwa bity. Można w ten sposób łączyć do pięciu bitplanów. Obrazy uzyskane dzięki trzem, czterem i pięciu bitplanom pozwalają na wyświetlenie odpowiednio ośmiu, szesnastu i trzydziestu dwóch kolorów.

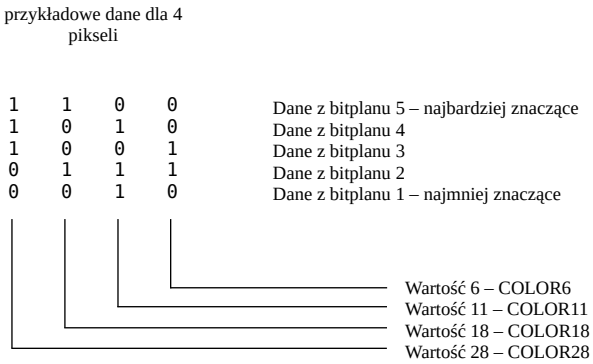
Kolor każdego punktu jest określony dzięki binarnej kombinacji bitów. W chwili kiedy system łączy bitplany obrazu, kombinacja bitów każdego punktu odpowiada właściwemu rejestrowi koloru. Ta metoda kolorowania punktów nosi nazwę *color indirection*. Amiga posiada 32 rejestry kolorów, każdy z nich składa się z bitów określających zdefiniowany wcześniej kolor (z palety 4096 możliwych kolorów).

Rys. 3-3 ukazuje w jaki sposób złączenie do pięciu bitplanów składa się na numer, który wskazuje odpowiedni rejestr koloru, który będzie użyty do pokolorowania danego punktu.



Rys. 3-3: Jak bitplany określają kolor

Bity z bitplanów o wyższym numerze mają wyższe znaczenie w liczbie dwójkowej. Jak pokazano na obrazie 3-4, wartość każdego piksela z najwyżej numerowanego bitplanu odpowiada pierwszyemu bitowi od lewej strony. Wartość z bitplanu o numerze o jeden niższym określa kolejny bit idąc od lewej do prawej, itp.



Rys. 3-4: Znaczenie danych z bitplanów przy określaniu kolorów

Istnieje również możliwość zdefiniowania dwóch oddzielnych playfieldów, każdy składający się z maksymalnie trzech bitplanów. Playfieldy te mogą używać dwóch oddzielnych zestawów ośmiu kolorów. Nazywamy to trybem podwójnego playfieldu (ang. *dual playfield mode*).

Tworzenie podstawowego playfieldu

Ta sekcja opisuje w jaki sposób ustawić rejestry sprzętowe aby uzyskać pojedynczy podstawowy playfield o takim samym rozmiarze jak obraz wideo. „Taki sam rozmiar” oznacza, że playfield będzie miał ten sam rozmiar co okno wyświetlania. Spowoduje to pozostawienie pewnego marginesu pomiędzy playfieldem a krawędziami ekranu. Playfield zazwyczaj nie rozciąga się do samych krawędzi ekranu.

Aby stworzyć playfield musisz zdefiniować następujące cechy:

- Wysokość i szerokość playfieldu oraz rozmiar okna wyświetlania (czyli jak jaka część playfieldu pojawi się na ekranie).
- Kolor każdego z puntów playfieldów
- Rozdzielczość poziomą.

- Rozdzielczość pionową, przeplot.
- Pobór danych (ang. *data fetch*) i modulo, które mówią systemowi jak dużo danych składa się na pojedynczą linię obrazu i jak te dane przesłać z pamięci na ekran.

Dodatkowo, trzeba zarezerwować pamięć do zapisania danych playfieldu, ustawić wskaźniki i poinformować system gdzie te dane znaleźć i (opcjonalnie) napisać procedurę dla Coppera, która obsłuży cykliczne wyświetlanie playfieldu.

WYSOKOŚĆ I SZEROKOŚĆ PLAYFIELDU

Aby stworzyć playfield, który ma taki sam rozmiar jak ekran możesz użyć szerokości zarówno 320 punktów jak i 640 punktów, zależnie od rozdzielczości jaką wybrałeś. Wysokość obrazu może być 200 lub 400 linii dla odbiorników pracujących z trybie NTSC lub 256 albo 512 linii dla trybu PAL, zależnie czy wybrałęś tryb bez przeplotu czy z przeplotem.

BITPLANY I KOLORY

Kolor playfieldu określa się następująco:

1. Zdecyduj ile kolorów chesz użyć i jaki kolor maja poszczególne punkt obrazu.
2. Ustaw rejestry kolorów odpowiednimi wartościami
3. Zarezerwuj pamięć na taką ilość bitplanów jaką potrzebujesz i ustaw wskaźniki do każdego bitplanu.
4. Napisz instrukcję do ustawienia poszczególnych bitów w bitplanach, żeby uzyskać właściwy kolor.

Tablica 3-1 prezentuje zależność ilości bitplanów niezbędnych do uzyskania założonej liczby kolorów.

Liczba kolorów	**Ilość bitplanów**
1-2	1
3-4	2
5-8	3
9-16	4
17-32	5

Tabela 3-1: Kolory dla pojedynczego playfieldu

Tablica kolorów

Tablica kolorów zawiera 32 rejestry, do których możesz załadować dowolny kolor. Poniższa tabela zawiera skondensowany podgląd tablicy kolorów:

Nazwa rejestru	Zawartość	Znaczenie
COLOR00	12 bitów	Zdefiniowany przez użytkownika kolor tła i obramowania wokół ekranu
COLOR01	12 bitów	Zdefiniowany przez użytkownika kolor o numerze 1
COLOR02	12 bitów	Zdefiniowany przez użytkownika kolor o numerze 2
...	...	...
COLOR31	12 bitów	Zdefiniowany przez użytkownika kolor o numerze 31

Tabela 3-2: Fragmenty tablicy kolorów

COLOR00 jest zarezerowowany dla koloru tła. Kolor ten jest widoczny w każdej części wyświetlanego obrazu, w której nie ma widocznych innych obiektów. Tło jest również wyświetlane poza zdefiniowanym oknem wyświetlania jako obramowanie.

> *Genlocki i kolor tła*. Jeśli używasz dodatkowego modułu genlocka jako wejście obrazu z kamery wideo, magnetowidu czy odtwarzacza dysków laserowych, tło będzie zastąpione sygnałem wideo z tych urządzeń.

12 bitów definiujących kolor pozwala na zapisanie w każdym z 32 rejestrów jednego z 4096 możliwych kolorów w sposób pokazany w tabeli 3-3.

Bity	
Bity 15-12	nieużywane
Bity 11-8	czerwony
Bity 7-4	zielony
Bity 3-0	niebieski

Tabela 3-3: Zawartość rejestru kolorów

Tabela 3-4 pokazuje kilka przykładowych wartości kolorów, które można zapisać w rejestrach kolorów. Na końcu rozdziału można znaleźć obszerniejszą listę.

Zawartość rejestru koloru	Kolor wyjściowy
$FFF	biały
$6FE	błękitny
$DB9	Beżowy (ang. *tan*)
$000	czarny

Tabela 3-4: Przykładowe wartości rejestru koloru

Pnożej można zobaczyć przykładowe instrukcje do ustawienia rejestrów kolorów:

```
LEA CUSTOM,a0            ; Ustaw adres bazowy dla układów specjalizowanych
MOVE.W #$FFF,COLOR00(a0) ; załaduj kolor biały do rejestru 0
MOVE.W #$6FE,COLOR01(a0) ; załaduj kolor błękitny do rejestru 1
```

> *Rejestry kolorów są tylko do zapisu*. Tylko patrząc na rezultat na ekranie jesteś w stanie stwierdzić, jak są ustawione. Ważne więc, żeby twój program ustawiał rejestry w taki sposób, żebyś zawsze wiedział jaki kolor jest w jakim rejestrze.

Wybór ilości bitplanów

Po decyzji o ilości kolorów i co się z tym wiąże jak wiele bitplanów będzie potrzebne do uzyskania ich wszystkich teraz musisz poinformować Amigę o ilości bitplanów.

Ilość bitplanów ustawiasz zapisując odpowiednią wartość w rejestrze BPLCON0 (rejestr numer 0 kontrolujący bitplany, ang. *Bitplane Control Register 0*). Odpowiednią wartość ustawia się korzystając z bitów o numerach 14, 13 i 12, nazywanych odpowiednio BPU2, BPU1 i BPU0 (użyte bitplany, ang. *BitPlanes Used*). Tabela 3-5 pokazuje jakie wartości przyjmują te bity dla uzyskania odpowiedniej ilości bitplanów.

Wartość	Ilość bitplanów	Nazwy bitplanów
000	Brak *	
001	1	PLANE 1
010	2	PLANES 1 i 2
011	3	PLANES 1-3
100	4	PLANES 1-4
101	5	PLANES 1-5
110	6	PLANES 1-6 **
111		Wartość nie używana

Tabela 3-5: Ustawienie ilości bitplanów

* Wyświetla tylko kolor tła, nie ma żadnych bitplanów

** Szósty bitplan jest używany tylko w trybie podwójnego playfieldu oraz w trybie hold-and-modify (opisanym w sekcji „Techniki zaawansowane”)

Rejestr BPLCON0. Bity w rejestrze BPLCON0 nie mogą być ustawione niezależnie. Aby zmienić jeden bit musisz ustawić pozostałe na ich obecne wartości.

Poniższy przykład pokazuje w jaki sposób poinformować Amigę o chęci użycia dwóch bitplanów niskiej rozdzielczości.

```
MOVE.W #$2200,BPLCON0+CUSTOM  ; zapisz wartość do BPLCON0
```

Ponieważ rejestr BPLCON0 jest używany do ustawiania innych cech wyświetlanego obrazu i nie jest możliwe ustawienie poszczególnych jego bitów niezależnie, przykład powyższy ustawia również inne parametry (opiszemy je dalej w tym rozdziale).

- Tryb hold-and-modify jest wyłączony
- Tryb pojedynczego playfieldu jest włączony
- Tryb kolorowego wyjścia composite video jest włączony (dotyczy Amigi 1000)
- Dźwięk genlocka jest wyłączony
- Pióro świetlne jest wyłączone
- Tryb z przeplotem jest wyłączony
- Zewnętrzna resynchronizacja jest wyłączona (genlock)

WYBÓR ROZDZIELCZOŚCI PIONOWEJ I POZIOMEJ

Telewizory z czasów Amigi były najlepiej przystosowane do pracy w niskiej rozdzielczości. Tryb niskiej rozdzielczości umożliwia wyświetlenie 320 punktów w jednej linii. Wysoko rozdzielcze monitory monochromatyczne lub kolorowe z tamtych czasów były w stanie wyświetlić 640 punktów w każdej linii. Jeśli zdefiniujesz obiekt w niskiej rozdzielczości a potem wyświetlisz go w wysokiej, obiekt będzie miał fizycznie połowę szerokości w porównaniu do niskiej rozdzielczości.

Aby włączyć wysoką rozdzielczość, trzeba ustawić bit 15, zwany HIRES, w rejestrze BPLCON0.

Wysoka rozdzielczość – zapisz 1 w bicie 15.
Niska rozdzielczość – zapisz 0 w bicie 15.

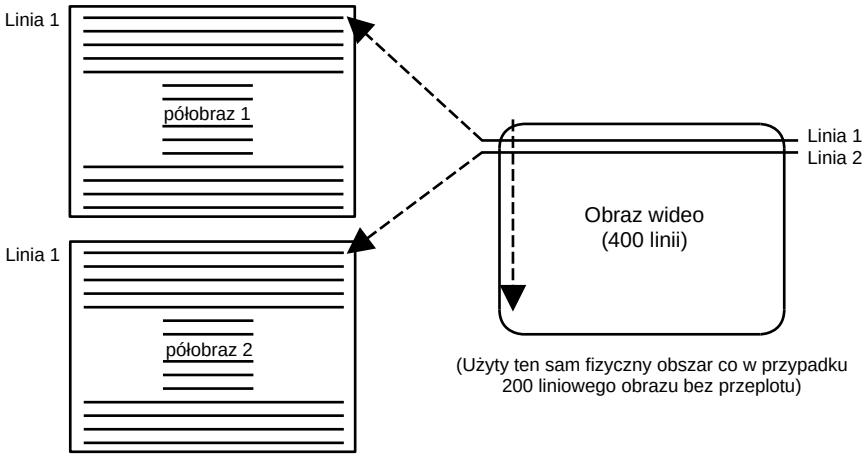
Pamiętaj, że w wysokiej rozdzielczości możesz użyć tylko 4 bitplanów a co się z tym wiąże, możesz użyć maksymalnie 16 kolorów.

Tryb z przeplotem wymaga dwukrotnie większej ilości danych wyświetlanych na takiej samej powierzchni ekranu. Uzyskuje się to podawjając ilość linii widocznych na ekranie. Pniższa tabelka pokazuje ilość linii wymaganą do zapełnienia standardowego ekranu bez tzw. overscanu.

	NTSC	**PAL**
Bez przeplotu	200	256
Z przeplotem	400	512

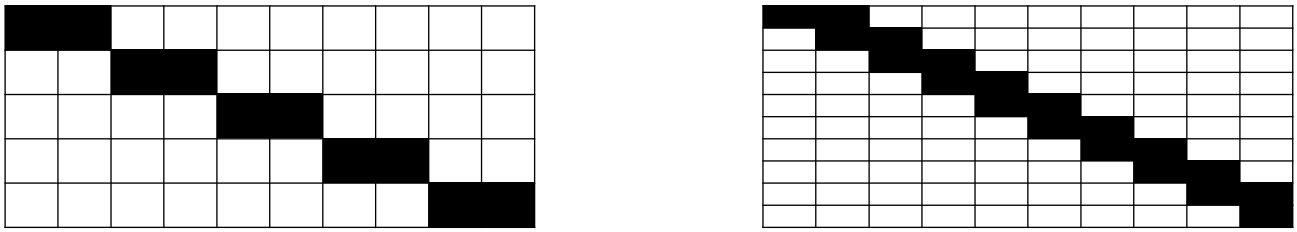
Tabela 3-6: Ilość linii w standardowym playfieldzie

W trybie z przeplotem układ wyświetlający obraz przesuwa co drugi półobraz o połowę linii.



Rys. 3-5: Przeplot

Mimo iż tryb z przeplotem wymaga niewielkiej ilości dodatkowej pracy przy ustawianiu rejestrów (jak zobaczysz w dalszej części tej sekcji), to zapewnia swego rodzaju polepszenie jakości, które można zaobserwować w przypadku niektórych efektów graficznych. Rozważmy ukośną linię na rysunku 3-6, która porównujemy w trybach bez przeplotu i z przeplotem. Przeplot pozwala zminimalizwować „schodki” jakie obserwujemy na ukośnych krawędziach.



bez przeplotu z przeplotem

Rys. 3-6: Wpływ przeplotu na krawędzie

Kiedy używasz specjalnego kanału DMA Blittera do wysowania linii lub wielokątow na playfieldzie z przeplotem, playfield ten jest traktowany jako jeden obraz a nie jako dwa półobrazy. Mimo to wciąż widać efekt wygładzania krawędzi w trybie z przeplotem.

Aby ustawić tryb z przeplotem lub bez włączasz lub wyłączasz bit 2, zwany LACE, w rejestrze BPLCON0:

Tryb z przeplotem – zapisz bit 2 wartością 1.
Tryb bez przeplotu – zapisz bit 2 wartością 0.

Jak to zostało opisane w sekcji „Wybór liczby bitplanów”. Bity w BPLCON0 nie są ustawiane niezależnie.

Poniższy przykład pokazuje jak ustawić wysoką rozdzielczość z przeplotem.

```
MOVE.W #$A204,BPLCON0+CUSTOM ; ustaw rejestr BPLCON0
```

Przykład powyżej ustawia także poniższe parametry:

- Tryb wysokiej rozdzielczości włączony.
- Dwa bitplany.
- Hold-and-modify wyłączony.
- Pojenynczy playfield włączony.
- Colorowe wyjście composite video włączone.
- Audio dla genlocka wyłączone.
- Obsługa pióra świetlnego wyłączona.
- Włączony tryb z przeplotem.
- Zewnętrzna synchronizacja wyłączona.

Od wybranej rozdzielczości zależy ilość pamięci jaką powinieneś zarezerwować dla każdego z bitplanów. Ustawienie wysokiej rozdzielczości czy też trybu z przeplotem zwiększa zapotrzebowanie na pamięć gdyż wymaga większych bitplanów.

REZERWACJA PAMIĘCI DLA BITPLANÓW

Po ustawieniu ilości bitplanów i określeniu rozdzielczości powinieneś zarezerwować pamięć. Bitplan to ciągły obszar pamięci, w którym przechowywane będą dane tego bitplanu. Program działający pod systemem operacyjnym Amigi powinien użyć funkcji systemowej takiej jak AllocMem() do zarezerwowania obszaru pamięci. Funkcja ta wyłącza fragment pamięci z listy pamięci dostępnej i udostępnia go programowi.

Do rezerwacji pamięci przeznaczonej na dane bitplanu warto jednak używać funkcji specjalnie do tego przeznaczonych. Taką funkcją jest AllocRaster() z biblioteki graphics.library. Funkcja ta odpowiednio przygotuje obszar pamięci do pracy z układem generującym grafikę.

W przypadku gdy pomija się system operacyjny, wystarczy zarezerwować obszar pamięci do przechowywania bitplanów. Następnie trzeba ustawić rejestry wskaźników bitplanów (BPLxPTH i BPLxPTL) aby wskazywały na początek każdego z bitplanów umieszczonych w zarezerwowanej pamięci.Adres początkowy bitplanu jest wielkości słowa (16 bitów) i zawiera pozycję górnego lewego rogu bitplanu.

Tabele 3-7 i 3-8 pokazują ile pamięci zajmują proste bitplany w systemach NTSC oraz PAL. Musisz kontrolować rozdzielczość i ilość kolorów zależnie od ilości posiadanej pamięci.

Rozmiar obrazka	**Tryby**	**Ilość bajtów dla każdego bitplanu**
320x200	Niska rozdzielczość, bez przeplotu	8000
320x400	Niska rozdzielczość, z przeplotem	16000
640x200	Wysoka rozdzielczosć, bez przeplotu	16000
640x400	Wysoka rozdzielczość, z przeplotem	32000

Tabela 3-7: Wymagania pamięciowe playfieldu w NTSC

Pamiętaj, że ilość bajtów, jaką musisz zarezerwować dla bitplanu musi być parzysta.

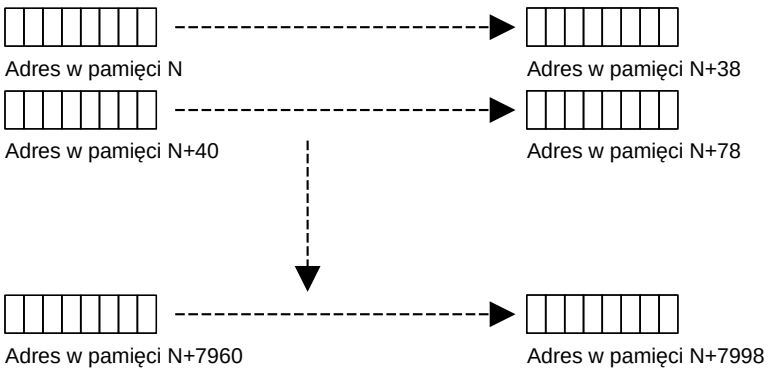
Rozmiar obrazka	Tryby	Ilość bajtów dla każdego bitplanu
320x256	Niska rozdzielczość, bez przeplotu	8192
320x512	Niska rozdzielczość, z przeplotem	16384
640x256	Wysoka rozdzielczosć, bez przeplotu	16384
640x512	Wysoka rozdzielczość, z przeplotem	32768

Tabela 3-8: Wymagania pamięciowe playfieldu w PAL

Przykład rozmaru bitplanu dla NTSC

Jako przykładu użyjemy standardowego obrazu w trybie NTSC o niskiej rozdzielczości, bez przeplotu. Taki obraz ma 320 pikseli w linii. Piksel ja pojedynczym bitplanie to jeden bit, łątwo więc zauważyć, że 8 kolejnych pikseli potrzebuje bajta do zachowania stanu. Każda z takich linii potrzebuje więc 40 bajtów aby zapisać dane wszystkich pikseli (320 punktów / 8 bitów = 40 bajtów). Ponieważ linii w trybie NTSC bez przeplotu jest 200, potrzeba 8000 bajtów (200 linii * 40 bajtów = 8000 bajtów) w pamięci aby przechować kompletne dane pojedynczego bitplanu. Dokładnie tyle ile podane jest w tabelce 3-7.

Playfield skłądający się z dwóch bitplanów w niskiej rozdzielczości bez przeplotu potrzebuje 16000 bajtów pamięci, czyli dwukrotnie więcej. Pamięć przydzielona każdemu bitplanowi musi być ciągła (nie może mieć „przerw” zajętych przez inne dane) tak więc dla dwóch bitplanów potrzeba dwóch ciągłych obaszarów pamięci po 8000 bajtów każdy. Poszczególne bitplany mogą być położone w osobnych obszarach pamięci, ważne, że każdy z nich „jest ciągły”. Obrazek 3-7 pokazuje organizację pamięci dla pojedynczego bitplanu zawierającego 200 linii po 40 bajtów każda, gdzie każdy piksel jest reprezentowany przez 1 bit w tym obszarze pamięci.



Rys. 3-7: Organizacja pamięci prostego bitplanu

Dostęp do bitplanów w pamięci jest realizowany przez dwa rejestry adresowe: BPLxPTH i BPLxPTL. Każdy z możliwych bitplanów posiada własną parę tych rejestrów, literę „x” zastępuje się odpowiednim numerem bitplanu. Dla przykładu, aby ustawić adres startu bitplanu numer 1 korzystamy z rejestrów BPL1PTH i BPL1PTL. Każda para przechowuje 19-bitowy adres. W przypadku programowania w języku maszynowym procesora 68000 można traktować te rejestry

jako jeden rejestr 32-bitowy. Wystarczy więc jedna instrukcja assemblera aby zapisać adres bitplanu do obu rejestrów jednocześnie, jako operand docelowy wystarczy podać rejestr „PTH”.

Przykład poniżej pokazuje jak ustawia się wskaźniki bitplanów. Zakładamy, że mamy dwa bitplany, jeden pod adresem $21000 a drugi $25000. Procesor zapełnia rejestr BPL1PT adresem $21000 a do rejestru BPL2PT zapisuje adres $25000. Warto jednak zauważyć, że najczęściej używa się Coppera do tych operacji.

```
;
; Ponieważ rejestry wskazujące na bitplany są widziane przez 680x0 jako
; 32-bitowe, możemy użyć instrukcji move o rozmiarze long-word
;
    LEA CUSTOM,a0     ; adres bazowy dla układów specjalizowanych
    MOVE.L $21000,BPL1PTH(a0)   ; wskaźnik pierwszego bitplanu
    MOVE.L $25000,BPL2PTH(a0)   ; wskaźnik drugiego bitplanu
```

Pamiętaj, że wyżej opisane wymagania pamięciowe dotyczą tylko playfieldu. W przypadku korzystania ze sprite’ów, animacji czy dźwięku będziesz również musiał rezerwować pamięć na ich potrzeby. Zapotrzebowanie pamięci dla tych i innych elementów będzie opisane w rozdziałach im poświęconcyh.

USTAWIENIA BITPLANÓW POD KĄTEM KOLORÓW

Po określeniu ilości bitplanów i ustawienia ich wskaźników możesz teraz przejść do konfiguracji odpowiadających im rejestrów kolorów.

Jedno lub dwukolorowe playfieldy

Dla jednokolorowego bitplanu wszystko co musisz zrobić to zapisać 0 w każdym bicie pojedynczego playfieldu. Poniżej znajdziesz przykładowy zestaw instrukcji assemblera realizujący to zadanie. Pokazany kod wypełnia bitplan niskiej rozdzielczości kolorem tła (COLOR00) przez zapisanie obszaru pamięci zerami. Bitplan zaczyna się od adresu $21000 i ma 8000 bajtów długości.

```
    LEA $21000,a0     ; wskaż na bitplan
    MOVE.W #2000,d0   ; licznik – rozmiar bitplanu (2000 długich słów)
LOOP:
    MOVE.L #0,(a0)+   ; zapisz zero i zwiększ rejestr adresowy
    DBF d0,LOOP       ; zmniejsz licznik, sprawdź czy d0 == 0, jeśli nie
                      ; to wykonuj pętlę dalej
```

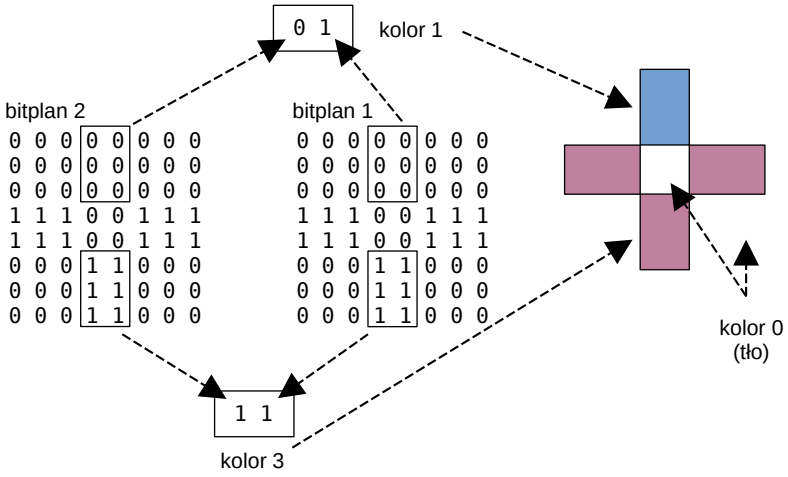
W przypadku dwukolorowego playfieldu używasz wartości bitu 0 w przypadku kiedy chcesz użyć koloru tła (COLOR00) a wartości bitu 1 dla koloru z rejestru pierwszego (COLOR01). Poniższy przykład jest identyczny z poprzednim z tą jednak różnicą, że zamiast 0 zapisujemy pamięć wartościami $FF00FF00. Dzięki temu uzyskamy w rezultacie dwa kolory.

```
    LEA $21000,a0     ; wskaż na bitplan
    MOVE.W #2000,d0   ; licznik – rozmiar bitplanu (2000 długich słów)
LOOP:
    MOVE.L #FF00FF00,(a0)+   ; zapisz zero i zwiększ rejestr adresowy
    DBF d0,LOOP       ; zmniejsz licznik, sprawdź czy d0 == 0, jeśli nie
                      ; to wykonuj pętlę dalej
```

Playfield z trzema i więcej kolorami

Jeśli chcesz użyć trzech lub więcej kolorów, potrzebujesz więcej niż jednego bitplanu. Zadaniem jest więc tak zdefiniować każdy z nich, żeby łącząc je w celu wyświetlenia na ekranie, każdy piksel miał właściwą kombinację bitów. To trochę bardziej skomplikowane niż w przypadku playfielda z pojedynczym bitplanem. Poniżej pokazano przykład dla czterokolorowego playfieldu ale zasada i sposób działania są identyczne dla playfieldów zawierających do 32 kolorów.

Rys. 3-8 pokazuje dwa bitplany tworzące czterokolorowy obraz:



Obras 3-8: Łączenie bitplanów

Ustawiasz odpowiednie punkty bitplanów wartościami „1” lub „0” aby uzyskać właściwy kolor dla każdego z punktów obrazu.

W przypadku pojedynczego playfieldu możesz złączyć do pięciu bitplanów używajac powyższego sposobu. Uzycie tylu bitplanów pozwala na wybór 32 różnych kolorów jakie można nadać poszczególnym pikselom. Tabela wyboru kolorów dla playfieldów umieszczona na końcu tego rozdziału prezentuje kombinacje bitów dla ekranów z czterema i pięcioma bitplanami.

OKREŚLANIE WIELKOŚCI OKNA WYŚWIETLANIA

Kiedy zakończysz definiowanie playfieldu, możesz przejść do określenia wielkości okna wyświetlania, które ogranicza wyświetlany obraz. Modyfikacje rozmiaru okna wyświetlania mają wpływ na cały wyświetlany obszar ekranu, włączając w to ramki i sprite’y, nie tylko playfield. Nie da się wyświetlić żadnego elementu poza zdefiniowanym oknem wyświetlania. Rozmiar obramowania wokół obrazu różnież zależy od rozmiaru tego okna.

Podstawowy playfield opisany w tej sekcji ma ten sam rozmiar co obraz wyświetlany na ekranie a także jak okno wyświetlania. Nie zawsze jednak tak jest. Często okno wyświetlania jest mniejsze niż faktyczny całe pole gry przechowywane w pamięci (raster).

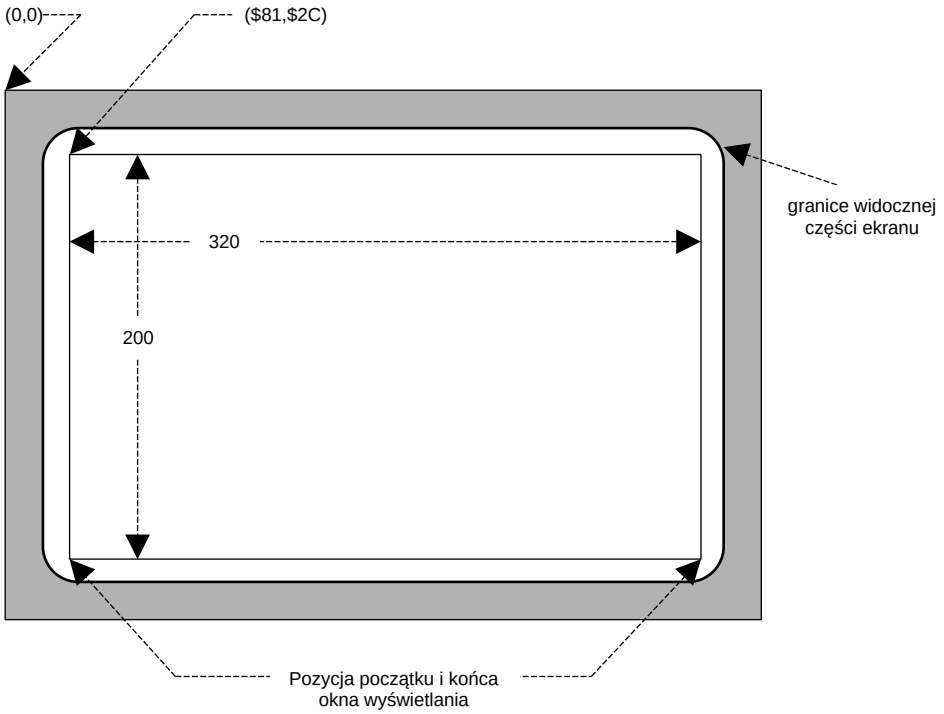
Okno wyświetlania mniejsze niż playfield pozwala pokazać na ekranie pewnien fragment większej planszy. Dzięki temu można uzyskać przesuwanie obrazu na ekranie. Oczywiście można również zdefiniować większe niż playfield okno wyświetlania. Większe playfieldy i okna wyświetlania o różnych rozmiarach są opisane w sekcji „Bitplany i okna wyświetlania o różnych rozmiarach” poniżej.

Rozmiar okna wyświetlania określa się przez zdefiniowania pionowej i poziomej pozycji początku i końca i zapisania tych wartości do odpowiednich rejestrów. Rozdzielczość pionowego początku i końca ma rozmiar jednej linii niskiej rozdzielczości. Rozdzielczość pozioma początku i końca wynosi jeden piksel nieskiej rozdzielczości. Każda pozycja na ekranie obrazuje pozycję pionową i poziomą jakiegoś piksela. Pozycja ta może zostać opisana przez współrzędne x i y tego punktu na ekranie. W dalszej części będziemy używać notacji (x,y) do określenia współrzędnych punktu na ekranie.

Pomimo iż współrzędne zaczynają się od punktu (0,0) znajdującego się w górnym lewym rogu ekranu, pierwszą używaaną pozycją poziomą jest zazwyczaj $81 a pierwszą pozycją pionową jest $2C. Pionowe i poziome pozycje startowe są identyczne zarówno dla systemu NTSC jak i PAL.

Amiga pozwala na zdefiniowanie pozycji startowej powyżej ($81,$2C) ale część obrazu poza tymi współrzędnymi może być niewidoczna na niektórych wyświetlaczach. Różnica pomiędzy absolutną pozycją początkową (0,0) a rzeczywistą pozycją ($81,$2C) wynika z ograniczeń konstrukcyjnych lam kineskopowych używanych w telewizorach i monitorach przed erą wyświetlaczy LCD.

Aby zapobiec zniekształceniom obrazu mogącym wystąpić na skrajach kineskopów działo elektronowe obejmuje swym zasięgiem znacznie większą powierzchnię niż standardowa, jaką posiadają kineskopy. Pozycja ($81,$2C) jest powszechnie uznana za parmetr, który centruje obraz na większosci odbiorników pozostawiając ramkę wielkości 8 punktów niskiej rozdzielczości wokół okna wyświetlania. Rys. 3-9 pokazuje zależność pomiędzy normalnym oknem wyświetlania, widoczną powierzchnią obrazu na ekranie a powierzchnią faktycznie obejmowaną przez działo elektronowe.



Rys. 3-9: Pozycjonowanie obrazu na ekranie

Ustawienie pozycji początkowej okna wyświetlania

Ustawienie punktu początkowego w poziomie wartością $81 a punktu początkowego w pionie wartością $2C powoduje, że ekran na standardowym telewizorze kineskopowym będzie wyśrodkowany. Jeśli ustawimy wysoką rozdzielczość (640 punktów w poziomie) lub tryb z

przeplotem (400 linii dla NTSC i 512 dla PAL) punkt początkowy się nie zmienia. Punkt początkowy zawsze podaje się w relacji do niskiej rozdzielczości bez przeplotu. Inaczej mówiąc, współrzędne punktu początkowego zawsze należy wyznaczać dla niskiej rozdzielczości bez przeploty niezależnie od faktycznej wybranej rozdzielczości pionowej czy poziomej.

Początkową pozycję okna wyświetlania zapisuje się w rejestrze DIWSTRT (ang. *Display Window Start*). Rejestr ten przechowuje zarówno pionową jak i poziomą pozycję. Oznacza się je HSTART i VSTART. Poniższy przykład pokazuje jak ustawić rejestr DIWSTRT dla przykładowego playfieldu. Bity VSTART przyjmują wartość $2C a bity HSTART $81.

```
LEA CUSTOM,a0     ; adres bazowy dla układów specjalizowanych
MOVE.W #$2C81,DIWSTRT(a0)   ; rejestr pozycji początkowej...
```

Ustawienie pozycji końcowej okna wyświetlania

Żeby poprawnie zdefiniować okno wyświetlania musisz także określić pozycję końca, która znajduje się w jego dolnym prawym rogu. Podobnie jak w przypadku pozycji startowej, ustawienie wysokiej rozdzielczości czy tryby z przeplotem nic tu nie zmienia. Pozycja końca okna wyświetlania jest zawsze wyznaczana tak, jakby obraz miał niską rozdzielczość bez przeplotu.

Do przechowywania końcowej pozycji okna wyświetlania służy rejestr DIWSTOP (ang. *Display Window Stop*). Rejestr ten również przechowuje obie współrzędne – poziomą i pionową – nazwane odpowiednio HSTOP i VSTOP.Instrukcje poniżej pokazują w jaki sposób zapisać do rejestru DIWSTOP odpowiednie wartości przy założeniu, że pozycja początkowa znajduje się w punkcie ($81,$2C). Zauważyć należy, że wartość HSTOP to rzeczysista wartość pomniejszona o 256 ($100). Pozycja wartość zapisywana w HSTOP ogranicza sie tylko do prawej połowy ekranu. Standardową wartością jest $1C1 ale w rejestrze zapisuje się $C1. Wartość HSTOP jest taka sama zarówno dla systemu NTSC jak i PAL.

Wartość VSTOP ogranicza się do dolnej połowy ekranu. Ponieważ wartość VSTOP zapisywana jest na 8 bitach a rzeczywista wartość może przekroczyć 255 ($FF), układy odpowiadające za wyświetlanie obrazu stosują pewien trik. Na podstawie MSB (najbardziej znaczący bit, ang. Most Significant Bit) pozycji końcowej wyznaczają kolejny MSB jako jego uzupełnienie. Czyli jeśli bit 8 pozycji VSTOP wynosi „0” to „wirtualny” bit 9 będzie miał wartość „1”. I odwrotnie, jeśli bit 8 pozycji VSTOP jest równy „1” to „wirtualny” bit 9 przyjmuje wartość 0. Standardowo, VSTOP przyjmuje wartość $F4 dla NTSC albo $2C dla PAL.

Standardowa wartość DIWSTRT dla NTSC wynosi $2C81.
Standardowa wartość DIWSTOP dla NTSC wynosi $F4C1.

Standardowa wartość DIWSTRT dla PAL wynosi $2C81.
Standardowa wartość DIWSTOP dla PAL wynosi $2CC1.

Poniższy przykład pokazuje jak ustawić rejestr DIWSTOP dla przykładowego playfieldu wartościami $F4 dla pozycji poziomej i $C1 dla pozycji pionowej.

```
LEA CUSTOM,a0     ; adres bazowy dla układów specjalizowanych
MOVE.W #$F4C1,DIWSTOP(a0)   ; rejestr pozycji końcowej...
```

	Wartości nominalne		Możliwe wartości	
	NTSC	PAL	MIN	MAX
DIWSTRT				
VSTART	$2c	$2c	$00	$FF
HSTART	$81	$81	$00	$FF
DIWSTOP				
VSTOP	$F4	$2C (=$12C)	$80	$7F (=$17F)
HSTOP	$C1	$C1	$00 (=$100)	$FF (=$1FF)

Tabela 3-9: Podsumowanie DIWSTRT i DIWSTOP

Wartości minimum i maksimum zostały zwiększone w nowszej wersji układów specjalizowanych Amigi (ECS). Szczegóły można znaleźć w rozdziale „Dodatek C, Enchanced Chip Set”.

WSKAZANIE SYSTEMOWI JAK POBRAĆ I WYŚWIETLIĆ DANE

Po określeniu rozmiaru i pozycji okna wyświetlania trzeba poinformować system o położeniu obszaru, który będzie wyświetlony na ekranie po pobraniu danych. Aby to zrobić określa się pozycję początku i końca każdej linii i zapisuje się te wartości w odpowiednich rejestrach poboru danych (ang. *data fetch*). Rejestry te mają „rozdzielczość” czterech pikseli (odmiennie do rejestrów okna wyświetlania przyjmujących wartości co do piksela). Każda pozycja przekakuje co cztery punkty. Piksel 0 ma pozycję 0, piksel 4 ma pozycję 1, itd.

Początek poboru danych oraz pozycja początku okna wyświetlania współdziałają razem. Zaleca się aby wartości początkowe poboru danych były określane z rozdzielczością co 16 punktów (8 taktów zegarowych w niskiej rozdzielczości lub 4 takty w wysokiej). Układy specjalizowane potrzebują pewnego czasu od pobrania pierwszych danych zanim dane te trafią na ekran. W rezultacie istnieje różnica pomiędzy wartością początku okna a wartością początku poboru danych wynosząca 4.5 taktów zegara kolorów.

Standardowa wartość DDFSTRT dla niskiej rozdzielczości wynosi $38.
Standardowa wartość DDFSTRT dla wysokiej rozdzielczości wynosi $3C.

Wartości te można wyliczyć przy pomocy następujących wzorów (dla przypomnienia, sprzętowa rozdzielczość wartości początku i końca okna wyświetlania jest dwukrotnością sprzętowej rozdzielczości poboru dancyh):

$81/2 – 8.5 = $38
$81/2 – 4.5 = $3C

Zależność między wartościami początku i końca poboru danych wygląda tak:

DDFSTRT = DDFSTOP – (8 * (ilość słów – 1)) dla niskiej rozdzielczości
DDFSTRT = DDFSTOP – (4 * (ilość słów – 2)) dla wysokiej rozdzielczości

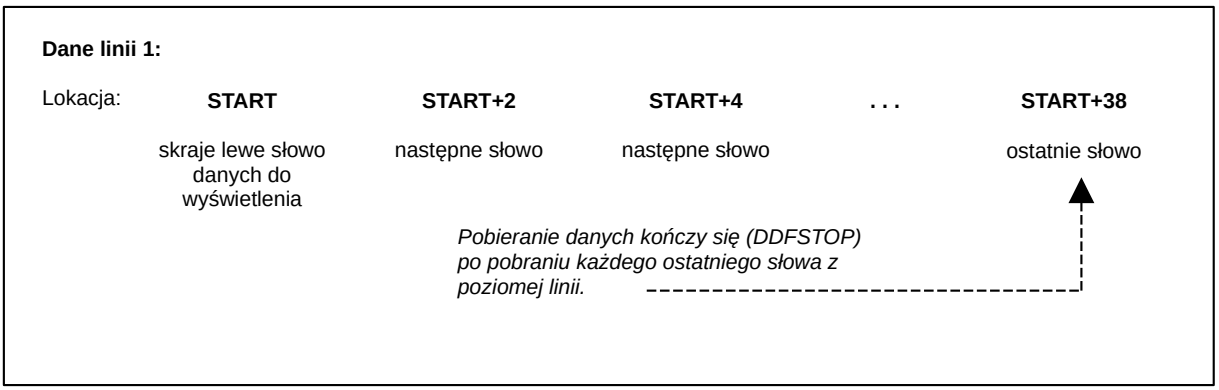
Standardowa wartość DDFSTOP dla niskiej rozdzielczosci wynosi $D0 a dla wysokiej $D4.

Poniższy przykład ustawia oba rejestry na odpowiednie wartości dla niskiej rozdzielczości dla przykładowego playfieldu.

```
LEA CUSTOM,a0     ; adres bazowy dla układów specjalizowanych
MOVE.W #$38,DDFSTRT(a0)  ; najpierw DDFSTRT
MOVE.W #$D0,DDFSTOP(a0)  ; teraz DDFSTOP
```

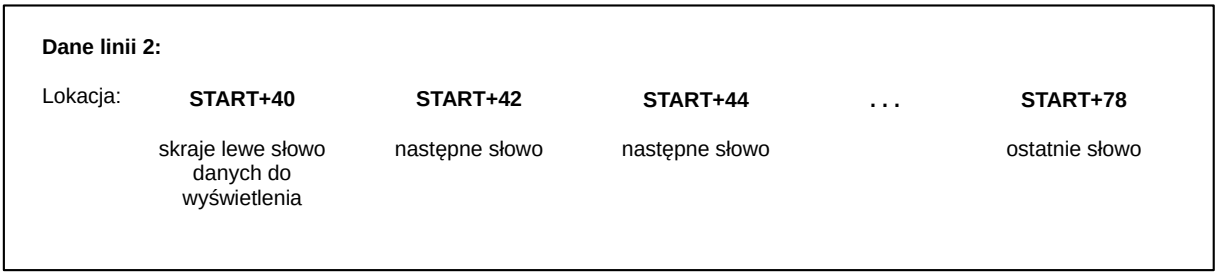
Teraz trzeba wskazać Amidze, które ciągi bajtów należą do każdej z linii poziomych na ekranie. Służą do tego rejestry wartości modulo. Modulo określa ilość bajtów w pamiędzy pomiędzy ostatnim słowem linii poziomej a pierwszym słowem następnej linii. Modulo umożliwia systemowi na konwersję danych bitplanu umieszczonych w pamięci liniowo (w pamięci dane są umieszczone sekwencyjnie, jedna za drugą w kierunku od niższego adresu do wyższego) do formy „prostokątnej” (linia sekwencyjnych danych umieszczona „pod” poprzednią). Dla podstawowego playfieldu, gdzie dane w pamięci mają ten sam rozmiar co okno wyświetlania, modulo ma wartość zero, ponieważ pamięć zawiera dokładnie tyle samo danych ile chcemy wyświetlić na ekranie. Obrazki 3-10 i 3-11 pokazują proste ułożenie w pamięci danych bitplanu oraz w jaki sposób upewnić się, że pobrane zostaną właściwe dane.

System używa rejestrów przechowujących wskaźniki adresów bitplanów (BPLxPTH i BPLxPTL) do zaciągnięcia danych do ekranu. Wskaźniki te są dynamiczne, jak tylko dane zaczną się pobierać, wkaźniki są automatycznie zwiększane aby wskazać następne słowo do pobrania (sane pobierane są w tępie dwa bajty na raz). Kiedy zostaje osiągnięty warunek końca linii (określony przez rejestr DDFSTOP) do wskaźnika dodawana jest wartość modulo tak, aby wskaźnik wskazywał na początek kolejnej linii.



Rys. 3-10: Dane pobierane dla pierwszej linii w przypadku kiedy modulo = 0

Po pobraniu pierwszej linii wskaźniki bitplanów BPLxPTH i BPLxPTL zawierają wartość START+40. Modulo (w tym przypadku równe 0) dodawane jest do aktualnej wartości wskaźnika więc kiedy zaczyna się pobieranie danych dla kolejnej linii, dane są pobierane z poprawnej lokacji w pamięci. Dane kolejnej linii są ulokowane pod adresem START+40.



Rys. 3-11: Dane pobierane dla drugiej linii w przypadku kiedy modulo = 0

Warto zauważyć, że wskaźniki zawsze wskazują na wartość parzystą ponieważ dane pobierane każdorazowo mają rozmiar słowa (dwa bajty).

Istnieją dwa rejestry modulo – BPL1MOD przechowujący modulo dla nieparzystych bitplanów, oraz BPL2MOD dla bitplanów parzystych. Pozwala to na zróżnicowanie modulo dla każdego playfieldu w trybie podwójnego playfieldu. W większości jednak przypadków oba rejestry będą trzymać takie same wartości.

Poniższy przykład ustawia modulo wartością 0 dla playfieldu w niskiej rozdzielczości z pojedynczym bitplanem o nieparzystym numerze.

```
MOVE.W #0,BPL1MOD+CUSTOM   ; przypisz modulo wartość 0
```

Pobieranie danych w wysokiej rozdzielczości

W przypadku wysokiej rozdzielczości dla standardowego playfieldu trzeba pobrać 80 bajtów danych zamiast 40.

Modulo w trybie z przeplotem

Dla trybu z przeplotem trzeba będzie ustawić inną wartość modulo, ponieważ w trybach z przeplotem wyświetlane są dwa półobrazy. Podczas wyświetlenia pierwszego półobrazu Amiga wyświetla tylko linie nieparzyste a wyświetlając drugi półobraz wyświetlane są tylko linie parzyste.

Pełnoekranowe bitplany w trybie z przeplotem mają wysokość 400 linii w NTSC (512 w PAL). Zakładając, że playfield w pamięci ma szerokość 320 punktów, dane dla obrazka z przeplotem mają następujace lokacje w pamięci:

Linia 1	START
Linia 2	START+40
Linia 3	START+80
Linia 4	START+120
...	...

Modulo przyjmuje wartość 40 żeby ominąć linie z drugiego półobrazu. Dane pierwszego półobrazu zaczynają się pod adresem START. Dane drugiego półobrazu mają początek w START+40.

Żeby ustawić wskaźniki bitplanów dla trybu z przeplotem można użyć Coppera.

WYŚWIETLANIE I WYŚWIETLENIE PONOWNE PLAYFIELDU

By rozpocząć wyświetlanie playfieldu trzeba ustawić odpowiednie rejestry wskaźników bitplanów i włączyć DMA bitplanów. Aby je włączyć ustawiamy bit BPLEN w rejestrze DMACON (ang. *DMA Control*). Innstrukcja jak ustawień tego rejestru znajduje się w rozdziale 7, „Układy kontroli systemu”.

Za każdym razem kiedy playfield ma być wyświetlony ponownie musisz ponownie ustawić wskaźniki bitplanów. Ponowne ustawienie jest konieczne, ponieważ wskaźniki te zmieniają się w trakcie wyświetlania obrazu. Używa się do tego odpowiednich instrukcji Coppera i ustawia się je tak, aby zostały uruchomione podczas tzw. pionowego wygaszenia (ang. *vertical blank*).

WŁĄCZANIE WYŚWIETLANIA KOLORÓW

Amiga 1000 jako jedyny model tego komputera ma kolorowe wyjście komponentowe i wymaga ustawienia bitu 9 w rejestrze BPLCON0 aby generować kolorowy obraz na wyjściu. Pozostałe Amigi nie będą generować takiego sygnału bez dodatkowego sprzętu.

> *UWAGA*: Włączenie trybu *color burst* nie ma wpływu na sygnał RGB. Sygnał ten jest generowany poprawnie niezależnie od sygnału wyjściowego na złączu komponentowym.

PODSUMOWANIE WIEDZY O PLAYFIELDZIE

Poniżej wypisane są kroki niezbędne do utworzenia podstawowego playfieldu:

1. **Zdefiniuj cechy playfieldu**

   a) Ustal kolor dla każdego piksela

   - Ustaw rejestry kolorów
   - Zdefiniuj kolor każdego punktu obrazu (wartość bitowa, która wskazuje na odpowiedni rejestr koloru)
   - Utwórz bitplany i ustaw rejestry bitplanów:
     Bity 12-14 rejestru BPLCON0 – ilość bitplanów (BPU2-BPU0)
     BPLxPTH – wskaźnik na pozycję początkową bitplanu w pamięci (długie słowo = 4 bajty)

   b) Określ rozdzielczość:

   - niska rozdzielczosć:
     320 pikseli w każdej linii
     Wyczyść bit 15 w rejestrze BPLCON0 (HIRES)
   - wysoka rozdzielczość:
     640 pikseli w linii
     Ustaw bit 15 w rejestrze BPLCON0 (HIRES)

   c) Określ użycie trybu z przeplotem

   - tryb z przeplotem
     400 linii dla NTSC (512 dla PAL)
     Ustaw bit 2 w rejestrze BPLCON0 (LACE)
   - tryb bez przeploteu
     200 linii dla NTSC (256 dla PAL)
     Wyczyść bit 2 w rejestrze BPLCON0 (LACE)

2. **Zarezerwuj pamięć**. Aby wyliczyć potrzebną ilość bajtów użyj poniższego wzoru:

   ilość bajtów w linii * ilość linii w playfieldzie * ilość bitplanów

3. **Określ rozmiar okna wyświetlania**

   a) Ustaw pozycje startową w rejestrze DIWSTRT
      Pozycja pozioma – byty 0-7
      Pozycja pionowa – bity 8-15

   b) Ustaw pozycję końca w rejestrze DIWSTOP
      Pozycja pozioma – byty 0-7
      Pozycja pionowa – bity 8-15

4. **Ustaw pobieranie danych**. Ustaw rejestrzy DDFSTRT i DDFSTOP:

   a) Rejestr DDFSTRT ustaw według przepisu z sekcji „Ustawienie pozycji początkowej okna wyświetlania”

   b) Rejestr DDFSTOP ustaw według instrucji z sekcji „Ustawienie pozycji końcowej okna wyświetlania”

5. **Ustaw modulo**. Ustaw rejestry BPL1MOD i BPL2MOD. W większości przypadków będą to wartości 0 dla trybu bez przeplotu i 40 dla trybu z przeplotem.

6. **Napisz instrukcje Coppera wyświetlające obraz wielokrotnie**

7. **Włącz kolor**. Dla Amigi 1000: ustaw bit 9 w rejestrze BPLCON0. Pozostałe modele Amigi nie mają takiego wyjścia więc ustawienie to nie ma dla nich znaczenia. Sygnał RGB jest generowany poprawnie niezależnie od tego ustawienia.

PRZYKŁADY KONFIGUROWANIA PROSTYCH PLAYFIELDÓW

Poniższe przykłady pokazują jak ustawić rejestry i napisać copperlistę dla dwóch różnych playfieldów.

Pierwszy przykład definiuje playfield o rozmiarze 320x200 z jednym bitplanem, umieszczonym w pamięci pod adresem $21000. Copperlistę umieszczono w pamięci pod adresem $20000.

Poniższy przykład wymaga pliku „hw_examples.i”, którego zawartość można znaleźć w dodatku I.

```
       LEA CUSTOM,a0             ; a0 – adres bazowy rejestrów układów spec.
       MOVE.W #$1200,BPLCON0(a0) ; jeden bitplan, A1000 „kolorowy komponent”
       MOVE.W #0,BPLCON1(a0)     ; przesunięcie poziome 0
       MOVE.W #0,BPL1MOD(a0)     ; modulo = 0 dla nieparzystych bitplanów
       MOVE.W #$0038,DDFSTRT(a0) ; start poboru danych $38
       MOVE.W #$00D0,DDFSTOP(a0) ; koniec poboru danych $D0
       MOVE.W #$2C81,DIWSTRT(a0) ; DIWSTRT $2C81
       MOVE.W #$F4C1,DIWSTOP(a0) ; DIWSTOP $F4C1
       MOVE.W #$0F00,COLOR00(a0) ; kolor tła czerwony
       MOVE.W #$0FF0,COLOR01(a0) ; kolor 1 żółty
;
; wypełnij bitplan wartościami $FF00FF00 aby uzyskać paski
;
       MOVE.L #$21000,a1         ; początek bitplanu
       MOVE.L #$FF00FF00,d0      ; zapisuj $FF00FF00 w długich słowach
       MOVE.W #2000,d1           ; 2000 długich słów = 8000 bajtów
;
LOOP:  MOVE.L d0,(a1)+           ; zapisz długie słowo
       DBRA d1,LOOP              ; zmniejsz licznik i kontynuuj pętlę do końca...
;
; ustaw copperlistę pod adresem $20000
;
       MOVE.L #$20000,a1         ; wskaż adres docelowy copperlisty
       LEA COPPERL(pc),a2        ; a2 wkazuje dane copperlisty
CLOOP: MOVE.L (a2),(a1)+         ; skopiuj słowo (2 bajty)
       CMPI.L #$FFFFFFFE,(a2)+   ; sprawdź czy końcowe długie słowo copperlisty
       BNE CLOOP                 ; wykonaj pętlę do końca lista Coppera
;
; wskaż Copperowi pozycję copperlisty
;
       MOVE.L #$20000,COPILCH(a0); zapisz w rejestrze lokacji Coppera
       MOVE.W COPJMP1(a0),d0     ; uruchom nową copperlistę spod adresu $20000
;
; start DMA
;
       MOVE.W #(DMAF_SETCLR!DMAF_COPPER!DMAF_RASTER!DMAF_MASTER),DMACON(a0)
                    ; włącz DMA cla bitplanów i Coppera
       BRA  ....    ; skocz do dalszej części programu
;
; dane copperlisty
;
COPPERL :
       DC. W BPL1PTH,$0002 ; zapisz $0002 pod adresem $0E0 (BPL1PTH)
       DC. W BPL1PTL,$1000 ; zapisz $1000 pod adresem $0E2 (BPL1PTL)
       DC. W $FFFF,$FFFE   ; koniec copperlisty
;
```

Drugi przykład ustawia wysoką rozdzielczość w trybie z przeplotem i jedym bitplanem. Ten przykład również wymaga pliku „hw_examples.i” z dodatku I.

```
       LEA CUSTOM,a0             ; a0 – adres bazowy rejestrów układów spec.
       MOVE.W #$9204,BPLCONO(a0) ; wysoka rozdzielczosć, jeden bitplan, przeplot
       MOVE.W #0,BPLCON1(a0)     ; przesunięcie poziome 0
       MOVE.W #80,BPL1MOD(a0)    ; modulo = 80 dla nieparzystych bitplanów
       MOVE.W #80,BPL2MOD(a0)    ; to samo dla parzystych
       MOVE.W #$003C,DDFSTRT(a0) ; data-fetch start dla hires
       MOVE.W #$00D4,DDFSTOP(a0) ; data-fetch stop
       MOVE.W #$2C81,DIWSTRT(a0) ; początek okna wyświetlania
       MOVE.W #$F4C1,DIWSTOP(a0) ; koniec okna wyświetlania
;
; ustaw rejestry kolorów
;
       MOVE.W #$000F,COLOR00(a0) ; kolor tła niebieski
       MOVE.W #$0FFF,COLOR01(a0) ; kolor 1 biały
;
; ustaw bitplan pod adresem $20000
;
       LEA $20000,a1       ; a1 wskazuje na bitplan
       LEA CHARLIST(pc),a2 ; a2 wskazuje na listę znaków
       MOVE.W #400,d1      ; zapisz 400 linii danych
       MOVE.W #20,d0       ; 20 długich słów na linię
L1:
       MOVE.L (a2),(a1)+   ; zapisz długie słowo
       DBF d0,L1           ; zmniejsz licznik i kontynuuj pętlę aż zejdzie poniżej zera
;
       MOVE.W #20,d0       ; zresetuj licznik długich słów
       ADDQ.L #4,a2        ; wskaż następne słowo na liście znaków
       CMPI.L #$FFFFFFFF,(a2)    ; czy koniec listy znaków?
       BNE L2
       LEA CHARLIST(pc),a2 ; tak, zresetuj a2 do początku listy
L2:    DBF d1,L1           ; zmniejsz licznik linii i kontynuuj pętlę aż do końca
;
; start DMA
;

       MOVE.W #(DMAF_SETCLR!DMAF_RASTER!DMAF_MASTER),DMACON(a0)

; Włącz DMA tylko dla bitplanów, bez Coppera
; Ponieważ ten przykład nie korzysta z copperlisty, program czeka w pętli
; na pionowe wygaszenie. Jak nastąpi sprawdza bit LOF (długa ramka,
; ang. long frame) w rejestrze VPOSR. LOF = 0 oznacza krótką ramkę i wskaźniki
; bitplanów są ustawiane na adres $20050. Jeśli LOF = 1 to mamy długą ramkę
; i wskaźniki bitplanów ustawiane są na adres $20000. Dzięki temu poprawnie
; obsługujemy długie i krótkie ramki (półekrany wyświetlane w trybie z przeplotem)

VLOOP:
       MOVE.W INTREQR(a0),d0     ; odczytaj rządania przerwań
       AND.W #$0020,d0           ; maskuj wszystkie bity poza bitem wygaszenia pionowego
       BEQ VLOOP                 ; wykonuj pętlę aż do pionowego wygaszenia
       MOVE.W #$0020,INTREQ(a0)  ; resetuj przerwanie pionowe
       MOVE.W VPOSR(a0),d0       ; odczytaj bit LOF, zapisz w d0
       BPL VL1                   ; If LOF = 0, skocz
       MOVE.L #$20000,BPL1PTH(a0); LOF = 1, wskaż $20000
       BRA VLOOP                 ; powrót do początku pętli
VL1:
       MOVE.L #$20050,BPL1PTH(a0); LOF = 0, wskaż $20050
       BRA VLOOP                 ; powrót do początku pętli
;
;  lista znaków
;
CHARLIST:
       DC.L   $18FC3DF0,$3C6666D8,$3C66C0CC,$667CC0CC
       DC.L   $7E66C0CC,$C36666D8,$C3FC3DF0,$00000000
       DC.L   $FFFFFFFF
```

Tworzenie obrazu w trybie podwójnego playfieldu

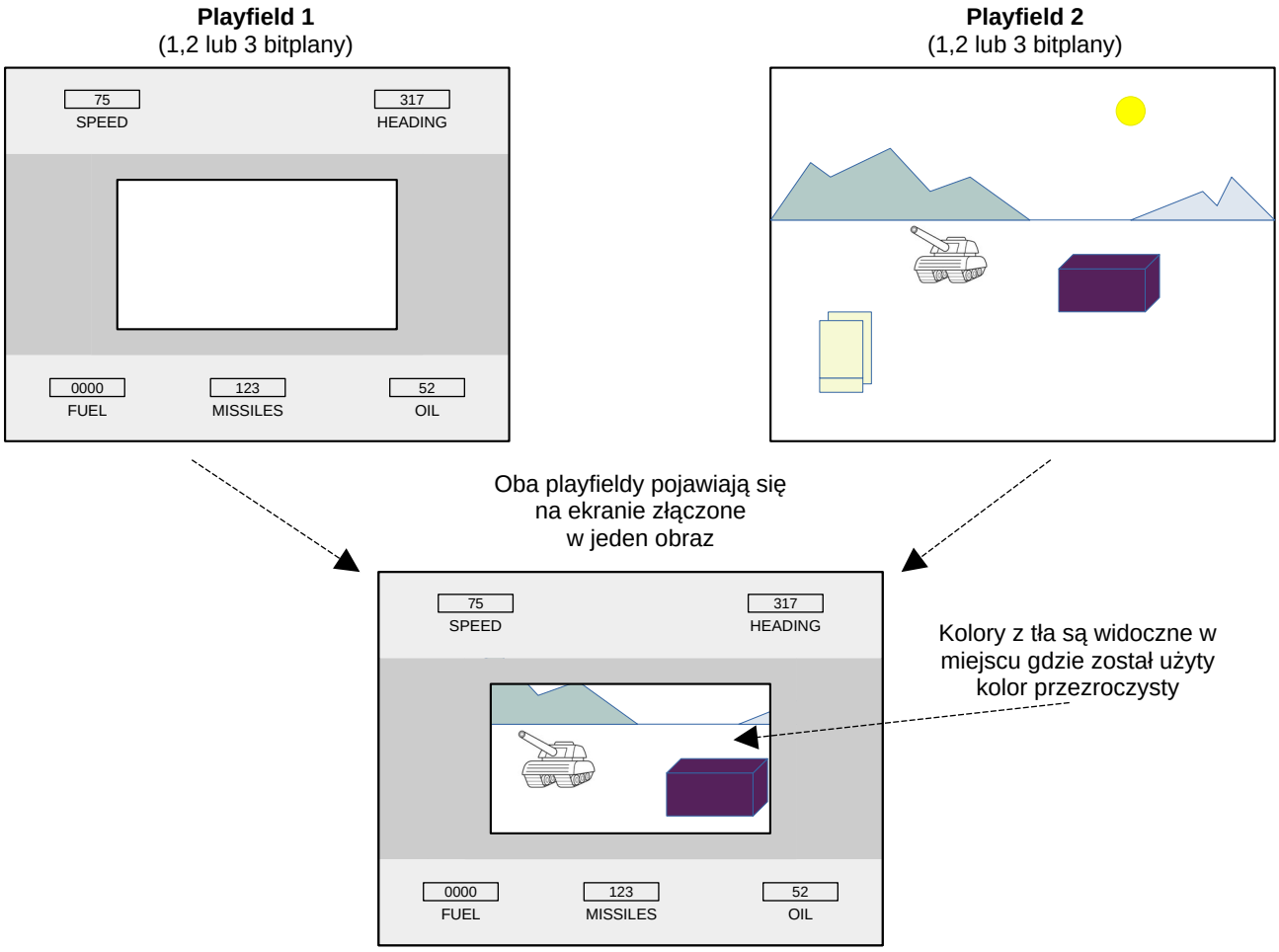
Aby uzyskać większą elastyczność w projektowaniu wyświetlanego tła można posłużyć się dwoma playfieldami. W trybie podwójnego playfieldu pierwszy playfield jest wyświetlany ponad drugim. Weźmy dla przykładu grę, w której w tle może odbywać się jakaś akcja a z przodu wyświetlany jest panel sterowania. Dzięki temu można zmieniać tło lub front bez konieczności zmieniania parametrów wyświetlanego ekranu. W trybie podwójnego playfieldu można przemieszczać oba playfieldy zupełnie niezależnie od siebie.

Tryb podwójnego playfieldu w większości jest podobny do trybu pojedynczego playfieldu. Różnice w porównaniu do trybu pojedynczego są następujące:

- Każdy playfield może zawierać do trzech bitplanów.
- Kolory każdego playfieldu (7 kolorów plus przezroczysty) są brane z niezależnych od siebie zestawów rejestrów.
- Musisz ustawić odpowiedni bit kontrolny, żeby włączyć tryb podwójny.

Na rys. 3-12 jeden z kolorów każdego playfieldu jest „przezroczysty” (kolor 0 w playfieldzie 1 i kolor 8 w playfieldzie 2). Możesz używać przezroczystego koloru aby uzyskać efekt „prześwitywania” przez tło.

W trybie podwójnego playfieldu, każdy z playfieldów składa się z maksymalnie trzech bitplanów. Rejestry kolorów od 0 do 7 są przypisane do playfieldu 1 a rejestry od 8 do 15 przypisane są do playfieldu 2.

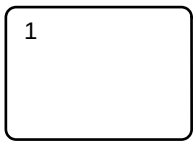
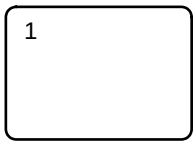
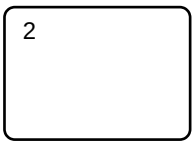
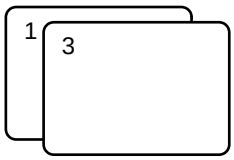
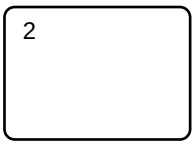
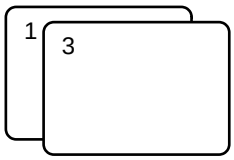
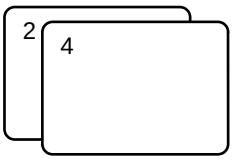
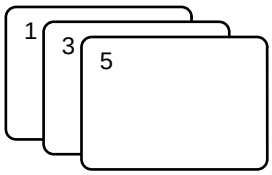
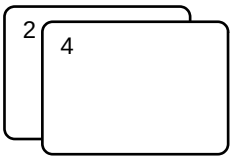
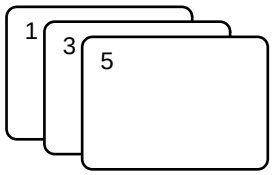
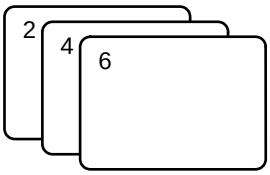


Rys. 3-12: Obraz w trybie podwójnego playfieldu

PRZYPISANIE BITPLANU W TRYBIE PODWÓJNEGO PLAYFIELDU

Playfield 1 składa się z trzech zgrupowanych razem nieparzystych bitplanów (1,3 i 5). Podonie trzy parzyste bitplany (2,4 i 6) tworzą grupę i razem składają się na playfield 2. Rys. 3-13 pokazuje na jakie możliwe sposoby można przypisać bitplany do obu playfieldów.

O bitplanach w trybie podwójnego playfieldu. W wysokiej rozdzielczości każdy playfield może składać się z maksymalnie dwóch bitplanów: 1 i 3 dla playfieldu 1 i 2 i 4 dla playfieldu 2.

Liczba dostępnych bitplanów	Playfield 1 *	Playfield 2 *
0	brak	brak
1		
2		
3		
4		
5		
6		

* Każdy playfield może być położony przed lub za drugim zależnie od ustawień odpowiedniego bitu.

Rys. 3-13: W jaki sposób bitplany przypisane są do podwójnych playfieldów

REJESTRY KOLORÓW W TRYBIE PODWÓJNEGO PLAYFIELDU

Gdy używa się podwójnego playfieldu odpowiedni rejestr koloru dla playfieldu 1 określany jest na podstawie konfiguracji bitów z bitplanów 1, 3 i 5. Bity z bitplanu 5 mają największe znaczenie (skrajna lewa pozycja w liczbie dwójkowej). Bit z bitplanu 1 ma najniższe znaczenie (skrajna prawa pozycja z liczbie dwójkowej). Poniższa tabela pokazuje możliwe kombinacje bitów i odpowiadające im rejestry kolorów dla pierwszego playfieldu.

PLAYFIELD 1	
Kombinacja bitów	**Wybrany rejestr koloru**
000	przezroczysty
001	COLOR01
010	COLOR02
011	COLOR03
100	COLOR04
101	COLOR05
110	COLOR06
111	COLOR07

Tabela 3-10: Rejestry kolorów playfieldu 1 – tryb niskiej rozdzielczości

Dla playfieldu drugiego kombinacja bitów definiująca rejestr koloru jest brana z bitplanów 2, 4 i 6. Bity z bitplanu 6 mają największe znaczenie (skrajna lewa pozycja w liczbie dwójkowej). Bit z bitplanu 2 ma najniższe znaczenie (skrajna prawa pozycja z liczbie dwójkowej). Poniższa tabela pokazuje możliwe kombinacje bitów i odpowiadające im rejestry kolorów dla drugiego playfieldu.

PLAYFIELD 2	
Kombinacja bitów	**Wybrany rejestr koloru**
000	przezroczysty
001	COLOR09
010	COLOR10
011	COLOR11
100	COLOR12
101	COLOR13
110	COLOR14
111	COLOR15

Tabela 3-11: Rejestry kolorów playfieldu 2 – tryb niskiej rozdzielczości

Kombinacja 000 oznacza tryb przezroczystości. To znaczy, że w tym miejscu playfieldu będą widoczne obiekty znajdujące się „pod nim”.

Tabela 3-12 pokazuje rejestry kolorów dla podwójnego playfieldu w trybie wysokiej rozdielczości.

PLAYFIELD 1	
Kombinacja bitów	**Wybrany rejestr koloru**
00	przezroczysty
01	COLOR01
10	COLOR02
11	COLOR03
PLAYFIELD 2	
Kombinacja bitów	**Wybrany rejestr koloru**
00	przezroczysty
01	COLOR09
10	COLOR10
11	COLOR11

Tabela 3-12: Rejestry kolorów playfieldów 1 i 2 – tryb wysokiej rozdzielczości

PRIORYTETY I KONTOLA PODWÓJNYCH PLAYFIELDÓW

Jednemu z dwóch playfieldów można nadać priorytet, oznacza to, że jeden z playfieldów może być wyświetlany ponad drugim (przesłaniać drugi). Służy do tego bit PF2PRI (bit 6) rejestru BPLCON2. Ustawienie tego bitu określa priorytet playfieldu 2 nad playfieldem 1. Kiedy bit ten ma wartość 0 – playfield 1 ma priorytet.

Istnieje również możliwość kontroli priorytetów pomiędzy playfieldami i sprite’ami. Jak ją kontrolować dowiesz się z rozdziału 7.

Każdy playfield może być zarządzany osobno według poniższych zasad:

- mogą mieć różne rozmiary w pamięci, inna ich część może być oznaczona do wyświetlenia
- mogą być przesuwane niezależnie od siebie

Ważna uwaga. Przesuwanie jednego playfieldu przy jednoczesnym utrzymywaniu drugiego w stałej pozycji wymaga dodatkowej uwagi. Przesuwając playfieldy niskiej rozdzielczości msuisz pobierać jedno słowo więcej niż wynika z szerokości danego playfieldu. (dwa słowa dla wysokiej rozdzielczości). Te dodatkowe dane służą do pokazania na ekranie kiedy ma miejsce przesuwanie (jenda część obrazu znika za ekranem a po przeciwnej stronie pojawia się nowa część – to właśnie te dodatkowe dane). Ponieważ dla obu playfieldów dostępny jest tylko jeden rejestr dla początku pobou danych i jeden dla jego końca, przesuwając tylko jeden playfield musisz odpowiednio ustawić start i koniec poboru danych. Dodatkowo musisz odpowiednio ustawić modulo i wskaźniki bitplanów playfieldu, który nie zmienia swojej pozycji na ekranie. W niskiej rozdzielczości wkaźniki i modulo pomniejsza się o 2 (dodaje -2 do ich obecnej wartości). W wysokiej rozdzielczości musisz oba parametry pomniejszyć o 4.

WŁĄCZANIE TRYBU PODWÓJEGO PLAYFIELDU

Aby włączyć tryb podwójego playfieldu musisz ustawić bit 10 (DBLPF) w rejestrze BPLCON0. Włączenie trgo trybu zmienia sposób w jaki Amiga grupuje bitplany i rejestry kolorów: wszystkie nieparzyste bitplany łączą się w pierwszy playfield, wszystkie parzyste zaś w drugi. Zmienia się również sposób w jaki Amiga będzie przemiszczać bitplany po ekranie.

PODSUMOWANIE WIADOMOŚCI O PODWÓJNYM PLAYFIELDZIE

Kroki jakie należy wykonać przy aktywacji trybu podwójnego playfieldu są niemal identyczne jak w przypadku pojedynczego playfieldu. Różnice są następujące:

- **Rejetry kolorów**. Pamiętać trzeba, że pierwszy playfield używa rejestrów kolorów od 0-7, rejestry 8-15 przypisane są do playfieldu drugiego (oczywiście jeśli playfieldy mają po trzy bitplany, w przeciwnym wypadku możliwych rejestrów kolorów jest odpowiednio mniej).
- **Tworzenie bitplanów**. Playfield 1 łączy w sobie bitplany 1,3 i 5 a playfield 2 – 2,4 i 6
- **Ustawienie rejestrów modulo**. Ponieważ używane będą oba playfieldy trzeba ustawić oba rejestry: BPL1MOD i BPL2MOD.

Dodatkowo trzeba wykonać poniższe kroki:

- **Określenie priorytetu**. Jeśli priorytet ma należeć do playfieldu 2, trzeba ustawić bit 6 (PF2PRI) w rejestrze BPLCON2.
- **Włączenie trybu podwójnego playfieldu**. Ustaw bit 10 (DBLPF) w rejestrze BPLCON0.

Bitplany i okna wyświetlania o różnych rozmiarach

Wiesz już jak tworzyć pojedyncze i podwójne playfieldy, których rozmiary w pamięci były takie same jak zdefiniowane wcześniej okno wyświetlania. W tej sekcji omówimy jak używać playfieldu, którego rozmiar w pamięci będzie większy niż obraz wyświetlany na ekranie, jak definiować okna wyświetlania, które są większe lub mniejsze niż standardowy playfield, oraz jak jak przemieszczać oknem wyświetlania po większym obrazie.

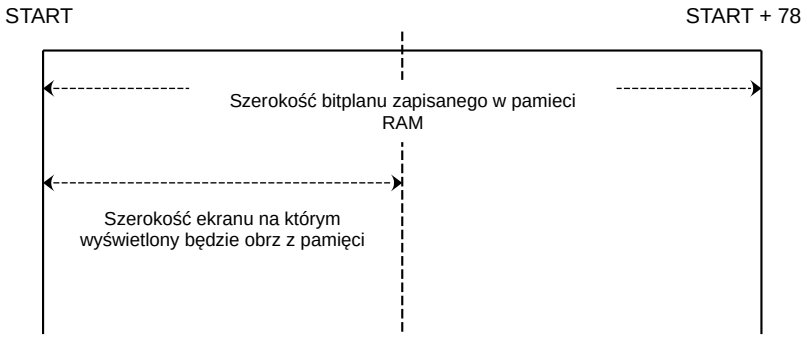
KIEDY OBRAZEK JEST WIĘKSZY NIŻ OKNO WYŚWIETLANIA

Jeśli dane w pamięci reprezentują większy obrazek niż okno wyświetlania, musisz określić które jego części będą wyświetlane. Sposób wyświetlania części większego obrazu różni się od sposobu wyświetlania standardowego playfieldu, który opisywaliśmy do tej pory:

- Jeśli obraz a pamięci jest większy od okna wyświetlania, musisz ponownie przeliczyć modulo. Modulo będzie miało wartość różną od 0.
- Większy obraz to też więcej zarezerwowanej pamięci.

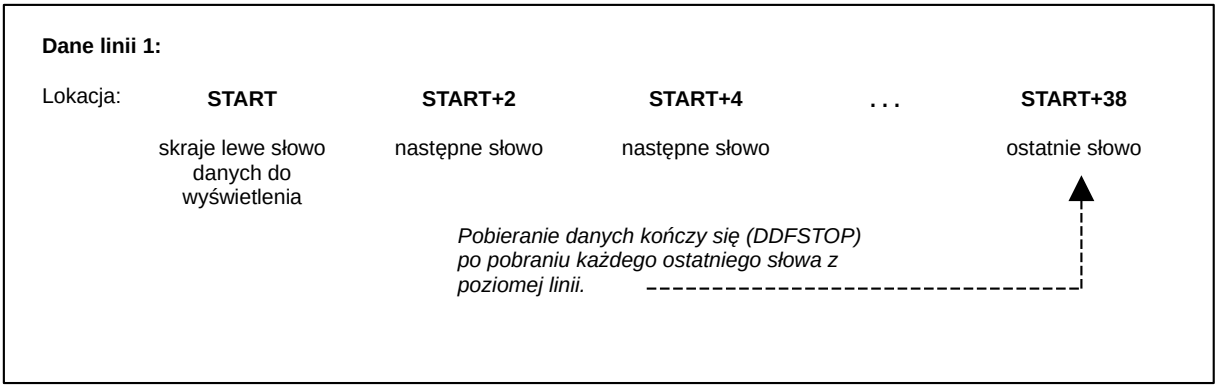
Określenie modulo

Dla obrazka szerszego niż okno wyświetlania, musisz tak określić modulo, żeby pobrane zostały odpowiednie dane w każdej linii do wyświetlenia. Jako przykład załóżmy, że okno wyświetlania jest rozmiaru 320 punktów, więc na każdą wyświetlaną linię przypada 40 bajtów danych. Obraz w pamięci natomiast ma dwa razy większą szerokość (80 bajtów w każdej linii). Załóżmy też, że chcemy wyświetlić lewą część tego dużego obrazka. Sytuację opisaną przedstwia rysunek 3-14 poniżej.



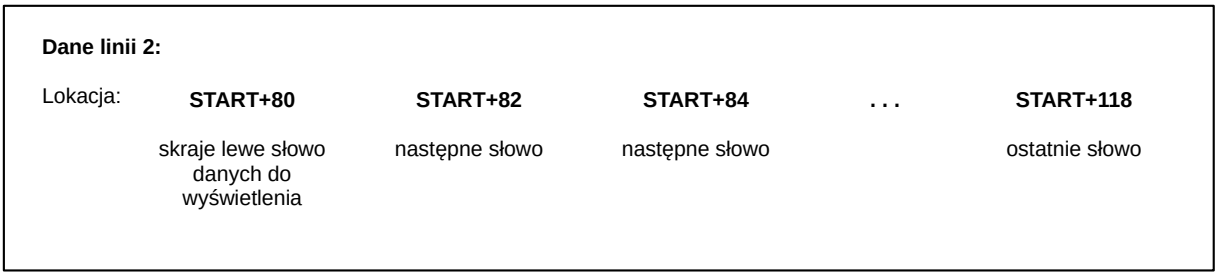
Rys. 3-14: Obrazek w pamięci większy niż okno wyświetlania

Na każdą linię przypada 40 bajtów danych do pobrania, pobór danych dla linii 1 pokazany został na rysunku 3-15.



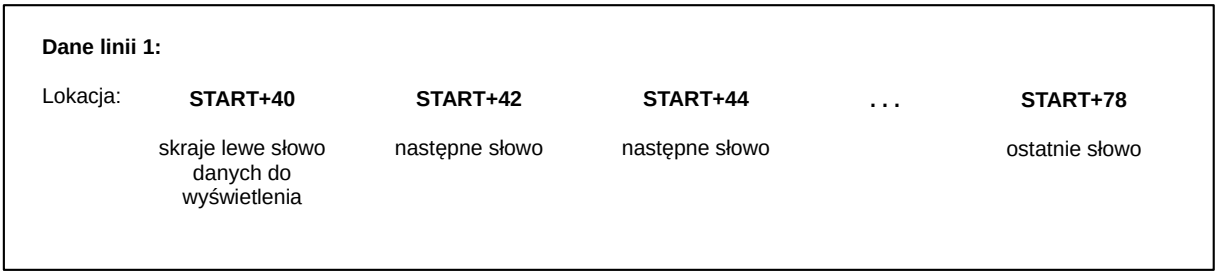
Rys. 3-15: Dane pobierane dla pierwszej linii w przypadku kiedy modulo = 40

Po pobraniu pierwszej linii wskaźniki bitplanów BPLxPTH i BPLxPTL zawierają wartość START+40. Modulo, tym razem równe 40, dodawane jest do aktualnej wartości wskaźnika więc kiedy zaczyna się pobieranie danych dla kolejnej linii, dane są pobierane z poprawnej lokacji w pamięci. Dane kolejnej linii są ulokowane pod adresem START+80 co pokazano na rysunku 3-16.



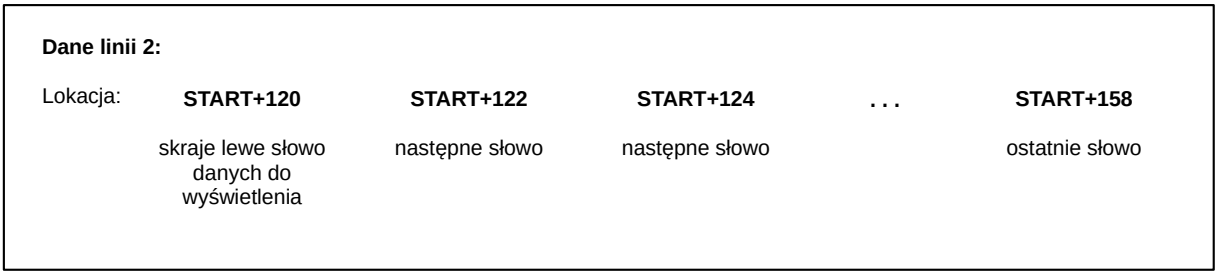
Rys. 3-16: Dane pobierane dla drugiej linii w przypadku kiedy modulo = 40

Aby wyświetlić prawą część obrazu trzeba ustawić początkowe wskaźniki bitplanów na wartosć START+40 zamiast START pozostawiając modulo o wartości 40. Ułożenie danych prezentują rysunki 3-17 i 3-18.



Rys. 3-17: Dane dla pierwszej linii – prawa część większego obrazu

Po pobraniu pierwszej linii rejestry wskaźników bitplanów przechowują wartość START+80. Następnie modulo (40) dodawane jest do rejestrów wskaźników aby pominąć niewidoczną lewą część dużego obrazka i rozpoczyna się pobieranie kolejnej linii do wyświetlenia.



Rys. 3-18: Dane dla drugiej linii – prawa część większego obrazu

Pamiętać należy, że w trybie wysokiej rozdzielczości trzeba pobrać dwa razy więcej danych. Dla standardowego ekranu pobierane jest 80 bajtów danych zamiast 40.

Określenie wartości poboru danych

Rejestry poboru danych określają początkową i końcową pozycję położenia danych dla każdej linii wyświetlanej na ekranie. Wartości zapisywane do tych rejestrów wyznacza się w ten sam sposób jak dla pojedynczego playfieldu, tak jak opisano w sekcji „Tworzenie podstawowego playfieldu”.

Rezerwowanie pamięci

Dla większych obrazków trzeba zarezerwować więcej pamięci. Oto wzór, który pomoże wyznaczyć odpowiednią wielkość pamięci:

*ilość bajtów na linię * ilość linii w playfieldzie * ilość bitplanów*

Ilość bajtów musi być parzysta. Więc jeśli weźmiemy pod uwagę playfield posiadający dwa bitplany, opisany w tej sekcji, to łątwo wyliczyć, że jego zapotrzebowanie na pamięć wynosi:

80 * 200 * 2 = 32000 bajtów

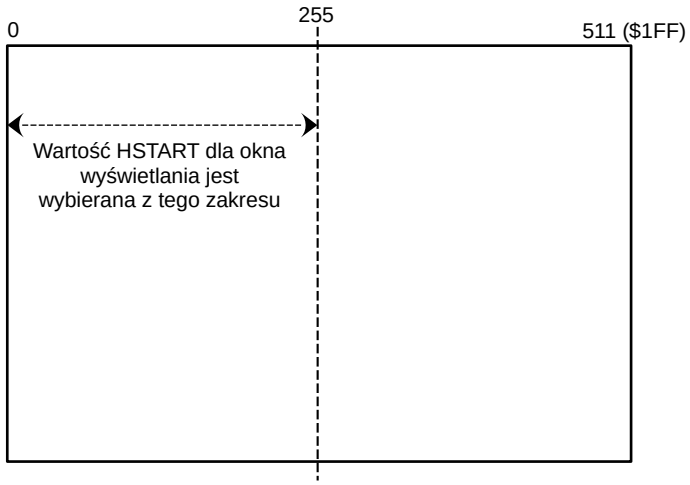
Warto zauważyć, że powyższa wartość to zapotrzebowanie na pamięć dla samych bitplanów. Sprity, animacje i dane dźwiękowe będą potrzebowały dodatkowych ilości pamięci.

Jednym z podstawowych ograniczeń dla wielkości playfieldu jest ilosć dostępnej pamięci typu Chip. Dla przykładu, playfield o rozmiarze 2000x2000 używający 5 bitplanów przekroczy ilość dostępnej pamięci Chip dostępnej nawet w Amigach posiadających jej najwięcej (Amiga 1200, 3000 i 4000 posiadają 2 MB pamięci Chip). Kolejnym ograniczeniem na rozmiar bitplanów jest wartość modulo, która ogranicza szerokość bitplanu do 262144 pikseli.

Z praktycznego punktu widzenia wpływ na ograniczenie rozmiaru playfieldu mają również rejestry blittera (chyba, że do operacji rysowania użyjemy procesora 680x0). W przypadku oryginalnych układów specjalizowanych (OCS – Original Chip Set) maksymalny obszar jaki może wyrysować blitter to 1008 na 1024 punkty. W nowszy, rozszerzonym zestawie układów (ECS – Enhanced Chip Set) powierzchnia ta została rozszerzona do 16368 x 16384 punktów. Więcej informacji o ECS i blitterze znajduje się w rozdziale „Dodatek C, Enhanced Chip Set”.

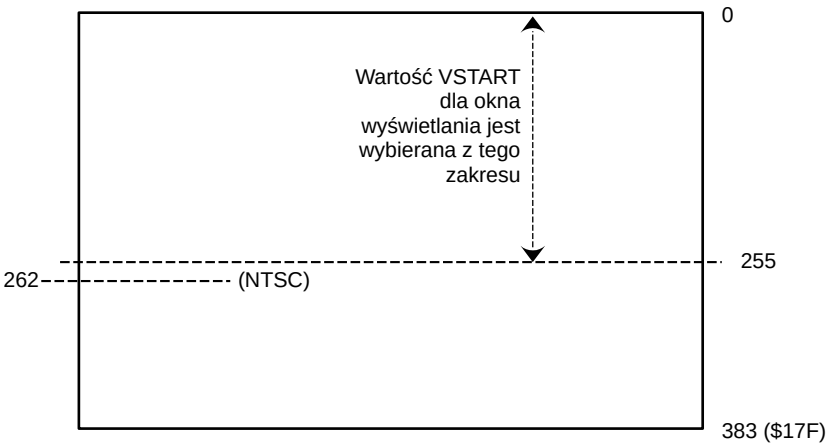
Wybór pozycji startowej okna wyświetlania

Pozycja początkowa okna wyświetlania to poziome i poinowe współrzędne górnego lewego rogu tego okna. Obie te wartości, nazywane HSTART i VSTART, są przechowywane w jednym rejestrze – DIWSTRT. 8 bitów przypisanych do HSTART mogą pomieścić pierwsze 256 z możliwych pozycji (punktów) licząc od lewej strony ekranu. Okno wyświetlania może mieć swój początek w dowolnym punkcie tego zakresu.



Rys. 3-19: Pozioma pozycja początkowa okna wyświetlania

8 bitów przypisanych do VSTART mogą przechować pierwsze 256 z możliwych pozycji (linii) licząc od góry ekranu.



Rys. 3-20: Pionowa pozycja początkowa okna wyświetlania

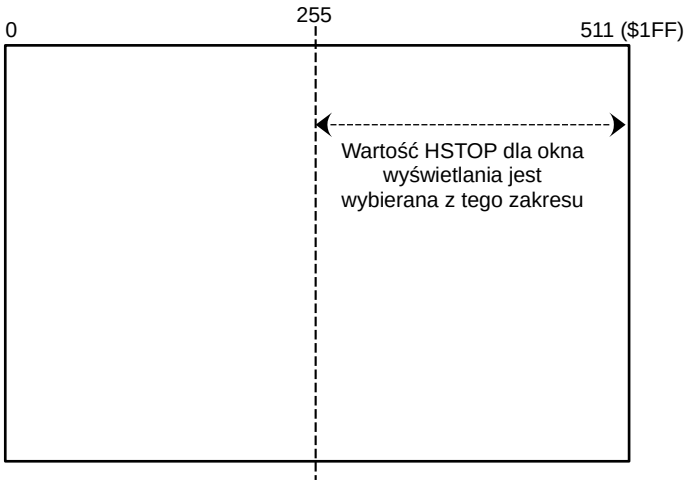
Warto zapamiętać, że wartości pozycji początkowej okna wyświetlania są dobierane dla ekranu w niskiej rozdzielczości w trybie bez przeplotu. W przypadku trybu z przeplotem warto zapamiętać, że wysokość okna wyświetlania musi być parzysta, aby oba półobrazy były tej samej wysokości.

Aby ustawić poprawnie pozycję początkową okna wyświetlania bity od 0 do 7 rejestru DIWSTRT zapisujemy wybraną wartością HSTART, natomiast ustaloną wartością VSTART zapisujemy bity od 8 do 15.

Wybór pozycji końcowej

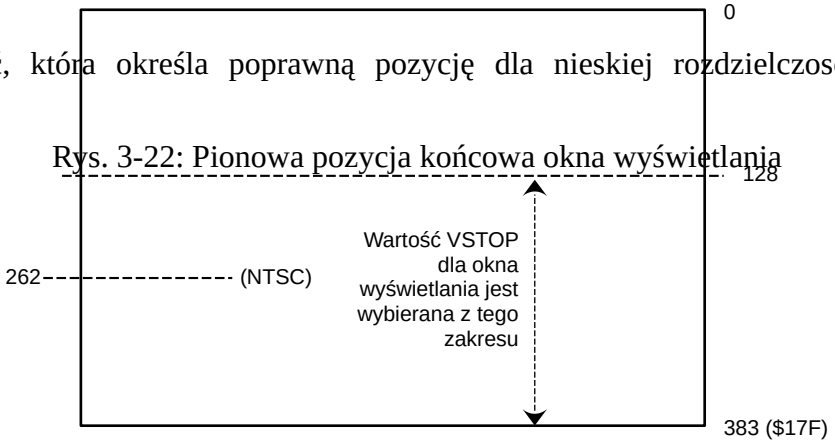
Końcowa pozycja okna wyświetlania ospowiadają poziomym i pionowym współrzędnym dolnego prawego rogu tego okna. Noszą nazwy HSTOP i VSTOP a przechowuje je rejestr DIWSTOP.

W sekcji „Tworzenie podstawowego playfieldu” znajdziesz informację w jaki sposób ustawia się rejestry DIWSTRT i DIWSTOP.



Rys. 3-21: Pozioma pozycja końcowa okna wyświetlania

Wybierz wartość, która określa poprawną pozycję dla nieskiej rozdzielczosci w trybie bez przeplotu.



Rys. 3-22: Pionowa pozycja końcowa okna wyświetlania

Aby ustawić poprawnie pozycję końcową okna wyświetlania bity od 0 do 7 rejestru DIWSTOP zapisujemy wybraną wartością HSTOP, natomiast ustaloną wartością VSTOP zapisujemy bity od 8 do 15.

MAKSYMALNY ROZMIAR OKNA WYŚWIETLANIA

Maksymalna wielkość wyświetlanego playfieldu jest określana przez maksymalną ilość linii i maksymalną ilość kolumn (punktów w linii). Pionowe ograniczenia są proste. Dane nie mogą być wyświetlone w obszarze wygaszania pionowego. Poniższa tabela pokazuje jakie numery linii dają możliwosć wyświetlenia obrazu.

	NTSC		**PAL**	
Początek pionowego wygaszenia	0		0	
Koniec pionowego wygaszenia	$15 (21)		$1D (29)	
	NTSC	**NTSC z przeplotem**	**PAL**	**PAL z przeplotem**
Ilość linii możliwych do wyświetlenia	241	483 =525-(21*2)	283	567 =625-(29*2)

Tabela 3-13: Maksymalna ilosć linii w pionie

W poziomie sytuacja jest podobna. Któro mówiąc, układy graficzne Amigi nie pozwalają na przekroczenie wartości $D8 jako DDFSTOP a $18 jako DDFSTRT. Daje to maksymalną ilość 25 słów danych pobieraną w niskiej rozdzielczości. W wysokiej rozdzielczosci wartość maksimum wynosi 49 słów, ponieważ maksymalna wartość DDFSTOP pozostaje niezmieniona ($D8) a w końcowej pozycji pobierane jest tylko jedno słowo. Poziome wygaszenie nakłada jednak dodatkowe ograniczenie dające w rezultacie 368 punktów niskiej rozdzielczości (23 słowa). Wartość ta jest identyczna dla NTSC i PAL. Dodatkowo warto zauważyć, że ustawienie poboru danych wartością poniżej $38 skutkuje wyłączeniem niektórych sprite’ów.

	Niska rozdzielczość	Wysoka rozdzielczość
DDFSTRT (standard)	$0038	$003C
DDFSTOP (standard)	$00D0	$00D4
DDFSTRT (ograniczony sprzętem)	$0018	$0018
DDFSTOP (ograniczony sprzętem)	$00D8	$00D8
Maksymalna ilość słów	25	49
Maksymalna ilość pikseli	368 (niskiej rozdzielczości)	

Tabela 3-14: Maksymalna ilosć punktów w poziomie

Ograniczenia opisane w pozyższej sekcji dotyczą Amig z układami OCS. Zastosowanie układów ECS pozwoliło na zmiany w tych ograniczeniach. Więej informacji znajduje się w rozdziale „Dodatek C, Enhanced Chip Set”.

Przemieszczanie (ang. *scrolling*) playfieldów

Jeśli chcesz przemieszczać wyświetlane tło, możesz stworzyć playfield większy niż okno wyświetlania i przesuwać go. Jeśli używasz podwójnego playfieldu, każdy playfield możesz przesuwać osobno.

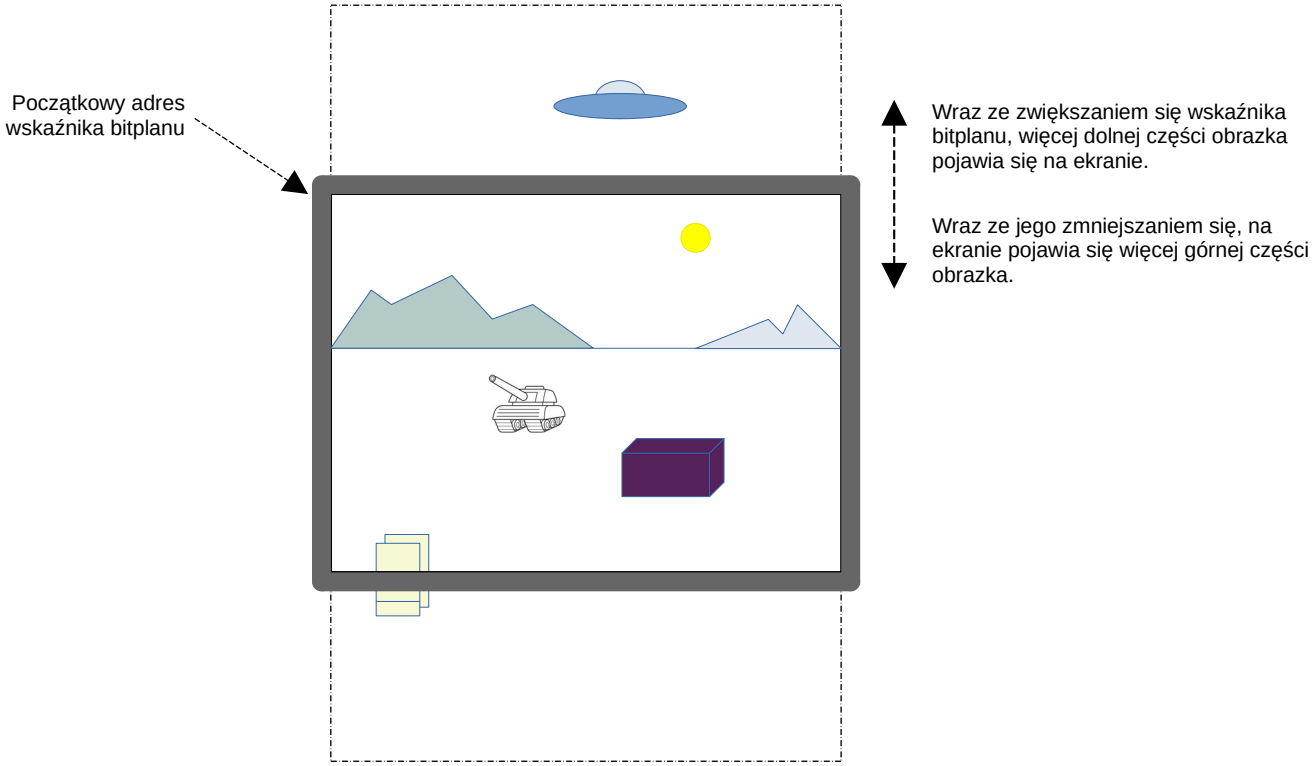
Przemiszczanie pionowe powoduje płynne przzesuwanie playfieldu na ekranie w górę lub w dół. Wszystko co musisz zrobić to zwiększać lub zmniejszać adres startowy wskaźników bitplanów o wartość równą wielkości jednej linii playfieldu. W efekcie będzie możliwe wyświetlenie na ekranie górnej lub dolnej części obrazka, niewidocznej do tej pory.

W przypadku przesuwania poziomego, playfield wydaje się poruszać w lewo lub w prawo. Poziome przesuwanie działa odmiennie od pionowego. W tym przypadku musisz odczytać o jedno słowo więcej danych dla każdej wyświetlanej linii i opóźnić wyświetlenie tych danych.

W obu przypadkach, ponowne ustawianie wskaźników lub rejestrów poboru danych może zostać przerzucone na barki Coppera i wykonane w czasie pionowego wygaszenia.

PRZESUWANIE W PIONIE

Playfield można przesuwać w pionie w górę lub w dół. Za każdym razem kiedy playfield jest wyświetlany, wskaźniki bitplanów są ustawiane na wyższą lub niższą część wyświetlanego obrazka znajdującego się w pamięci. Kiedy wartość wskaźnika jest zwiększana, coraz większa część dołu obrazka pojawia się na ekranie i obraz wydaje się przesuwać w górę. Na wyświetlaczach pracujących w systemie NTSC, który pozwala wyświetlić 200 linii w pionie, każdy pojedynczy krok może mieć wartość 1/200 wysokości ekranu (jednej linii). W trybie z przeplotem pojedynczy ktok to 1/400 wysokości ekranu pod warunkiem zastosowania sprytnej metody manipulującej wskaźnikami. Jednak zalecanym minimalnym krokiem dla trybu z przeplotem jest wartość dwóch linii.W systemie PAL wyświetlającym 256 linii, pojedynczy krok może wynośić 1/256 wysokości ekranu, lub 1.512 w trybie z przeplotem.



Rys. 3-23: Przesuwanie w pionie

Aby móc przesuwać playfield w pionie, bitplany muszą być wyższe o wartość równą zakładanemu przesunięciu. Następnie potrzebna będzie metoda modyfikująca wskaźniki bitplanów realizująca założone przesunięcie. Na koniec dane te należy dostarczyć Copperowi, który zadba o ich odpowienie użycie.

Załóżmy przesunięcie w pionie o jedną linię na raz. Żeby to osiągnąć, przed wyświetleniem każdej klatki, wskaźniki bitplanów należy zwiększyć o wartość konieczną do tego, aby były one unstawione na początek kolejnej linii obrazka. Dla obrazka w nieskiej rozdzielczości o szerokości równej szerokości ekranu (modulo równe 0) wskaźnik należy zwiększać o kolejne 40 bajtów każdorazowo.

PRZESUWANIE W POZIOMIE

Playfieldy można także przesuwać od lewej do prawej i od prawej do lewej. Prędkość przesunięcia zależna jest od opóźnienia wyrażonego w punktach. Opóźnienie oznacza, że dodatkowe słowo pobierane w czasie poboru danych linii nie jest natychmiastowo wyświetlane. To nadliczbowe słowo umieszczane jest zaraz z lewej strony lewej krawędzi okna jeszcze przed pobraniem samych danych. Jak tylko obraz przesuwa się od lewej do prawej, bity z tego dodatkowego słowa zaczynają pojawiać się na ekranie a bity przy prawej krawędzi znikają z ekranu. Każdy piksel opóźnienia jest równoważny z przesunięciem o jeden punkt. Im większe opóźnienie tym większa prędkość przesunięcia. Opóźnienie maksymalnie może mieć wartość 15 punktów. W wysokiej rozdzielczości przesunięcie jest zwiększane o dwa piksele. Na rysunku 3-24 pokazano w jaki sposób opóźnienie i dodatkowe pobrane dane są powiązane z ruchem obrazu w poziomie.

Aby umożliwić przemieszczanie playfieldu w poziomie trzeba:

- zdefiniować bitplany szerokie na tyle, żeby umożliwić ich przesunięcie
- ustawić rejestry poboru danych na odpowiednie miejsce w każdej linii, wliczając w to dodatkowe dwa bajty danych

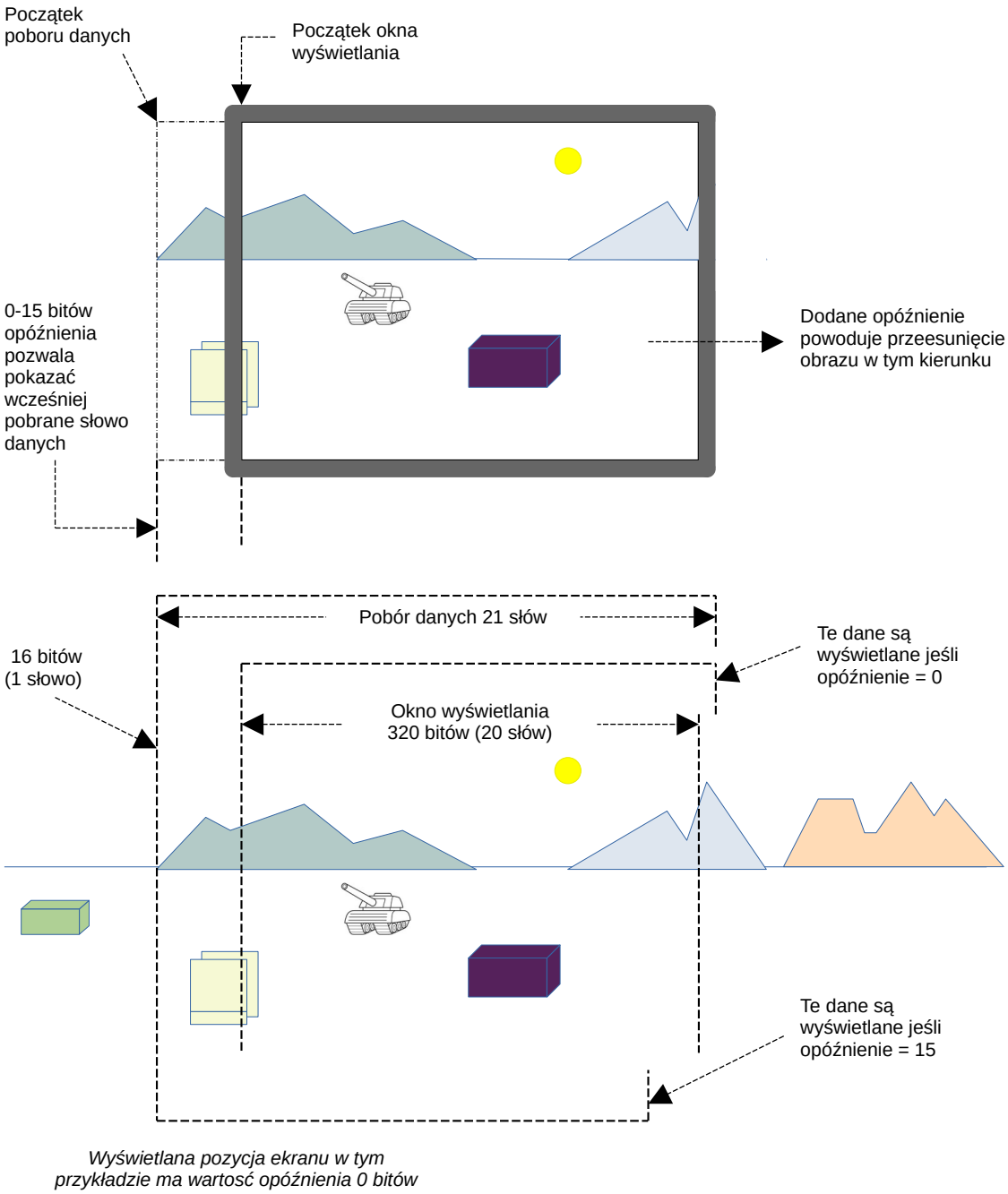
- ustawić bity opóźnienia
- ustawić modulo tak, żeby wkaźniki bitplanów ustawiły się w poprawnej pozycji dla każdej linii
- napisać program Coppera do obsługi zmian wskaźników w czasie wygaszenia pionowego.

Wyznaczenie poboru danych dla przesuwania poziomego

Standardowy pobór danych dla ekranów statycznych ustala się wartością $38. Jeśli chcemy użyć przesunięcia w poziomie, pobór danych musi rozpocząć się słowo wcześniej (DDFSTRT = $0030). Niestety w ten sposób blokujemy możliwość użycia siódmego sprite'a. DDFSTOP nie zmiena się. Pamiętać należy, że ustawienie poboru danych ma wpływ na oba playfieldy.

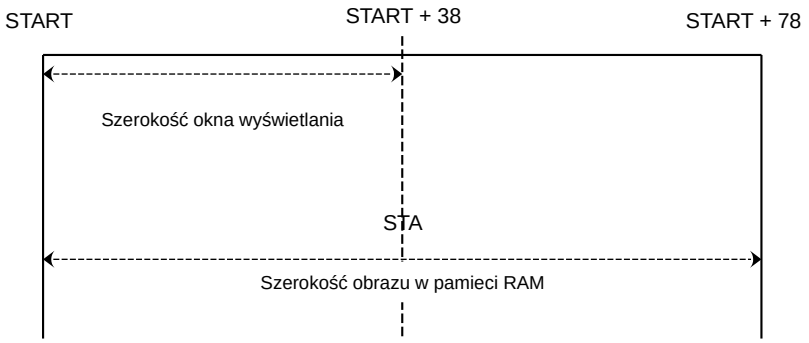
Wyznaczenie modulo dla przesuwania poziomego

Jak zawsze, modulo jest o dwie wartości mniejsze niż odległość pomiędzy adresem następnego słowa jakie chcemy pobrać a adresem ostatniego pobranego słowa. Dla przykładu załóżmy, że mamy wyświetlaną linię zawierająco 40 bajtów z większego obrazka zawierającego 80 bajtów w linii. Ponieważ pobór danych jest większy o dwa bajty, ilość danych dla każdej linii będzie wynosić 42 bajty.



Rys. 3-24: Przesuwanie poziome

UWAGA: Pobranie dodatkowych danych zablokuje użycie niektórych sprite’ów



Rys. 3-25: Obraz w pamięci większy niż okno wyświetlania

Dane linii 1:

Lokacja:	**START**	**START+2**	**START+4**	**. . .**	**START+40**
	skraje lewe słowo danych do wyświetlenia	następne słowo	następne słowo		ostatnie słowo

Rys. 3-26: Dane pierwszej linii – przesuwanie w poziomie

Po pobraniu pierwszej linii rejestry wskaźników bitplanów przechowują wartość START+42. Wartość modulo (38) zwiększa wskaźniki dzięki czemu wkazują poprawnie na kolejną linię.

Dane linii 2:

Lokacja:	**START+80**	**START+82**	**START+84**	**. . .**	**START+120**
	skraje lewe słowo danych do wyświetlenia	następne słowo	następne słowo		ostatnie słowo

Rys. 3-27: Dane drugiej linii – przesuwanie w poziomie

Modulo ustawia się w rejestrach BPLxMOD odpowiednio dla nieparzystych i parzystych bitplanów.

Określenie wielkości opóźnienia

Wielkość opóźnienia dla przesuwu poziomego ustawia się na bitach 7-0 rejestru BPLCON1. Wartości opóźnienia ustawia się osobno dla każdego playfieldu. Bity 3-0 odpowiadają wartości opóźnienia dla playfieldu 1 (bitplany 1, 3 i 5) natomiast dla playfieldu 2 (bitplany 2, 4 i 6) korzysta się z bitóe 7-4.

> *UWAGA.* Należy zawsze ustawiać obie grupy bitów, nawet jeśli używa się jednego playfieldu. W takim przypadku bity 3-0 i 7-4 zapisuje się tymi samymi wartościami.

Poniższy przykład ustawia opóźnienie wartością 7 dla obu playfieldów:

```
MOVE.W #$77,BPLCON1+CUSTOM
```

PODSUMOWANIE PRZEMIESZCZANIA PLAYFIELDU

Kroki do zdefiniowania ruchomego playfieldu są identyczne z krokami dla podstawowego playfieldu. Różnice wyliczono poniżej:

- **Określ pobór danych**. Pobierz jedno słowo (dwa bajty) więcej dla każdej linii rozpoczynając 16 pikseli przed norlamnym (nieruchomym) poborem danych.
- **Określ modulo**. Modulo jest o dwa mniejsze niż w przypadku bez przesuwania obrazu.

Poniższe kroki są dodatkowe:

- **Dla przesuwu pionowego zmieniaj wkaźniki bitplanów o ilość jednorazowo przesuwanych linii**. Ustawiaj rejestry BPLxPTH i BPLxPTL w czasie wygaszenia pionowego.
- **Przesuwając w poziomie określ opóźnienie**. Ustaw bity 7-0 w rejestrze BPLCON1 wartościami 0-15.

Techniki zaawansowane

Ta sekcja opisuje zagadnienia żadziej używane lub opcjonalne.

WZAJEMNE ODDZIAŁYWANIA PLAYFIELDÓW Z INNYMI OBIEKTAMI

Plafieldy współdzielą ekran zze sprite’ami. Rozdział 7, „Układy kontroli systemu” opisuje w jaki sposób można nadać playfieldom różne priorytety w oddziaływaniach ze sprite’ami oraz w jaki sposób playfieldy mogą kolidować (nachodzić na) ze sprite’ami lub ze sobą.

TRYB HOLD-AND-MODIFY

To specjalny tryb, który pozwala na wyświetlenie na ekranie nawet 4096 kolorów jednocześnie. Standardowo, wartości, które definiują poszczególne punkty w bitplanach, określają numer rejestru koloru z jakiego kod koloru zostanie przesłany do układu generującego obraz powodując zabarwienie odpowiedniego punktu na ekranie właściwym kolorem.

Jednak w trybie hold-and-modify (HAM) wartość w obwodzie wyjściowym koloru jest wstrzymywana i jeden z trzech składników koloru (czerwony, zielony lib niebieski) jest zmieniany bitami przychodzącymi z odpowiednio wybranych bitplanów. Po modyfikacji piksel jest wyświetlony na ekranie.

Tryb hold-and-modify pozwala na uzyskanie bardzo płynnych przejść tonalnych (gradientów) lub płynnego cieniowania. Dla przykładu weźmy obrazek, na którym jest 16 wazonów, każdy w innym kolorze spośród 16 dostępnych w palecie. Używając trybu HAM, na każdym wazonie można uzyskać efekty rozjaśnienia lub przyciemnienia, lub nawet dodać zupełnie inny kolor do każdego z nich. Należy jednak zwrócić uwagę, że dla każdego punktu w trybie HAM można zmienić tylko jedną z trzech składowych koloru. Daje to więc ograniczoną kontrolę.

W trybie hold-and-modify używa się wszystkich sześciu bitplanów. Bitplany 5 i 6 są używane do zmiany sposobu interpretacji bitów z bitplanów 1-4 w następujący sposób:

- Jeśli bity 6 i 5 mają wartość 00, wybór koloru odbywa się w standardowy sposób. Czyli bity z bitplanów 1-4 określają numer rejestru koloru (od 0 do 15) jaki zostanie użyty do pokolorowania danego punktu na ekranie.

Jeśli w użyciu jest tylko 5 bitplanów, w domyśle uważa się, że bity z bitplanu 6 mają wartość 0.

- Jeśli bity 6 i 5 dają wartość 01, kolor punktu po lewej stronie bierzącego piksela jest powielany i zmieniany. Bity z bitplanów 1 – 4 zostają użyte do zastąpienia czterech bitów składowej niebieskiej w wybranym rejestrze koloru.
- Jeśli bity 6 i 5 dają wartość 10, kolor punktu po lewej stronie bierzącego piksela jest powielany i zmieniany. Bity z bitplanów 1 – 4 zostają użyte do zastąpienia czterech bitów składowej czerwonej w wybranym rejestrze koloru.
- Jeśli bity 6 i 5 dają wartość 11, kolor punktu po lewej stronie bierzącego piksela jest powielany i zmieniany. Bity z bitplanów 1 – 4 zostają użyte do zastąpienia czterech bitów składowej zielonej w wybranym rejestrze koloru.

Używając trybu HAM możliwym jest użycie jednokolorowego obrazu korzystającego tylko z rejestru koloru tła (COLOR00) i uzyskanie wielokolorowego obrazu przez zmodyfikowanie wszystkich punktów na ekranie w sposób podany powyżej.

Do włączenia trybu HAM służy bit 11 rejestru BPLCON0. Dodatkowo poniższe bity muszą przyjąć następujące wartości:

- bit 11 (HOMOD) = 1
- bit 10 (DBLPF) = 0 (tylko pojedynczy playfield)
- bit 15 (HIRES) = 0 (tylko niska rozdzielczość)
- bity 14 – 12 (BPU2 – BPU0) = 101 lub 110 (pięć lub sześć bitplanów)

Poniższy kod źródłowy włącza sześciobitplanowy widok w trybie HAM. Wszystkie 32 rejestry kolorów są zainicjalizowane kolorem czarnym aby pokazać, że kolory są generowane przez algorytmy hold-and-modify.

```
; First, set up the control registers.
;
       LEA CUSTOM,a0              ; a0 wskazuje na układy specjalizowane
       MOVE.W #$6AO 0,BPLCON0(a0); sześć bitplanów, tryb hold-and-modify
       MOVE.W #0,BPLCON1(a0)      ; przesunięcie poziome = 0
       MOVE.W #0,BPL1MOD(a0)      ; modulo parzystych bitplanów = 0
       MOVE.W #0,BPL2MOD(a0)      ; to samo dla parzystych
       MOVE.W #$0038,DDFSTRT(a0)  ; początek data-fetch
       MOVE.W #$00D0,DDFSTOP(a0)  ; koniec data-fetch
       MOVE.W #$2C81,DIWSTRT(a0)  ; początek okna wyświetlania
       MOVE.W #$F4C1,DIWSTOP(a0)  ; koniec okna wyświetlania
;
; Zapisz wszystkie rejestry kolorów kolorem czarnym
;
       MOVE.W #32,d0              ; inicjalizuj licznik
       LEA CUSTOM+COLOR00,a1      ; wkaż pierwszy rejestr
CREGLOOP:
       MOVE.W #$0000,(a1)+        ; zapisz kolor czarny w rejestrze
       DBRA d0,CREGLOOP           ; zmniejsz licznik i powtarzaj pętlę do końca
;
; Wypełnij wszystkie 6 bitplanów wzorem łatwym do rozpoznania.
; UWAGA: Kod użyty jako przykład. W rzeczywistym programie pamięć dla bitplanów
; musi zostać zarezerowowana w obszarze MEMF_CHIP.
;
       MOVE.W #2000,d0            ; 2000 długich słów na bitplan
       MOVE.L #$21000,a1          ; wskaźnik bitplanu 1 w a1
       MOVE.L #$23000,a2          ; wskaźnik bitplanu 2 w a2
       MOVE.L #$25000,a3          ; wskaźnik bitplanu 3 w a3
       MOVE.L #$27000,a4          ; wskaźnik bitplanu 4 w a4
       MOVE.L #$29000,a5          ; wskaźnik bitplanu 5 w a5
       MOVE.L #$2B000,a6          ; wskaźnik bitplanu 6 w a6
       MOVE.L #$55555555,(a1)+    ; zapełnij bitplan 1 $55555555
```

```
       MOVE.L #$33333333,(a2)+   ; zapełnij bitplan 2 $33333333
       MOVE.L #$0F0F0F0F,(a3)+   ; zapełnij bitplan 3 $0F0F0F0F
       MOVE.L #$00FF00FF,(a4)+   ; zapełnij bitplan 4 $00FF00FF
       MOVE.L #$CF3CF3CF,(a5)+   ; zapełnij bitplan 5 $CF3CF3CF
       MOVE.L #$3CF3CF3C,(a6)+   ; zapełnij bitplan 6 $3CF3CF3C
       DBF d0,FPLLOOP            ; zmniejsz licznik i kontynuuj jeśli nie koniec
;
; Ustaw copperlistę pod adresem $20000.
; UWAGA: Podobnie jak w przypadku bitplanów, to tylko przykład. W rzeczywistości
; pamięć do tego celu powinna zostać zarezerwowana w obszarze MEMF_CHIP
;
       MOVE.L #$20000,a1         ; początek copperlisty w a1
       LEA COPPERL(pc),a2        ; początek bitplanów w pamięci w a2
CLOOP:
       MOVE.L (a2),(a1)+         ; kopiuj długie słowo
       CMPI.L #$FFFFFFFE,(a2)+   ; sprawdź czy koniec copperlisty
       BNE CLOOP                 ; wykonuj pętlę dla całek copperlisty
;
; wskaż Copperowi jego nowy program
;
       MOVE.L #$20000,COP1LCH(a0)       ; załaduj początek copperlisty do rejestru
       MOVE.W COPJMP1(a0),d0            ; wymuś start copperlisty
;
; Start DMA
;
       MOVE.W #$8380,DMACON(a0)  ; włącz DMA dla bitplanów i coppera
       BRA ....dalszy ciąg programu....
;
; Copperlista dla sześciu bitplanów. Bitplan 1 - $21000; 2 - $23000;
; 3 - $25000; 4 - $27000; 5 - $29000; 6 - $2B000.
;
; UWAGA: Adresy bitplanów są przykładowe. Zobacz uwagi powyżej.
;
COPPERL:
       DC.W BPL1PTH,$0002  ; Bitplane 1 pointer = $21000
       DC.W BPL1PTL,$1000
       DC.W BPL2PTH,$0002  ; Bitplane 2 pointer = $23000
       DC.W BPL2PTL,$3000
       DC.W BPL3PTH,$0002  ; Bitplane 3 pointer = $25000
       DC.W BPL3PTL,$5000
       DC.W BPL4PTH,$0002  ; Bitplane 4 pointer = $27000
       DC.W BPL4PTL,$7000
       DC.W BPL5PTH,$0002  ; Bitplane 5 pointer = $29000
       DC.W BPL5PTL,$9000
       DC.W BPL6PTH,$0002  ; Bitplane 6 pointer = $2B000
       DC.W BPL6PTL,$B000
       DC.W $FFFF,$FFFE    ; oczekuj niemożliwego, czyli koniec copperlisty
```

TWORZENIE EKRANU Z KIKOMA RÓŻNYMI PLAYFIELDAMI

Biblioteka graficzna umożliwia podział ekranu na kilka tze. „ViewPortów”, każdy z własną rozdzielczością i paletą kolorów. Więcej informacji na ten temat znajdziesz w książce „*Amiga ROM Kernel Manual: Biblioteki*”.

UŻYCIE ZEWNĘTRZNEGO ŹRÓDŁA OBRAZU

Amiga została zaprojektowana z myślą o prostym użyciu *genlocków*. Genlock to urządzenie, dzięki któremu można wyświetlać obraz z zewnętrznego źródła, takiego jak magnetowid czy kamera wideo. Używając genlocka kolor tła zostaje zastąpiony obrazem z tych źródeł. Więcej informacji na ten temat powinny dostarczać instrukcje załączone do genlocków.

Podsumowanie informacji o rejestrach playfieldów

W sekcji zostały zebrane razem i opisane krótko razem ze znaczeniem ich bitów wszystkie rejestry jakie pojawiły się w rozdziale. Następna sekcja podsumowuje rejestry kolorów.

BPLCON0 – kontrola bitplanów

(Uwaga: bitów w tym rejestrze nie można ustawiać niezależnie)

Bit 0 – nie używany

Bit 1 – ERSY (synchronizacja z zewnętrznym sygnałem)
1 = zewnętrzna synchronizacja włączona (pozwala na synchronizację z genlockiem)
0 = zewnętrzna synchronizacja wyłączona

Bit 2 – LACE (tryb z przeplotem)
1 = przeplot włączony
0 = przeplot wyłączony

Bit 3 – LPEN (obsługa pióra świetlnego, 1 = włączona, 0 = wyłączona)

Bit 4 – 7 nie używane, wpisz 0

Bit 8 – GAUD (obsługa dźwięku z genlocka, 1 = włączona, 0 = wyłączona)
(bit ten pojawia się również na pinie ZD układu Denise w czasie wygaszenia)

Bit 9 – COLOR_ON (kolor na wyjściu kompoytowym, tylko Amiga 1000)
1 = color-burst włączony
0 = color-burst wyłączony

Bit 10 – DBLPF (podwójny playfield, 1 = podwójny playfield, 0 = pojedynczy playfield)

Bit 11 – HOMOD (tryb hold-and-modify)
1 = HAM włączony
0 = HAM wyłączony; włączony tryb extra-half brite (EHB) jeśli DBLPF=0 i BPUx=6

Bity 14, 13, 12 – BPU2, BPU1, BPU0
Ilość użytych bitplanów

000 = tylko kolor tła
001 = 1 bitplan
010 = 2 bitplany
011 = 3 bitplany
100 = 4 bitplany
101 = 5 bitplanów
110 = 6 bitplanów
111 wartość nie używana

Bit 15 – HIRES (wysoka rozdzielczość)
1 = włączona wysoka rozdzielczość
0 = wyłączona (włączona niska rozdzielczość)

BPLCON1 – kontrola bitplanów

Bity 3 – 0 – PF1H(3-0) opóźnienie dla playfieldu 1

Bity 7 – 4 – PF2H(3-0) opóśnienie dla playfieldu 2

Bity 15 – 8 nie używane

BPLCON2 – kontrola bitplanów

Bit 6 – PF2PRI
1 = priorytet dla bitplanu 2
0 = priorytet dla bitplanu 1

Bity 0 – 5 – priorytety playfieldów i prite'ów

Bity 7 – 15 nie używane

DDFSTRT – początek poboru danych
(pozycja w linii rozpoczynająca pobór danych)

Bity 15 – 8 nie używane

Bity 7 – 2 – położenie punktu H8-H3 (bit H3 brany pod uwagę w wysokiej rozdzielczości)

Bity 1 – 0 nie używane

DDFSTOP – koniec poboru danych
(pozycja w linii kończąca pobór danych)

Bity 15 – 8 nie używane

Bity 7 – 2 – położenie punktu H8-H3 (bit H3 brany pod uwagę w wysokiej rozdzielczości)

Bity 1 – 0 nie używane

BPLxPTH – wskaźnik bitplanu
(bardziej znaczące słowo adresu początku bitplanu numer „x")

BPLxPTL – wskaźnik bitplanu
(mniej znaczące słowo adresu początku bitplanu numer „x")

DIWSTRT – początek okna wyświetlania
(współrzędne początkowe okna wyświetlania)

Bity 15 – 8 – VSTART (V7-V0)

Bity 7 – 0 – HSTART (H7-H0)

DIWSTOP – koniec okna wyświetlania
(współrzędne końcowe okna wyświetlania)

Bity 15 – 8 – VSTOP (V7-V0)

Bity 7 – 0 – HSTOP (H7-H0)

BPL1MOD – modulo dla bitplanów
(nieparzyste bitplany, playfield 1)

BPL2MOD – modulo dla bitplanów
(parzyste bitplany, playfield 2)

Podsumowanie informacji o rejestrach wyboru koloru

Ta sekcja zawiera podsumowanie informacji o rejestrach wyboru koloru dla playfieldu. Zebrane tu informacje obejmują zawartości tych rejestrów, przykładowe kolory oraz różnice wynikające z użycia wysokiej rozdzielczości w porównaiu do niskiej. Amiga posiada 32 rejestry kolorów. Każdy rejestr ma po 4 bity odpowiadające za każdą składową koloru: R – czerwony, G – zielony i B – niebieski. Tabelka poniżej pokazuje przypisanie bitów do odpowiednich składowych. Wszystkie rejestry kolorów są tylko do zapisu.

Bity rejestru koloru	Zawartość
15 – 12	nie używane
11 – 8	czerwony
7 – 4	zielony
3 – 0	niebieski

Tabela 3-15: Zawartość pojedynczego rejestru koloru

PRZYKŁADOWE KOLORY

Tabela 3-16 prezentuje przykładowe kolory i ich szesnastkową wartość, którą można zapisać bezpośrednio w wybranym rejestrze koloru.

Wartość	Kolor	Wartość	Kolor
$FFF	biały	$1FB	jasno-morski
$D00	ceglasty	$6FE	błękitny
$F00	czerwony	$6CE	jasno-niebieski
$F80	czerono-pomarańczowy	$00F	niebieski
$F90	pomarańczowy	$61F	rozświetlony niebieski
$FB0	pomarańczowo-złoty	$06D	ciemno-niebieski
$FD0	żółć kadmu	$91F	purpurowy
$FF0	cytrynowy	$C1F	fioletowy
$BF0	limonkowy	$F1F	magenta
$8E0	jasna zieleń	$FAC	różowy
$0F0	zielony	$DB9	złoty
$2C0	ciemna zieleń	$C80	brązowy
$0B1	leśna zieleń	$A87	ciemno-brązowy
$0BB	niebiesko-zielony	$CCC	jasno-szary
$0DB	morski	$999	średni szary
		$000	czarny

Tabela 3-16: Wybrane wartości kolorów i ich słowne opisy

WYBÓR KOLORÓW W NISKIEJ ROZDZIELCZOŚCI

Poniższa tabela prezentuje sposób wyboru kolory w trybie niskiej rozdzielczości. Numer rejestru koloru jest określany przez liczbę dwójkową złożoną z bitów odczytanych z wszystkich używanych bitplanów dla pojedynczego piksela.

Pojedynczy playfield		**Podwójny playfield**	**Numer rejestru koloru**
Normalny tryb (bitplany 5, 4, 3, 2, 1)	Tryb HAM (bitplany 4, 3, 2, 1)		
		Playfield 1 (bitplany 5, 3, 1)	
00000	0000	000**	0*
00001	0001	001	1
00010	0010	010	2
00011	0011	011	3
00100	0100	100	4
00101	0101	101	5
00110	0110	110	6
00111	0111	111	7
		Playfield 2 (bitplany 6, 4, 2)	
01000	1000	000**	8
01001	1001	001	9
01010	1010	010	10
01011	1011	011	11
01100	1100	100	12
01101	1101	101	13
01110	1110	110	14
01111	1111	111	15
10000	Nie używane w tym trybie	Nie używane w tym trybie	16
10001			17
10010			18
10011			19
10100			20
10101			21
10110			22
10111			23
11000			24
11001			25
11010			26
11011			27
11100			28
11101			29
11110			30
11111			31

* Rejestr koloru 0 zawsze określa kolor tła

** Oznacza punkt przezroczysty, numer rejestru koloru ignorowany

Tabela 3-17: Wybór koloru w niskiej rozdzielczości

WYBÓR KOLORU W TRYBIE WYSOKIEJ ROZDZIELCZOŚCI

Poniższa tabela prezentuje sposób wyboru kolory w trybie wysokiej rozdzielczości. Numer rejestru koloru jest określany przez liczbę dwójkową złożoną z bitów odczytanych z wszystkich używanych bitplanów dla pojedynczego piksela.

Pojedynczy playfield (bitplany 4, 3, 2, 1)	**Podwójny playfield**	**Numer rejestru koloru**
	Playfield 1 (bitplany 3, 1)	
0000	00*	0**
0001	01	1
0010	10	2
0011	11	3
0100	Nie używane w tym trybie	4
0101		5
0110		6
0111		7
	Playfield 2 (bitplany 4, 2)	
1000	00*	8
1001	01	9
1010	10	10
1011	11	11
1100	Nie używane w tym trybie	12
1101		13
1110		14
1111		15

* Oznacza punkt przezroczysty, wyznaczony numer rejestru koloru ignorowany

** Rejestr koloru 0 zawsze określa kolor tła

Tabela 3-18: Wybór koloru w wysokiej rozdzielczości

WYBÓR KOLORU W TRYBIE HOLD-AND-MODIFY

W trybie hold-and-modify zawartość rejestru koloru jest zmieniana zgodnie z zasadami przedstawionymi w tabelce 3-19. Tryb ten jest aktywny tylko jeśli ustawiny bit 10 rejestru BPLCON0 (ustawimy go wartością 1).

Bitplan 6	**Bitplan 5**	**Rezultat**	
0	0	Standardowe działanie (wybierz rejestr koloru)	
0	1	zielony i czerwony zostaje	niebieski weź z bitplanów 4-1
1	0	zielony i niebieski zostaje	czerwony weź z bitplanów 4-1
1	1	niebieski i czerwony zostaje	zielony weź z bitplanów 4-1

Tabela 3-19: Wybór koloru w trybie HAM

WYBÓR KOLORU W TRYBIE EXTRA-HALF BRITE (EHB)

Amiga posiada specjalny tryb nazywany Extra-Half Brite (EHB) który podwaja maksymalną ilość kolorów wyświetlaną jednocześnie. Do użycia trybu EHB potrzeba sześciu bitplanów więc należy ustawić bity BPU=6 w rejestrze BPLCON0. Bity 11 (HAM) i 10 (DPF) w tym rejestrze czyścimy (ustawiamy na 0). W tym trybie bity z szóstego bitplanu kontrolują intensywność kolorów opsanych na bitplanach 5 – 1. W wartości odczytanej z rejestru koloru dla danego piksela, zależnie od ustawienia bitu z bitplanu 6 jest zmniejszana intensywność o połowę. Pozwala to na uzyskanie do 64 kolorów zamiast standardowych 32.

> *Rejestry playfieldu w układach ECS*. Informacje dotyczące użycia playfieldów z użyciem układów ECS opisano w dodatku C.