rozdział czwarty

SPRITE'Y

W tym rozdziale omówimy sprite'y[1] – łatwe do zdefiniowania i animowania obiekty graficzne. Rozdział opisuje poniższe zagadnienia:

- Definiowanie rozmiaru, kształtu, koloru i pozycji sprite'a na ekranie.
- Wyświetlanie i poruszanie sprite'ów.
- Łączenie sprite'ów w celu osiągnięcia dodatkowej szerokości lub ilości kolorów.
- Wielokrotne ponowne użycie kanału DMA sprite'a w ramach jednej klatki obrazu w celu wyświetlenia więcej niz ośmiu obiektów na ekranie jednocześnie.

Czym są sprite'y?

Sprite'w to obiekty graficzne, które są niezależne od pozostałej grafiki na ekranie. Poruszanie playfieldu nie ma wpływu na sprite'y, tak samo zmiana pozycji sprite'ów na ekranie nie ma wpływu na pozostałą grafikę. Razem z playfieldami, sprite'y tworzą kompletny obraz graficzny na ekranie Amigi. Dzięki blitterowi możemy do tego zestawu dorzucić jeszcze dosć skomplikowane animacje, co postaramy się opisać w rozdziale „Blitter". Sprite'y pojawiają się na ekranie dzięki ośmiu kanałom DMA. Standardowo, sprite ma szerokość 16 punktów i dowolną wysokość. Każdy sprite może mieć trzy kolory oraz dodatkowy kolor „przeźroczysty", dzięki któremu można zobaczyć obiekty „pod" spritem. Sprite'y można łączyć ze sobą aby uzyskać większe, bardziej skomplikowane i bardziej kolorowe obiekty.

Kanały DMA sprite'ów można kilkukrotnie użyć w czasie wyświetlania jednej klatki obrazu. W ten sposób można obejść ograniczenie jednoczesnego wyświetlenia maksymalnie ośmiu sprite'ów.

Definiowanie sprite'a

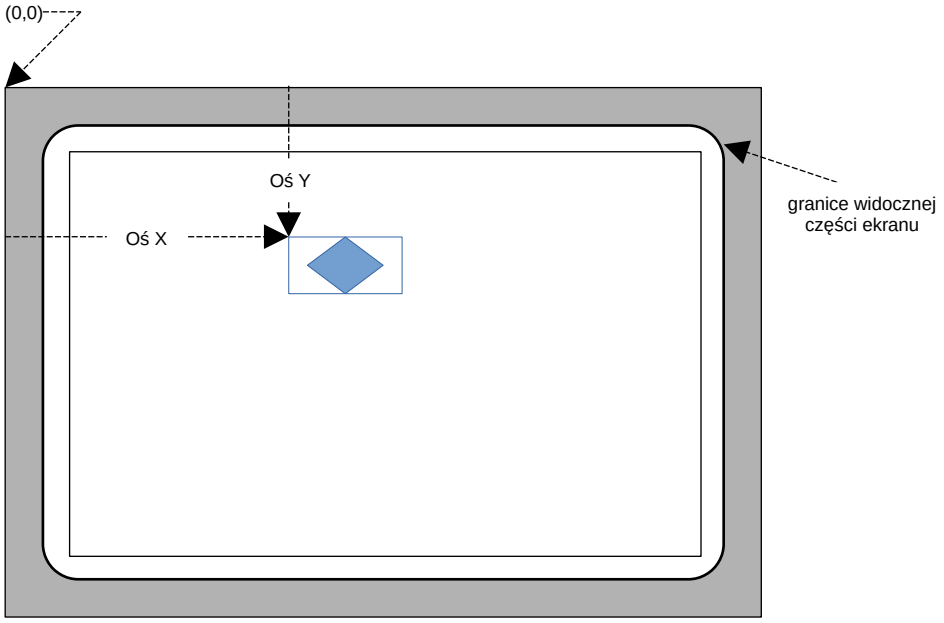
Aby zdefiniować sprite'a, najpierw trzeba zdefiniować odpowiednią strukturę w pamięci. Aby to zrobić, określa się następujace cechy:

- Szerokość do 16 punktów.
- Nieograniczona wysokość.
- Dowolny kształt.
- Trzy kolory plus przeźroczystość.
- Dowolna pozycja na ekranie.

1 (przypis tłumacza) Nie będę silił się na tłumaczenie tego słowa na język polski. W literaturze, w latach 80-tych i 90-tych, zwłaszca dotyczącej komputerów 8-bitowych, często pojawiało się określenie „duszki". Cóż, może jest to ładny polski odpowiednik, jednak mi do gustu nie przypadł.

POZYCJA NA EKRANIE

Pozycja sprite’a na ekranie jest opisywana współrzędnymi (X, Y). Pozycja (0, 0), gdzie X=0 i Y=0 to lewy górny róg ekranu. Pozycję sprite’a określa się przez współrzędne jego lewego górnego puktu. Pozycja sprite’a zawsze jest określana względem obrazu o niskiej rozdzielczości bez przeplotu. Sposób określania współrzędnych pozycji sprite’a przedstawiono na ryzunku 4-1 poniżej. Warto zauważyć, że pozycja (0, 0) nie jest możliwa do wyświetlenia na standardowym ekranie (znajduje się poza widocznym obszarem ekranu).



Rys. 4-1: Określanie pozycji sprite’a na ekranie

Na wielkość widocznego obszaru ekranu wpływ mają ustawienia rozmiaru okna wyświetlania playfieldu (zapisane w rejestrach DDFSTRT, DDFSTOP, DIWSTRT, DIWSTOP, itp.). Więcej informacji znaduje się w rozdziale czwartym - „Playfieldy”.

Pozycja pozioma

Pozioma pzycja sprite’a (wartość X) może przyjąć wartość każdego piksela w zakresie od 0 do 447. Aby jednak sprite był widoczny, musi zmieścić się w granicach określonych przez okno wyświetlania. W przykładach przedstawionych w tym rozdziale przyjęto, że okno wyświetlania rozpoczyna się w punkcie 64 a kończy punktem 383 (dając linię o długości 320 punktów). Oczywiście można zdefiniować większe lub mniejsze okna wyświetlania, zależnie od potrzeb. Zalecamy zapoznanie się z rozdziałem „Playfieldy” przed przystąpieniem do definicji tego okna. Mimo iż wiązka elektronów „omiata” większą powierzchnię, część z niej jednak pozostanie niewidoczna.

Część lub cały sprite może stać się niewidoczny, jeśli zdefiniujesz pozycję X sprite’a znajdującą się poza widzialną częścią ekranu. Czasami jest to pożądane; o takim obiekcie mówimy, że jest „obcięty” (ang. *clipped*).

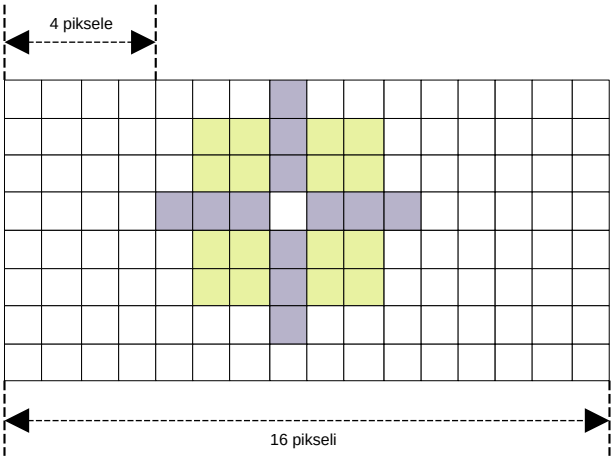
Aby sprite był widoczny na ekranie wystarczy do jego współrzędnej dodać offset w postaci poziomej pozycji początkowej okna wyświetlania. Biorąc pod uwagę co napisaliśmy w pierwszym akapicie tej sekcji, do pozycji X sprite’a trzeba dodać wartość 64. Przykładowo, jeśli chcemy wyświetlić sprite’a w odległości 94 punktów od lewej krawędzi ekranu, musimy wykonać poniższe działanie:

Pożądana pozycja X + poziomy offset okna wyświetlania = 94 + 64 = 158

Stąd pozycja 158 staje się wartością X, którą zapiszemy do struktury danych sprite'a.

> *Liczenie pikseli*. Pozycja X określa położenie najbardziej skrajnego (lewego) punktu sprite'a o pełnej, 16-punktowej szerokości. Trzeba to zawsze mieć na uwadze, zwłaszcza, jeśli skrajne punkty sprite'a zdefiniowano jako przeźroczyste.

Jeśli sprite pokazany na ryzunku 4-2 jest umiejscowiony na pozycji X równej 158, widoczna część sprite'a na ekranie zacznie się od punktu 162. Powodem tego są pierwsze cztery punkty sprite'a, które w tym przypadku zdefiniowano jako przeźroczyste.



Rys. 4-2: Pozycja sprite'a

Pozycja pionowa

Jako pozycję górnej krawędzi sprite'a można wybrać dowolną linię od 0 do 262. Przykłady z tego rozdziału używają okna w trybie NTSC umiejscowionego pionowo pomiędzy 44 a 243 linią ekranu. Daje to w wyniku okno o wysokości 200 linii w trybie bez przeplotu. Jeśli zdefiniujesz pozycję pionową mniejszą niż 44 (ponad górną krawędzią okna wyświetlania) górna część sprite'a może wyjść poza widoczną część obrazu.

Aby sprite był w całości widoczny na ekranie do zadanej pozycji ekranowej Y trzeba dodać offset okna wyświetlania (z akapitu powyżej – 44). Dla przykładu, aby górna krawędź sprite'a pojawiła się 25 linii poniżej górnej krawędzi widocznego obszaru na ekranie, należy wykonać następujące działanie:

Pożądana pozycja Y + pionowy offset okna wyświetlania = 25 + 44= 69

Mamy więc współrzędną Y naszego sprite'a, którą zapisujemy w strukturze danych tego obiektu.

Obcięte sprite'y

Jak wspomniano wyżej, sprite'y będą częściowo lub całkowicie obcięte, jeśli wyjdą poza zakres widocznej części ekranu. Pozycje 64 (poziomo) i 44 (pionowo) są standardowymi granicami okna wyświetlania dla monitorów lib telewizorów pracujących w trybie NTSC. „Rozdział trzeci, Playfieldy" dostarczy więcej informacji ogranicach widocznej części ekranu. Tam też znajdują się informacje o ekranach pracujących w trybie PAL. Jeśli wybierzesz inne wartości prze definiowaniu okna wyświetlania, sprite'y będą obcinane zgodnie z wybranymi przez Ciebie wartościami.

ROZMIARY SPRITE'ÓW

Sprite'y mają szerokość 16 pikseli i prawie dowolną wysokość – mogą mieć jedną linię wysokości lub też mogą być wyższe niż obraz wyświetlany na ekranie. W przypadku sprite'a większego niż ekran, tylko jego fragment będzie widoczny na ekranie, zależnie od przesunięcia jakie zdafiniujesz.

Rozmiar sprite'a bazuje na punktach, z których każdy ma rozmiar 1/320 szerokości ekranu i 1/200 wysokości ekranu w systemie NTSC lub 1/256 w PALu. Rozmiar pojedynczego punktu odpowiada pikselowi w niskiej rozdzielczości bez przeplotu standardowego pełnowymiarowego playfieldu. Sprite'y jednakże są niezależne od playfieldu, zmiana rozdzielczości czy też tryb z przeplotem nie mają wpływu na rozmiar i rozdzielczość sprite'a.

KSZTAŁT SPRITE'ÓW

Prite może mieć dowolny kształt, o ile jego szerokość nie przekroczy 16 pikseli. Kształt sprite'a określa się przez wskazanie, które punkty mają być widoczne a które przeźroczyste. Rysunki poniżej przedstawiają latający spodek. Na pierwszym rysunku widzimy sam kształt sprite'a. Można go zaprojektować używając arkusza papieru milimetrowego. Drugi rysunek pokazuje jak wygląda nasz sprite naniesiony na tablicę o szerokości 16 puntów. Przy pomocy litery „X" oznaczono widoczną część sprite'a, literami „O" oznaczono przeźroczystą powierzchnię sprite'a.

```
        X X
    X X X X X X
X X X X X X X X X X
X X X X X X X X X X
    X X X X X X
        X X
```

Rys. 4-3: Kształt latającego spodka

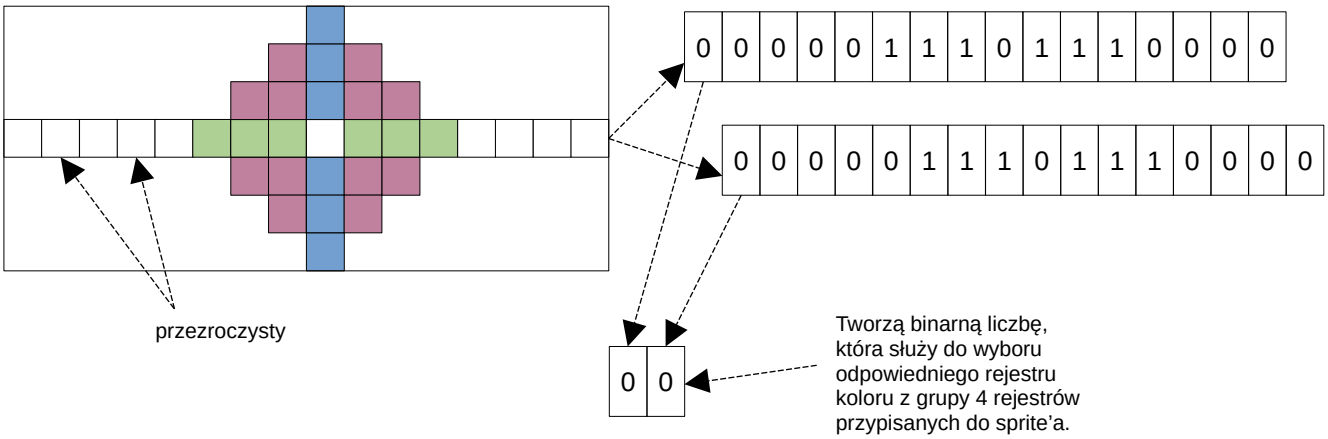
```
O O O O X X O O O O O O O O O O
O O X X X X X X O O O O O O O O
X X X X X X X X X X O O O O O O
X X X X X X X X X X O O O O O O
O O X X X X X X O O O O O O O O
O O O O X X O O O O O O O O O O
```

Rys. 4-4: Sprite ze zdefiniowanym kształtem spodka

Kształt sprite'a zdefiniowany powyżej w najszerszym miejscu ma 10 punktów szerokości i jest wyrównany do lewej strony. Jeśli tylko kształt sprite'a jest węższy niż szerokość sprite'a, masz wpływ na to, która część sprite'a jest użyta do przechowania kształtu. Kształt powyższy może równie dobrze zaczynać się od dowolnego piksela z zakresu 2 – 7.

KOLOR SPRITE’A

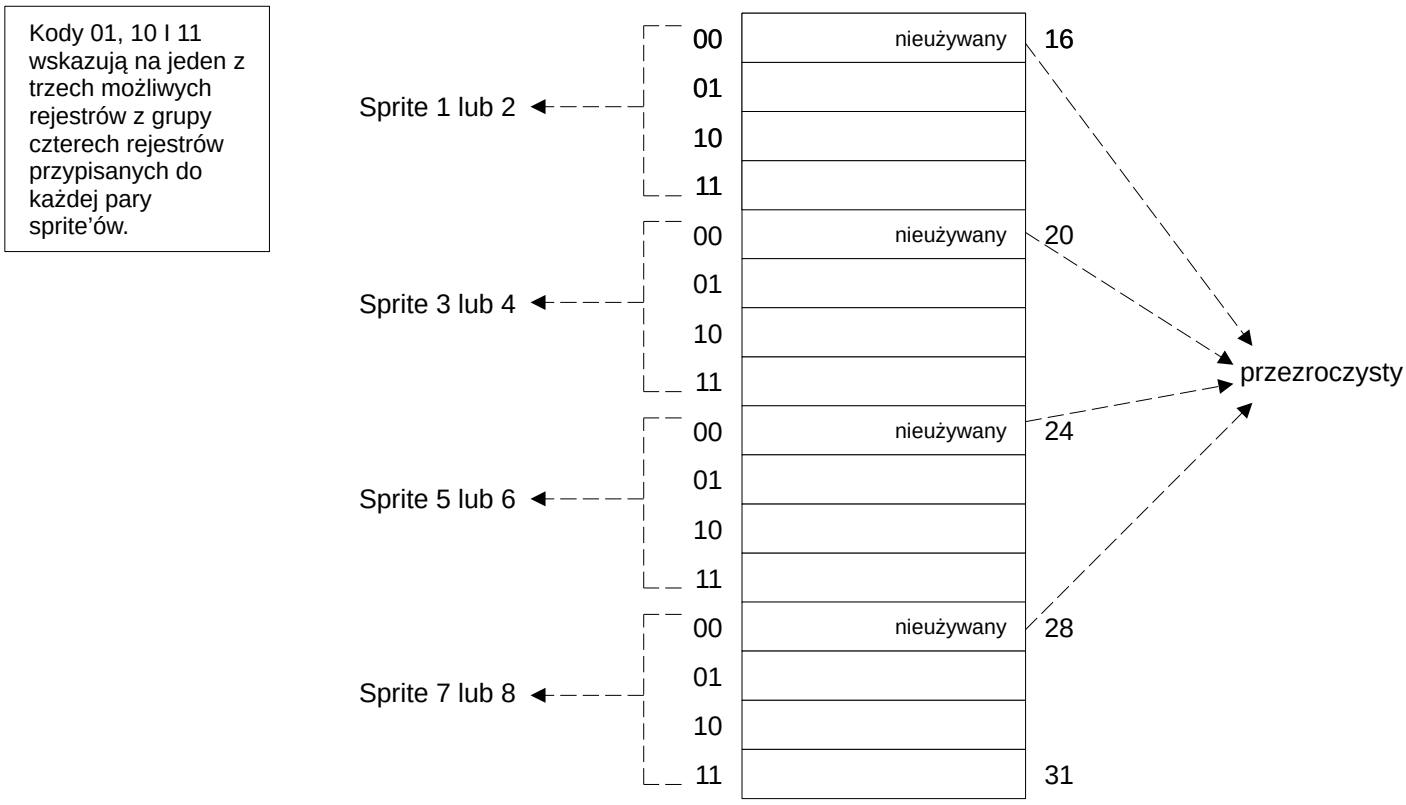
W Sprite’ach używanych osobno (to znaczy nie złączone ze sobą, jak to jest opisane w sekcji „Złączone sprite’y”), każdy punt może przyjać jeden z trrzech kolorów lub kolor przeźrozysty. Ustalenie koloru jest bardzo podobne do metody używanej w przypadku playfieldów. Na rysunku 4-5 poniżej pokazano w jaki sposób określany jest kolor każdego punktu sprite’a.



Rys. 4-5: Ustalanie kolorów sprite’a

Zera i jedynki w każdym z dwóch słów, które składają się na jedną linię sprite’a, w strukturze danych sprite’a tworzą liczbę dwójkową. Każdy taki numer wskazuje na jeden z czterech rejestrów kolorów przypisanych do poszczególnych kanałów DMA sprite’ów. 8 sprite’ów używa sumarycznie rejestrów z numerami od 16 do 31. Sprite’y są zgrupowane parami, każda para jest przypisana do określonej czwórki rejestrów tak, jak pokazano na rysunku 4-6.

> *Rejestry kolorów sprite’a*. Wartość koloru z pierwszego rejestru z każdej grupy czterech rejestrów przypisanych do sprite’a jest ignorowana. Jeśli bity sprite’a wybierają ten rejestr, kolor „przeźroczysty” jest użyty.



Rys. 4-6: Przypisanie rejestrów kolorów

Jeśli sprite wymaga konkretnych kolorów, musisz ustawić odpowiednie rejestry kolorów sprite'a określonymi wartościami. W jaki sposób ustawia się rejestry kolorów opisuje rodział „Playfieldy".

Rejestr koloru sprite'a o dwójkowym numerze 00 ma specjalne zadanie. Punkt, którego wartość jest ustawiona na 00 staje się przeźroczysty. W jego miejscu będzie widać kolor innego obiektu (sprite'a lub playfieldu) o niższym priorytecie wyświetlania, jeśli ten znajdzie się „pod spodem". Każdy sprite ma z góry ustalony priorytet wyświetlania. Kontrola priorytetu jest możliwa pomiędzy sprite'ami a playfieldami. Więcej o priorytetach sprite'ów dowiecie się z rozdziału 7, „Układy kontroli systemu".

PROJEKT SPRITE'A

Aby ułatwić sobie zaprojektowanie sprite'a można użyć kartki papieru. Do odwzorowania koloru można użyć cyfr od 0 do 3. Poniżej przedstawiamy projekt latającego spodka użytego wcześniej.

```
0000122332210000
0001223333221000
0012223333222100
0001223333221000
0000122332210000
```

W następnym kroku konwertujemy cyfry 0 – 3 do numerów dwójkowych, których użyjemy do zakodowania słów opisu koloru w strukturze danych sprite'a. Poniższa sekcja opisuje jak to zrobić.

BUDOWA STRUKTURY DANYCH

Po zaprojektowaniu sprite’a musisz stworzyć jego strukturę danych. Ma ona postać serii 16-bitowych słów umieszczonych w ciągłym obszarze pamięci. Niektóre z tych słów zawierają pozycję oraz informacje kontrolne a inne odpowiadają za opis kolorów. Aby utworzyć strukturę danych sprite’a musisz:

- w pierwszym słowie określić poziomą i pionową pozycję sprite’a
- w drugim słowie określić pionową pozycję końcową sprite’a
- przekształcić dziesiętne cyfry od 0 do 3 użytych w siatce projektu sprite’a w dwójkowy numer rejestru koloru. Wartości te zebrane w słowa zapisujemy na kolejnych pozycjach struktury danych.
- Jako ostatnie w strukturze danych są słowa kontrolne określające jej koniec.

  Uwaga: Dane sprite’a, podobnie jak wszystkie inne dane, do których dostęp będą miały układy specjalizowane, muszą być umieszczone w pamięci CHIP. Pamiętać należy również o wyrównaniu struktur danych do jednego słowa (parzysty adres w pamięci).

Tabela 4-1 pokazuje strukturę danych sprite’a w pamięci i opisuje funkcję każdego słowa w strukturze.

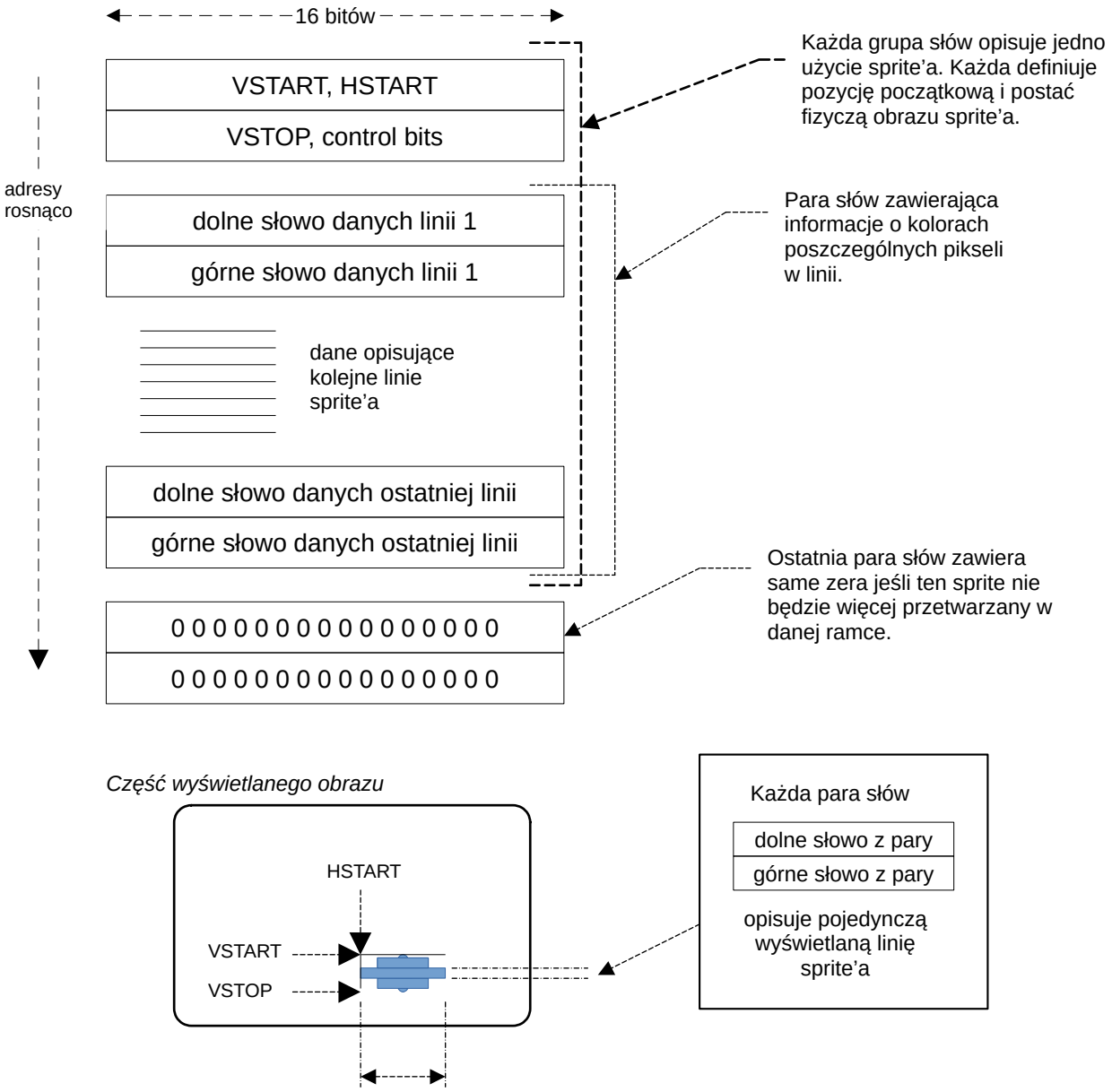
Lokacja w pamięci	**Słowo 16-bitowe**	**Funkcja**
N	Pierwsze słowo kontrolne	Pozycja pionowa i pozioma początku sprite’a
N+1	Drugie słowo kontrolne	Pozycja pionowa końca sprite’a
N+2	Dolne słowo określające kolor	Bity kolorów pierwszej linii
N+3	Górne słowo określające kolor	Bity kolorów pierwszej linii
N+4	Dolne słowo określające kolor	Bity kolorów drugiej linii
N+5	Górne słowo określające kolor	Bity kolorów drugiej linii
	.	
	.	
	.	
	Słowa końca danych	Dwa słowa określające następne użycie sprite’a

Tabela 4-1: Struktura danych sprite’a

Wszystkie adresy w pamięci dla sprite’ów wskazują na słowa. Minimalna ilość ciągłej pamięci niezbędnej dla sprite’a musi pomieścić dwa początkowe słowa kontrolne, dwa słowa określające kolory dla każdej linii sprite’a oraz dwa słowa kończące strukturę danych.

Ponieważ ta struktura musi być dostępna dla układów specjalizowanych – musi być umieszczona w pamięci CHIP.

Poniższy obrazek pokazuje w jaki sposób dane w strukturze danych sprite’a wpływają na sam obiekt sprite’a.



Rys. 4-7: Struktura danych sprite'a

Pierwsze słowo kontrolne sprite'a: SPRxPOS

Słowo to służy do przechowywania pionowej (VSTART) i poziomej (HSTART) pozycji początkowej sprite'a. Od tej pozycji pierwsza linia sprite'a będzie wyświetlana na ekranie.

Bity 15-8 zawierają niższe 8 bitów pozycji VSTART
Bity 7-0 zawierają wyższe 8 bitów pozycji HSTART

Drugie słowo kontrolne sprite'a: SPRxCTL

To słowo przechowuje końcową pionową pozycję sprite'a na ekranie (wskazuje na linię PO ostatniej wyświetlonej linii sprite'a). Słowo to zawiera też dodatkowe dane sprita, niektóre z nich są

uzupełnieniem danych z pierwszego słowa a inne związane są z łączeniem sprte'ów, co będzie opisane później.

SPRxCTL	
Bity 15-8	Dolne 8 bitów pozycji VSTOP
Bit 7	Używany w łączeniu sprite'ów
Bity 6-3	Nieużywane, ustaw 0
Bit 2	Górny bit pozycji VSTART
Bit 1	Górny bit pozycji VSTOP
Bit 0	Dolny bit pozycji HSTART

Wartość (VSTOP – VSTART) określa wysokość wyświetlanego sprite'a wyrażoną w liniach obrazu.

Słowa opisujące kolor sprite'a

Każdą poziomą linię sprite'a opisują dwa słowa opisujące kolor: słowo górne i dolne. Aby wyznaczyć ile słów opisu koloru potrzebujesz wystarczy pomnożyć wysokość sprite'a (ilość linii) przez 2. Bity górnego słowa odpowiadają lewemu bitowi wyznaczającemu rejestr koloru każdego piksela sprite'a. Na bity dolnego słowa składają się prawe bity oznaczajace rejestr koloru dla każdego punktu sprite'a.

Najlepiej wytłumaczyć to na przykładzie. Zajmijmy się znanym już nam projektem statku kosmicznego opracowanym kilka stron wcześniej. Jak już było wspomniane, każdy punkt sprite'a może przyjać wartość 0 – 3, co określa rejestr koloru dla danego punktu. Popatrzmy teraz na nasz statek:

```
0000122332210000
0001223333221000
0012223333222100
0001223333221000
0000122332210000
```

Teraz każdy numer należy zamienić na jego binarną postać. Przetworzona pierwsza linia przedstawiona jest poniżej. Liczby dwójkowe są odwzorowane pionowo z mniej znaczącym bitem na górze a bardziej znaczącym pod nim. W ten właśnie sposób wartości definiujące rejestry kolorów dla sprite'a są zapisane w pamięci.

```
0000100110010000 <- dolne słowo linii sprite'a
0000011111100000 <- górne słowo linii sprite'a
```

Pierwszy rząd zawiera mniej znaczące słowo rejestru koloru dla pierwszej linii naszego sprite'a. W drugim rzędzie mamy bardziej znaczące słowo. Każdą linię naszego spritea należy opisać dwoma słowami używając powyższego sposobu. Zobacz Rys. 4-7.

Każda liczba opisana przez odpowiednią kombinację dwóch bitów zawartych w obu słowach wyznacza odpowiedni rejestr koloru, jaki będzie użyty dla danego punktu sprite'a w zależności od kanału DMA. Przykładowo, dla kanału 0, sprite będzie używał kolorów z rejestrów 17 – 19. Liczby dwójkowe odpowiadające rejestrom kolorów 17 – 19 dla kanału 0 przedstawiamy w Tebeli 4-2 poniżej.

Liczba dwójkowa	Number rejestru koloru
00	Przeźroczysty
01	17
10	18
11	19

Tabela 4-2: Rejestry kolorów sprite’a

Pamiętaj, że wartość dwójkowa 00 zawsze oznacza kolor „przeźroczysty”, nigdy nie oznacza rejestru koloru z wyjątkiem tła.

Słowo końca danych

Kiedy pionowa pozycja licznika promienia wizji jest równa wartości VSTOP zdefiniowanej w słowach kontrolnych sprite’a, następne dwa słowa pobrane ze struktury danych sprite’a są zapisywane w rejestrach kontrolnych sprite’a zamiast w rejestrach kolorów. Te dwa słowa są interpretowane przez układ graficzny w taki sam sposób jak słowa, które zostały zapisane w rejestrach kontrolnych na początku. Jeśli wartość VSTART zawarta w tych słowach jest mniejsza niż obecna pozycja promienia wizji, dany sprite nie zostanie ponownie użyty podczas jednego wyświetlenia. Czyli w przypadku, kiedy nie chcemy wielokrotnie użyć danego sprite’a, oba słowa powinny zawierać same 0. Ponowne użycie sprite’ów zostanie omówione później.

Poniższa struktura danych wyświetli nasz statek kosmiczny na pozycji V = 65 i H = 128 na widocznej części ekranu .

```
SPRITE:
      DC. W $6D60,$7200 ;VSTART, HSTART, VSTOP
      DC. W $0990,$07E0 ;pierwsza para słów opisujących
      DC. W $13C8,$0FF0
      DC. W $23C4,$1FF8
      DC. W $13C8,$0FF0
      DC. W $0990,$07E0
      DC. W $0000,$0000 ;koniec danych sprite’a
```

Wyświetlanie sprite’a

Po zbudowaniu struktury danych sprite’a musisz poinformować system, że chcesz go wyświetlić. Ta seksja opisuje tryb „automatyczny” wyświetlania sprite’a. W tym trybie, jak tylko kanał DMA sprite’a zaczyna otrzymywać dane, wyświetla je na ekranie. Wyświetlanie to trwa aż do osiągnięcia pozycji VSTOP. Tryb ręczny zostanie opisany niżej w tym rozdziale.

Aby wyświetlić sprite’a konieczne są następujące kroki:

1. Określ, który z ośmiu kanałów DMA chcesz użyć. Upewnij się, że wybrany kanał jest dostępny.
2. Ustaw wskaźniki sprite’ów informując tym samym system, gdzie znajdują się dane sprite’ów.

3. Włącz bezpośredni dostęp do pamięci (DMA) dla sprite’a, jeśli jeszcze tego nie zrobiłeś.
4. Dla każdego wyświetlenia, ustaw ponownie wskaźniki sprite’ów podczas wygaszenia poziomego.

Jeśli DMA dla sprite’a zostanie wyłączone podczas gdy sprite jest wyświetlany (czyli po przekroczeniu VSTART ale przed osiągnięciem VSTOP), system będzie nadal wyświetlał ostatnio pobraną linię sprite’a. Spowoduje to pojawienie się pionowego paska na ekraie. Zaleca się wyłączanie DMA dla sprite’a tylko podczas pionowego wygaszenia lub w momencie osiągnięcia przez promień wizyjny pozycji, w której jesteśmy pewni, że dany sprite się nie znajduje.

WYBÓR KANAŁU DMA I USTAWIENIE WSKAŹNIKÓW

Przy określieniu którego kanału DMA użyć należy wziąć pod uwagę kolory sprite’a oraz jego priorytet wyświetlania.

Kanał DMA sprite’a uzywa dwóch wskaźników do odczytu danych i słów kontrolnych. W czasie wygaszenia pionowego, zanim sprite zostanie wyświetlony, wskaźniki te powinny zostać zapisane adresem danych sprite’a. Wskaźniki te noszą nazwę SPRxPTH i SPRxPTL gdzie „x” definiuje numer kanału DMA dla sprite’a. SPRxPTH zawiera bardziej znaczżce trzy bity adresu pierwszego słowa danych sprite’a. Pod adresem SPRxPTL zapisujemy mniej znaczące 16 bitów tego adresu. Najmniej znaczący bit SPRxPTL jest ignorowany. Oznacza to, że dane sprite’a muszą być wyrównane do słowa. Czyli tylko 15 bitów adresu zapisanego w SPRxPTL jest używanych. Jak zwykle, nie ma konieczności zapisywania obu wskaźników osobno. Możliwe jest zapisane długiego słowa do SPRxPTH.

W poniższym przykładzie procesor inicjalizuje wskaźniki danych dla sprite’a o numerzez 0. Zazwyczaj odbywa się to przy użyciu odpowiedniej instrukcji Coppera. Dane sprite’a znajdują się pod adresem $20000.

```
MOVE.L #$20000,SPR0PTH+CUSTOM      ; Write $20000 to sprite 0 pointer...
```

Wskaźniki te są dynamiczne. Są one zwiększane przez DMA sprite’a aby wskazywały na słowa kontrolne, potem na słowa zawierające dane sprite’a i ostatecznie na słowa końca danych. Po przeczytaniu informacji kontrolujących sprite’a i zapisaniu ich w odpowiednich rejestrach, odczytywane są dane rejestrów kolorów. Słowa rejestrów kolorów są zapisywane w rejestrach danych sprite’a, które są używane przez kanał DMA do wyświetlenia danych na ekranie. Więcej informacji na temat w jaki sposób kanały DMA obsługują wyświetlanie danych zaglądnij do sekcji „Szczegóły sprzętowe” poniżej.

PONOWNE USTAWIENIE WSKAŹNIKÓW

Dla pojedynczej klatki system automatycznie odczyta strukturę danych i wyświetli sprite’a na ekranie w kolorach opisanych przez rejestry kolorów sprite’a. Jeśli chcesz, żeby sprite był wyświetlany dalej, musisz ponownie ustawić wskaźniki danych sprite’a podczas każdego wygaszenia pionowego. Jest to konieczne, ponieważ podczas każdej klatki wskaźniki te są modyfikowane tak, żeby wskazywały na właściwe dane, które mają być wyświetlone w trakcie postępu promienia wizji.

Proces ponownego ustawienia wskaźników najlepiej zrzucić na barki Coppera.

PRZYKŁAD WYŚWIETLENIA SPRITE'A

Poniższy przykład wyświetli statek kosmiczny na pozycji V = 65, H = 128. Pamiętać należy o dołączeniu pliku „hw_examples.i", który znajdziesz w Dodatku I.

```
; Najpierw ustawiamy jeden bitplan
;
      LEA CUSTOM,a0                ; a0 wskazuje na układy specjalizowane
      MOVE.W #$1200,BPLCON0(a0)    ; 1 bitplan, kolor włączony
      MOVE.W #$0000,BPL1MOD(a0)    ; Modulo = 0
      MOVE.W #$0000,BPLCON1(a0)    ; przesunięcie poziome = 0
      MOVE.W #$0024,BPLCON2(a0)    ; Sprite'y mają priorytet przed plafieldami
      MOVE.W #$0038,DDFSTRT(a0)    ; początek pobierania danych
      MOVE.W #$00D0,DDFSTOP(a0)    ; koniec pobierania danych

; Definicje okna wyświetlania.

      MOVE.W #$2C81,DIWSTRT(a0)    ; Ustaw początek okna wyświetlania
                                   ; Pionowy początek w starszym bajcie
                                   ; Poziomy początek * 2 w młodszym bajcie
      MOVE.W #$F4C1,DIWSTOP(a0)    ; Ustaw koniec okna wyświetlania
                                   ; Pionowy koniec w starszym bajcie
                                   ; Poziomy koniec * 2 w młodszym bajcie
;
; Ustawienie rejestrów kolorów
;
      MOVE.W #$0008,COLOR00(a0)    ; Background color = dark blue
      MOVE.W #$0000,COLOR01(a0)    ; Foreground color = black
      MOVE.W #$0FF0,COLOR17(a0)    ; Color 17 = yellow
      MOVE.W #$00FF,COLOR18(a0)    ; Color 18 = cyan
      MOVE.W #$0F0F,COLOR19(a0)    ; Color 19 = magenta
;
; Ustaw listę Coppera pod adresem $20000
;
      MOVE.L #$20000,a1       ; Ustaw A1 na miejsce docelowe listy Coppera
      LEA COPPERL(pc),a2      ; Ustaw A2 na źródło Copperlisty
CLOOP:
      MOVE.L (a2),(a1)+       ; Skopiuj długie słowo
      CMP.L #$FFFFFFFE,(a2)+  ; Sprawdź czy nie koniec Copperlisty
      BNE CLOOP               ; Zapętl, aż cała lista zostanie skopiowana

;
; Dane sprite'a do $25000
;
      MOVE.L #$25000,a1       ; Ustaw A1 na miejsce docelpwe sprite'a
      LEA SPRITE(pc),a2       ; Ustaw A2 na źródło danych sprite'a
SPRLOOP
      MOVE.L (a2),(a1)+       ; Skopiuj długie słowo
      CMP.L #$00000000,(a2)+  ; Sprawdź czy nie koniec danych sprite'a
      BNE SPRLOOP             ; Zapętl, aż cały sprite zostanie skopiowany
;
; Teraz zapiszemy puste dane sprite'a pod adresem $30000, ponieważ  8 sprite'ów
; jest włączone w tym samym czasie ale my użyjemy tylko jednego. Pozostałe
; sprite'y będą wskazywać na puste dane.
;
      MOVE.L #$00000000,$30000     ; Ustaw puste dane
;
; Wskaż Copperlistę Copperowi
;
```

```
      MOVE.L #$20000,COP1LC(a0)
;
; Fill bitplane with $FFFFFFFF.
;
      MOVE.L #$21000,a1            ; Ustaw A1 na bitplan
      MOVE.W #1999,dO              ; 2000-1 (dla dbf) długih słów = 8000 bajtów
FLOOP
      MOVE.L #$FFFFFFFF,(a1)+      ; Skopiuj długie słowo o wartości $FFFFFFFF
      DBF d0,FLOOP                 ; Pomniejsz, powtarzaj aż osiągnie false
;
; Start DMA.
;
      MOVE.W d0,COPJMP1(a0)        ; Wymuś załadowanie do Coppera
      MOVE.W #$83A0,DMACON(a0)     ; Bitplan, Copper i DMA sprite'a
      RTS                          ;..powrót do reszty programu..

;
; To jest Copperlista dla jednego bitplanu i 8 sprite'ów
; Dane Bitplanu są pod adresem $21000
; Dane sprite'a 0 są pod adresem $25000; pozostałe 7 pod adresem $30000
;
COPPERL:
      DC.W BPL1PTH,$0002     ; Wskaźnik bitplanu 1 = $21000
      DC.W BPL1PTL,$1000
      DC.W SPR0PTH,$0002     ; Wskaźnik sprite'a 0 = $25000
      DC.W SPR0PTL,$5000
      DC.W SPR1PTH,$0003     ; Wskaźnik sprite'a 1 = $30000
      DC.W SPR1PTL,$0000
      DC.W SPR2PTH,$0003     ; Wskaźnik sprite'a 2 = $30000
      DC.W SPR2PTL,$0000
      DC.W SPR3PTH,$0003     ; Wskaźnik sprite'a 3 = $30000
      DC.W SPR3PTL,$0000
      DC.W SPR4PTH,$0003     ; Wskaźnik sprite'a 4 = $30000
      DC.W SPR4PTL,$0000
      DC.W SPR5PTH,$0003     ; Wskaźnik sprite'a 5 = $30000
      DC.W SPR5PTL,$0000
      DC.W SPR6PTH,$0003     ; Wskaźnik sprite'a 6 = $30000
      DC.W SPR6PTL,$0000
      DC.W SPR7PTH,$0003     ; Wskaźnik sprite'a 7 = $30000
      DC.W SPR7PTL,$0000
      DC.W $FFFF,$FFFE       ; Koniec Copperlisty
;
; Sdane sprite'a dla statku kosmicznego. Pojawi się na ekranie w pozycji
; X=65 i Y=128
;
SPRITE:
      DC.W $6D60,$7200       ; VSTART, HSTART, VSTOP
      DC.W $0990,$07E0       ; Pierwsza para słów
      DC.W $13C8,$0FF0
      DC.W $23C4,$1FF8
      DC.W $13C8,$0FF0
      DC.W $0990,$07E0
      DC.W $0000,$0000       ; Koniec danych sprite'a
```

Przemieszczanie sprite’a

Sprite stworzony w trybie automatycznym może zostać przemiszczony na ekranie dzięki określeniu nowej pozycji w strukturze danych. Podczas wyświetlania każdej klatki dane sprite’a są ponownie odczytywane i wyświetlane na ekranie. Tak więc jeśli zmienisz dane opisujące pozycję zanim sprite zostanie wyświetlony ponownie, pojawi się on w innym miejscu ekranu tworząc wrażenie ruchu.

Musisz zadbać o to, żeby sprite nie przemieszczał się (czyli nie należy zmieniać danych kontrolnych) w tym samym czasie kiedy system używa tych danych do wyświetlenia obiektu. Jeśli o to nie zadbasz, może się okazać, że system odczyta pozycję startową w jeden klatce a końcową w kolejnej. W wyniku tego mogą pojawić się różne nieoczekiwane „śmieci” na ekranie. Zazwyczaj zmiany danych kontrolnych powinno dokonywać się podczas wygaszenia pionowego. Przykład użycia do tego celu Coppera pokazuje przykład poniżej.

Sprite’y przemiszczając się po ekranie mogą kolidować między sobą albo też z jednym z dwóch pól gry. Sprzętowe wykrywanie kolizji może zostać użyte do dodania różnych efektów specjalnych. Dodatkowo, wykrywanie kolizji może zostać użyte do ograniczenia ruchu obiektów w obrębie wyznaczonych granic na ekranie. Wykrywanie kolizji zostało szerzej opisane w Rozdziale 7, „Układy kontroli systemu”

W poniższym przykładzie ukazującym poruszający się sprite, statek kosmiczny odbija się od krawędzi ekranu zmieniając kierunek ruchu w chwili osiągnięcia linii granicznej.

Dane pozycjonujące sprite’a, zawierające VSTART i HSTART, znajdują się w pamięci pod adresem $25000. VSTOP umieszczono pod adresem $25002. Aby zmienić pozycję sprite’a wystarczy odpowiednio zmienić zawartość tych adresów. Jednorazowo podczas wyświetlania każdej klatki obrazu, VSTART jest zwiększany lub zmniejszany o 1 a HSTART o 2. Następnie wyliczana jest nowa wartość VSTOP, która zawsze jest równa VSTART + 6.

```
      MOVE.B #151,d0    ; Initialize horizontal count
      MOVE.B #194,d1    ; Initialize vertical count
      MOVE.B #64,d2     ; Initialize horizontal position
      MOVE.B #44,d3     ; Initialize vertical position
      MOVE.B #1,d4      ; Initialize horizontal increment value
      MOVE.B #1,d5      ; Initialize vertical increment value
;
; Here we wait for the start of the screen updating.
; This ensures a glitch-free display.
;
      LEA CUSTOM,a0           ; Set custom chip base pointer
VLOOP:
      MOVE.B VHPOSR(a0),d6    ; Read Vertical beam position.
;Only insert the following line if you are using a PAL machine.
;
      CMP.B #$20,d6           ; Compare with end of PAL screen.
      BNE.S VLOOP             ; Loop if not end of screen.
;Alternatively you can use the following code:
;VLOOP:
;     MOVE.W INTREQR(a0),d6         ;Read interrupt request word
;     AND.W #$0020,d6               ; Mask off all but vertical blank bit
;     BEQ VLOOP                     ; Loop until bit is a 1
;     MOVE.W #$0020,INTREQ(aO)      ; Vertical bit is on, so reset it
;
; Please note that this will only work if you have turned OFF the Vertical
; blanking interrupt enable (not recommended for long periods).
```

```
      ADD.B d4,d2             ; Increment horizontal value
      SUBQ.B #1,d0            ; Decrement horizontal counter
      BNE L1
      MOVE.B #151,d0          ; Count exhausted, reset to 151
      EOR.B #$FE,d4           ; Negate the increment value
L1:   MOVE.B d2,$25001        ; Write new HSTART value to sprite
      ADD .B d5,d3            ; Increment vertical value
      SUBQ.B #1,d1            ; Decrement vertical counter
      BNE L2
      MOVE.B #194,d1          ; Count exhausted, reset to 194
      EOR.B #$FE,d5           ; Negate the increment value
L2:   MOVE.B d3,$25000        ; Write new VSTART value to sprite
      MOVE.B d3,d6            ; Must now calculate new VSTOP
      ADD.B #6,d6             ; VSTOP always VSTART+6 for spaceship
      MOVE.B d6,$25002        ; Write new VSTOP to sprite
      BRA VLOOP               ; Loop forever
```

Tworzenie dodatkowych sprite’ów

Aby użyć dodatkowych sprite’ów, dla każdego z nich należy przygotować odpowiednią strukturę danych tak jak opisaliśmy to do tej pory. Dla kanału DMA sprite’a 1 użyjemy wskaźników SPR1PTH oraz SPR1PTL, dla sprite’a 2 użyjemy SPR2PTH oraz SPR2PTL, i tak dalej.

> *Uwaga dla DMA sprite’a*. Włączając DMA dla jednego sprite włączasz jedocześnie kanały dla wszystkich sprite’ów i ustawiasz je wszystkie w tryb automatyczny. Dzięki temu nie ma konieczności powtarzania tego kroku osobno dla wszystkich sprite’ów.

W chwili kiedy kanały DMA dla sprite’ów są włączone, wskaźniki spriteów muszą wkazywać na rzeczystiste dane sprite’a albo na dane bezpiecznie zerujące sprite’a. Niezainicjalizowany sprite może spowodować pojawienie się „fałszywego” sprite’a zainicjalizowanego losowymi danymi z pamięci.

Należy pamiętać, że niektóre sprite’y mogą przestać być używalne, jeśli dodatkowe cykle DMA zostaną przydzielone do zadań obsługujących wyświetlanie ekranu takich jak wyświetlanie bardzo szerokiego ekranu albo przesuwanie poziome zawartości ekranu (Zobacz Rys. 6-9: Przydział slotów czasowych DMA).

Warto również zaznaczyć, że każda para sprite’ów pobiera kolory z różnych zestawów rejestrów, tak, jak pokazano w Tabeli 4-3 poniżej.

Numery sprite’ów	**Rejestry kolorów**
0 i 1	17 – 19
2 i 3	21 – 23
4 i 5	25 – 27
6 i 7	29 – 31

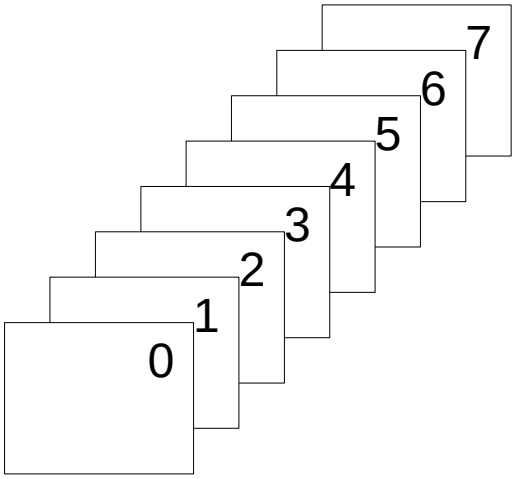
Tabela 4-3: Rejestry kolorów dla par sprite’ów

> Uwaga: Niektóre sprite’y stają się nieużywalne, w chwili gdy dodatkowe cykle DMA zostaną przydzielone do wyświetlania ekranu. (Zobacz Ryz. 6-9: Przydział slotów czasowych DMA).

PRIORYTET SPRITE'A

W przypadku kiedy będziemy używać więcej niż jednego sprite'a warto zastanowić się nad kolejnościa z jaką będą pojawiać się na ekranie. Kolejność ta określa, który sprite będzie wyświetlany „nad" a który „pod" jeśli oba będą zajmowały tą samą pozycję. Każdy sprite ma z góry ustalony priorytet. Sprite o najniższym numerze ma najwyższy priorytet i będzie przykrywał wszystkie pozostałe. Sprite o najwyższym numerze ma najniższy priorytet. Dobrze ilustruje to rysunek 4-8 poniżej.

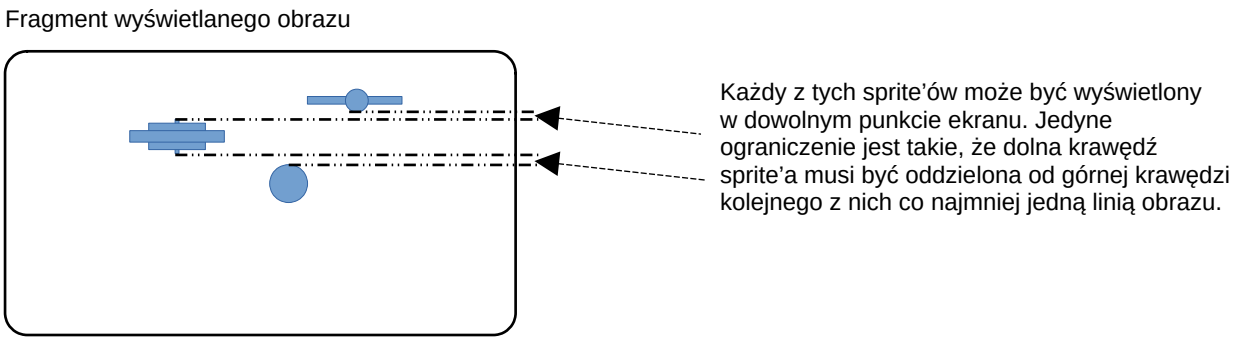
> *Więcej o priorytetach*. Dokładniejszy opis priorytetów sprite'ów zawarliśmy w Rozdziale 7, "Układy kontroli systemu"



Rys. 4-8. Priorytet sprite'ów

Ponowne użycie kanałów DMA spritea

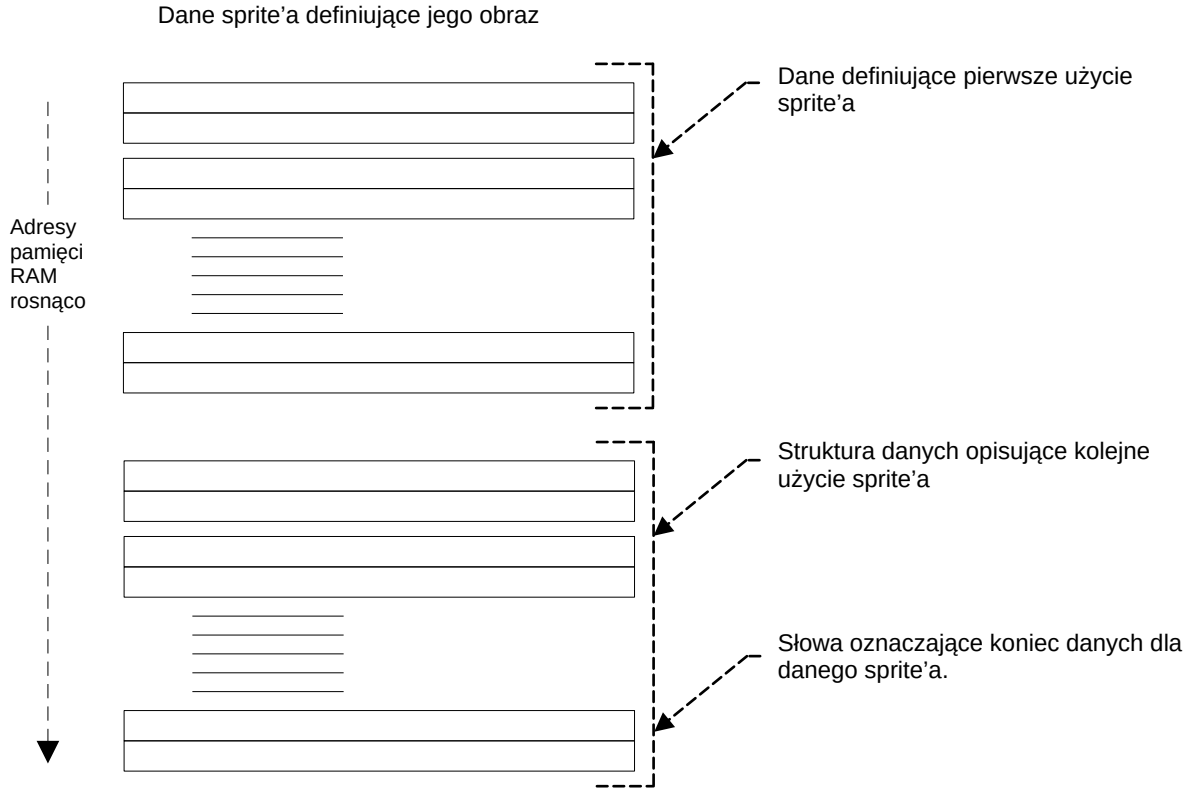
Każdy z ośmiu kanałów DMA sprite'ów może obsłużyć więcej niż jeden niezależny obiekt. Z pewnością będą chwile, kiedy będziesz potrzebował więcej niż 8 obiektów lub w celu uzyskania większej ilości kolorów lub większych obiektów połączysz kilka sprite'ów w jeden ograniczając tym samym ich maksymalną ilość. Na szczęście istnieje możliwość użycia danego kanału DMA więcej niż raz w trakcie jedej klatki. Pokazano to na ryzunku 4-9.



Rys. 4-9. Typowy przykład ponownego użycia sprite'a

Używając jednego sprite'a na kanał, na końcu struktury danych widoczne są dwa słowa o wartości zero. Kanał DM napotykając na nie kończy pobieranie danych dla sprite'a w danej klatce obrazu. W

celu dalszego użycia bierzącego kanału do wyświetlenia kolejnych obiektów, w miejsce dwóch zerowych słów wstawiamy kolejną strukturę danych sprite’a, który zostanie wyświetlony poniżej poprzedniego. Dwa zerowe słowa zamykające kanał sprite’a umieszcza się na końcu ostatniej struktury obsługiwanej przez dany kanał DMA. Dla przykładu, rysunek 4-10 pokazuje strukturę danych dla przypadku ponownie użytego sprite’a.



Rys. 4-10: Typowa struktura danych sprite’a wielokrotnego użytku

Jedynym ograniczeniem w przypadku ponownego użycia sprite’a jest konieczność oddzielenia dolnej krawędzi jednego obiektu od górnej krawędzi kolejnego obiektu przez przynajmniej jedną linię obrazu (jedno przejście promienia wizyjnego). Ograniczenie to jest niezbędne ponieważ tylko dwa cykle DMA na jedną poziomą linię są przydzielane do każdego z ośmiu kanałów. Aby kanał DMA mógł pobrać dane opisujące kolejne użycie sprite’a niezbędny jest czas jaki daje wygaszenie poziome.

Poniższy przykład wyświetla sprite’a w postaci obrazu pojazdu kosmicznego a następnie wyświetla go ponownie jako zupełnie inny obiekt. Ponieważ ma to wpływ tylko na dane sprite’a, tylko te dane są pokazane poniżej. Sprite jednak wygląda najlepiej jeśli rejestry kolorów są ustawione jak w poniższym przykładzie.

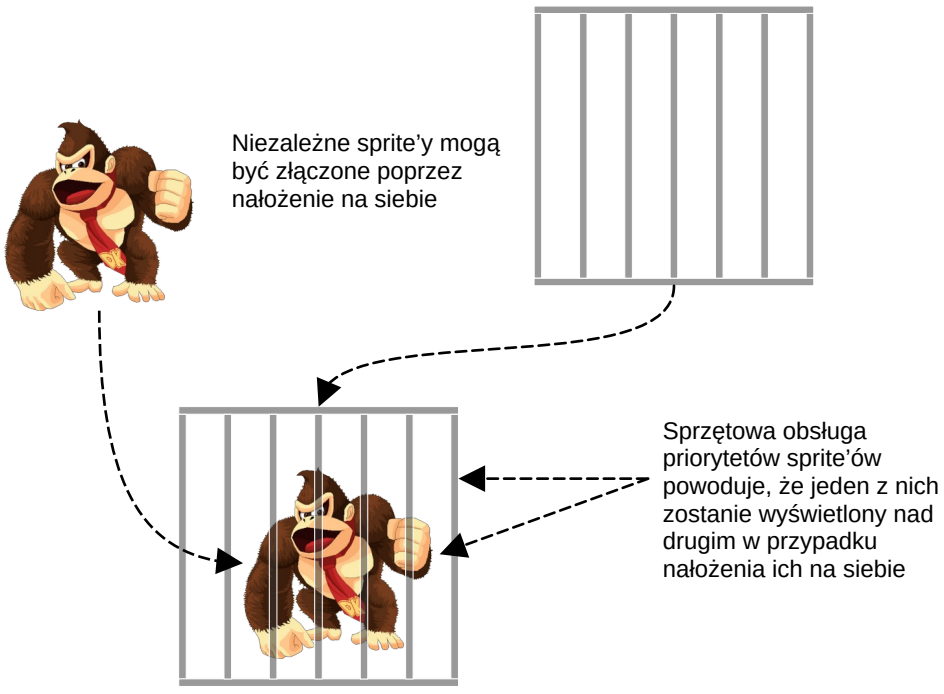
```
      LEA CUSTOM,a0
      MOVE.W #$0F00,COLOR17(a0)          ; Color 17 - czerwony
      MOVE.W #$0FF0,COLOR18(a0)          ; Color 18 - żółty
      MOVE.W #$0FFF,COLOR19(a0)          ; Color 19 - biały
SPRITE:
      DC.W $6D60,$7200
      DC.W $0990,$07E0
      DC.W $13C8,$0FF0
      DC.W $23C4,$1FF8
```

```
DC.W $13C8,$0FF0
DC.W $0990,$07E0
DC.W $8080,$8D00              ; VSTART, HSTART, VSTOP dla nowego sprite'a
DC.W $1818,$0000
DC.W $7E7E,$0000
DC.W $7FFE,$0000
DC.W $FFFF,$2000
DC.W $FFFF,$2000
DC.W $FFFF,$3000
DC.W $FFFF,$3000
DC.W $7FFE,$1800
DC.W $7FFE,$0C00
DC.W $3FFC,$0000
DC.W $0FF0,$0000
DC.W $03C0,$0000
DC.W $0180,$0000
DC.W $0000,$0000              ; koniec danych sprite'a
```

Sprite'y nakładane

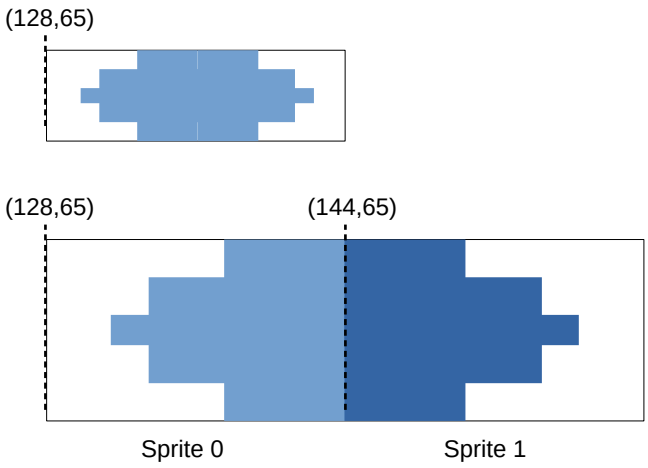
Aby uzyskać bardziej skomplikowane lub większe obiekty sprite'y można nakładać na siebie. Nakładanie oznacza, że sprite'y mają takie samo lub bardzo zbliżone do siebie położenie na ekranie. Stosunkowo bliskie umiejscowienie sprite'ów w rezultacie może dać obiekt, który jest szerszy niż 16 pikseli.

Video priorytety sprite'ów zapewniają, że jeden z nich zawsze będzie pojawiał się pod spodem drugiego gdy sprite'y nakładają się na siebie. Układy obsługujące sprite'y nadają sprite'om o najwyższym numerze najniższy priorytet. Projektując widok należy się upewnić, że w przypadku nakładania się sprite'ów, obiekty „bliższe" graczowi będą miały niższy numer niż te, które są bliżej tła. Na przykładzie pokazanym na rysunku 4-11, klatka powinna zostać wygenerowana przez kanał DMA sprite'a o niższym numerze niż kanał DMA wyświetlający małpę.



Rys. 4-11: Nakładanie sprite'ów (nie złączonych)

W celu uzyskania szerszego sprite’a można umieścić dwa sprite’y obok siebie. Rysunek 4-12 pokazuje sprite’a statku kosmicznego oraz sposób na jego dwukrotne powiększenie przy użyciu dwóch sprite’ów wyświetlonych jeden zaraz obok drugiego.



Rys. 4-12: Pozycjonowanie sprite’ów obok siebie

Łączenie sprite’ów

Sprite’y na Amidze mają trzy kolory (plus „kolor” przeźroczysty). Jest jednak pewien sposób na uzyskanie sprite’a 15-to kolorowego (plus przeźroczysty). W tym celu łączy się dwa sprite’y. Aby to zrobiś musisz wykonać poniższe kroki:

- Użyć dówch kanałów tworząc dwa sprite’y tego samego rozmiaru umieszczone w tej samej pozycji na ekranie.
- Ustawić bit o nazwie ATTACH w drugim słowie kontrolnym sprite’a.

15 kolorów pochodzi z pełnego zakresu rejestrów kolorów dostępnych dla sprite’ów, czyli z rejestrów o numerach od 17 do 31. Dodatkowe kolory mogą być wybrane dzięki temu, że każdy piksel sprite’a opisyany jest przez 4 bity zamiast 2 jak w przypadku pojedynczego, niezłączonego sprite’a. Każdy sprite w złączonej parze udostępnia po dwa bity dla określenia numeru rejestru koloru. Na przykład, jeśli używamy kanałów DMA o numerach 0 i 1, górne i dolne słowa rejestru koloru dla pierwszej linii sprite’a w strukturach danych obu obiektów są łączone w jedną linię złączonego sprite’a.

Sprite’y mogą być łączone w poniższych kombinacjach:

Sprite 1 ze spritem 0
Sprite 3 ze spritem 2
Sprite 5 ze spritem 4
Sprite 7 ze spritem 6

Dowolne lub wszystkie z tych złączonych obiektów mogą być aktywne w czasie wyświetlania pojedynczej klatki obrazu. Dla przykładu, załóżmy, że chcemy mieć sprite’a statku kosmicznego używającego większej liczby kolorów oraz, że używamy kanałów DMA o numerach 0 i 1. Mamy więc sprite’a posiadającego pięć kolorów plus przeźroczystość.

```
0000154444510000
0001564444651000
0015676446765100
0001564444651000
0000154444510000
```

Pierwsza linia tego sprite’a wymaga czterech słów danych, pokazanych w tabeli 4-4, aby określić poprawny numer rejestru koloru w postaci binarnej.

	Numer piskela															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Linia 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Linia 2	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
Linia 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Linia 4	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0

Tabela 4-4: Słowa danych dla pierwszej linii sprite’a statku kosmicznego

Największy numer sprite’a (w naszym przykładzie to numer 1) definiuje najstarsze bity liczby dwójkowej. Najstarsze słowo danych każdego sprite’a definiuje skrajnie lewą cyfrę. W ten sposób powyższe linie są zapisane w strukturze danych sprite’a jak pokazano poniżej:

Linia 1 Bardziej znaczące słowo Sprite’a 1 dla linii 1

Linia 2 Mniej znaczące słowo Sprite’a 1 dla linii 1

Linia 3 Bardziej znaczące słowo Sprite’a 0 dla linii 1

Linia 4 Mniej znaczące słowo Sprite’a 0 dla linii 1

Na rysunku 4-7 pokazano w jaki sposób słowa te są umieszczone w pamięci. Pamiętać należy, że dane te są zawarte w dwóch strukturach danych sprite’ów.

Wartości binarne od 0 do 15 określają rejestry 17 do 31 jak pokazano w tabeli 4-5.

Liczba dziesiętna	**Liczba dwójkowa**	**Numer rejestru koloru**
0	0000	16*
1	0001	17
2	0010	18
3	0011	19
4	0100	20
5	0101	21
6	0110	22
7	0111	23
8	1000	24
9	1001	25
10	1010	26
11	1011	27

12	1100	28
13	1101	29
14	1110	30
15	1111	31

Tabela 4-5: Rejestry kolorów dla złączonych sprite’ów

* Nieużywany, oznacza punkt przeźroczysty

Złączenie następuje tylko wtedy, kiedy bit ATTACH, (bit 7) w drugim słowie kontrolnym sprite’a jest ustawiony (ma wartość 1) w strukturze danych dla sprite’a o nieparzystym numerze. W naszym przykładzie musimy ustawić bit 7 w drugim słowie kontrolnym w strukturze danych sprite’a 1.

Kiedy sprite’y są przemieszczane, Copperlista musi zadbać o to, żeby oba były zawsze na tej samej pozycji na ekranie. Jeśli nie będą trzymane razem na ekranie, ich punkty zmienią kolory. Każdy ze sprite’ów znów stanie się trójkolorowy ale kolory mogą być inne niż w przypadku, gdyby to były sprite’y niezłączone. Kolory dla sprite’a o niższym kolorze będą pochodzić z rejestów 17-19. Dla sprite’a o wyższym numerze będą to rejestry 20, 24 i 28.

Poniższa struktura danych definiuje sześciolorowy statek kosmiczny składający się z dwóch złączonych sprite’ów.

```
SPRITE0:
      DC.W $6D60,$7200        ; VSTART = 65, HSTART = 128
      DC.W $0C30,$0000        ; First color descriptor word
      DC.W $1818,$0420
      DC.W $342C,$0E70
      DC.W $1818,$0420
      DC.W $0C30,$0000
      DC.W $0000,$0000        ; End of sprite 0
SPRITE1:
      DC.W $6D60,$7280        ; Same as sprite 0 except attach bit on
      DC.W $07E0,$0000        ; First descriptor word for sprite 1
      DC.W $0FF0,$0000
      DC.W $1FF8,$0000
      DC.W $0FF0,$0000
      DC.W $07E0,$0000
      DC.W $0000,$0000        ; End of sprite 1
```

Tryb ręczny

W większości przypadków najlepszym sposobem na ładowanie sprite’ów jest użycie kanałœ DMA w trybie automatycznym. Czasami jednak użytecznym jest załadować te rejestry bezpośrednio przy użyciu jednego z mikroprocesorów. Sprite’y mogą być aktywowane ręcznie w każdej chwili kiedy nie są używane przez kanały DMA. Ten sam sprite, który wyświetla kontrolowaną przez DMA ikonę w pobliżu górnej krawędzi ekranu może zostać przeładowany ręcznie w celu wyświetlenia poziomego paska kolorów w okolicach dolnej krawędzi ekranu. Sprite’y mogą być włączane ręcznie nawet wtedy, kiedy ich kanały DMA są wyłączone.

Ręczne wyświetlenie Sprite’a polega na ustawieniu odpowiednich rejestrów danych SPRxDATB i SPRxDATA w takiej właśnie kolejności. Zapis do rejestru SPRxDATA odbywa się na końcu dlatego, że ten adres „uzbraja” (lub bardziej obrazowo - „odpala”) wyświetlanie sprite’a po najbliższym wygaszeniu pionowym jak tylko promień wizyjny osiągnie pozycję, która została

zdefiniowana jako początek sprite’a (w skrócie sprite będzie wyświetlony przy następnym „porównaniu poziomym”, ang. horizontal comparison). Dane zapisane do powyższych rejestrów będą wyświetlane w każdej linii na pozycji poziomej zapisanej w części „H” rejestrów SPOxPOS i SPRxCTL. Jeśli dane te pozostaną niezmienione, sprite przyjmie postać pionowego paska. Jeśli dane te będą zmieniać się dla każdej wyświetlanej linii, sprite przyjmie bardziej skomplikowaną postać – tak jak „normalny” sprite w trybie automatycznym.

Sprite może zostać „rozbrojony” („ubity”) przez zapis do rejestru SPRxCTL. Zapis do rejestru SPRxPOS, pozwoli na ręczne przemieszczenie sprite’a poziomo nawet podczas normalnego jego użycia.

Dodatkowe informacje o sprzęcie kontrolującym sprite’y

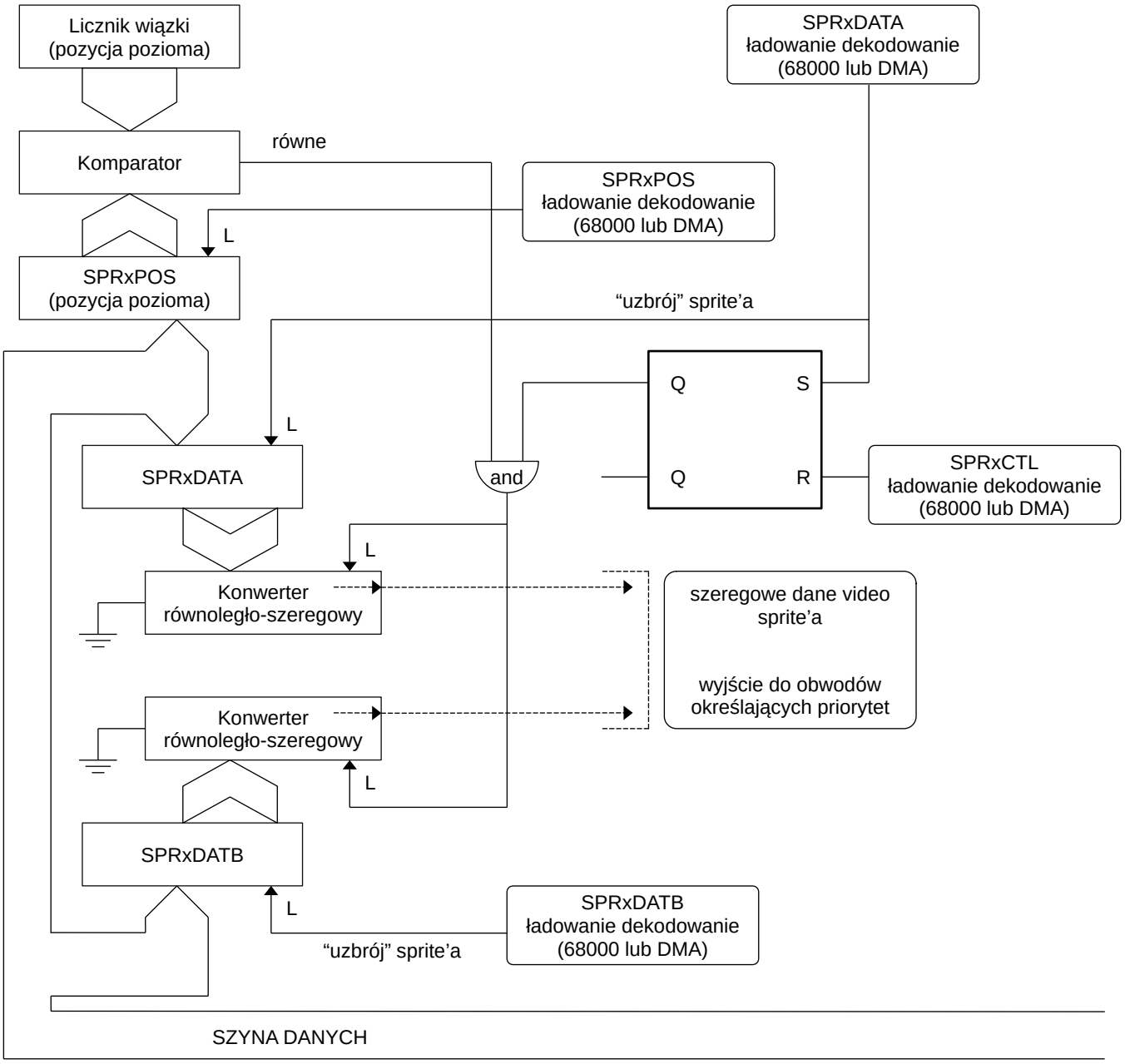
Sprite’y są tworzone przez obwody pokazane na ryzunku 4-13. Mamy tam schemat blokowy wyjaśniający w jaki sposób para słów staje się zestawem punktów wyświetlanych na ekranie.

Poniżej przedstawiono i opisano elementy obwodów odpowiedzialnych za wyświetlanie sprite’a.

- Rejesatry danych sprite’a. Rejestry SPRxDATA i SPRxDATB przechowują bitowo zakodowany opis pojedynczej linii sprite’a dla każdego z 8 możliwych obiektów. Linia ma szerokość 16 punktów, każda linia jest określona przez dwa słowa, które kodują jeden z trzech kolorów plus kolor przeźroczysty.
- Konwertery równoległy to szeregowego. Każdy z 16 bitów danych sprite’a jest wysyłany osobno do obwodu wybierającego kolor w chwili gdy piksel powiązany z tym bitem jest wyświetlany na ekranie.

  Natychmiast po tym jak dane są przeniesione z rejestrów danych sprite’a, każdy konwerter zaczyna „wysuwać” bity z siebie zaczynając od najbardziej znaczącego (skrajny lewy bit). Przesunięcie odbywa się jednorazowo podczas wyświetlenia każdego punktu niskiej rozdzielczości i trwa aż wszystkie 16 bitów zostanie przesłane do obwodów wyświetlających obraz. Kolejne przesunięcie i wyjście danych nie rozpocznie się aż nowe dane nie zostaną załadowane z rejestrów danych.

  Ponieważ obraz jest tworzony przez wiązkę elektronów podążającą od lewej do prawej krawędzi ekranu, bitowy obraz danych odpowiada dokładnie obrazowi, który pojawia się na ekranie (najbardziej znaczące dane po stronie lewej).
- Szeregowe dane wideo sprite’a. Dane sprite’a idą do obwodów kontrolujących priorytety w celu określenia właściwej kolejności wyświetlania pomiędzy sprite’ami a playfieldami.
- Rejestry pozycji sprite’a. Rejestry te, nazwane SPRxPOS, zawierają wielkość pozycji poziomej (wartość X) i wielkość pozycji pionowej (wartość Y) dla każdego z ośmiu sprite’ów.
- Rejestry kontrolne sprite’ów. Rejestry te, zwane SPRxCTL, zawierają końcową pozycję każdego ze sprite’ów oraz definiują czy sprite jest złączony czy nie.
- Licznik wiązki. Licznik wiązki udostępnia systemowi bierzącą pozycję wiązki video, która tworzy obraz.
- Komparator. To urządzenie porównuje wartość licznika wiązki z pozycją Y zapisaną w rejestrze SPRxPOS. Jeśli wiązka osiągnęła pozycję na której ma pojawić się górny lewy punkt sprite’a (pierwszy punkt sprite’a), komparator wysyła sygnał ładowania do konwertera szeregowo-równoległego i rozpoczyna się wyświetlanie sprite’a.



Rys. 4-13: Obwody kontrolujące sprite’a

Rysunek 4-13 pokazuje:

- Zapis do rejestrów kontrolnych sprite’a blokuje obwód komparatora. Zabezpiecza to system przed wysłaniem jakichkolwiek danych do rejestrów danych oraz na ekran.
- Zapis do rejestru A z danymi sprite’a włącza komparator. Dzięki temu znów jest możliwość wysyłania danych na ekran jeśli tylko pozioma pozycja promienia wizyjnego jest równa wartości z rejestru pozycji.
- Jeśli komparator jest włączony, dane sprite’a zostaną wysłane na ekran. Skrajny lewy piksel odczytany z danych sprite’a będzie umieszczony na pozycji zapisanej na poziomej składowej rejestru SPRxPOS.
- Tak długo jak komparator pozostaje włączony, bieżąca zawartość rejestru danych sprite’a będzie wyświetlana na wybranej poziomej pozycji linii na ekranie.

- Dane w rejestrach danych sprite’a nie ulegają zmianie. Mogą zostać nadpisane przez użytkownika albo zmienione pod kontrolą DMA.

Komponenty opisane wyżej tworzą automatyczny obraz DMA w następujący sposób: kiedy sprite’y są w trybie DMA, 18-bitowy rejestr wskaźnika sprite’a (składający się z SPRxPTH i SPRxPTL) zostaje użyty do odczytania pierwszych dwóch słów ze struktury danych sprite’a. Te słowa zapierają początkową i końcową pozycję sprite’a. Następnie słowa te zapisywane są do SPRxPOS i SPRxCTL. Po tym zapisie wartość we wskaźniku ustawiona jest na pierwsze słowo danych (mniej znaczące słowo danych pierwszej linii sprite’a).

Zapis do rejestru SPRxCTL wyłączył sprite’a. Teraz nakał DMA sprite’a będzie czekał aż pionowa pozycja wiązki wizyjnej będzie taka sama jak ta zapisana w częście VSTART (Y value) rejestru SPRxPOS. Gdy obie wartości są zgodne, system włącza dostęp do danych sprite’a.

Kanał DMA sprite’a sprawdza zawartość VSTOP (w rejestrze SPRxCTL, która to przechowuje pozycję linni tuż po ostatniej linii sprite’a) oraz VSTART (z rejestru SPRxPOS) w celu ustalenia ile linii danych sprite’a należy pobrać. Na każdą linię pobierane są dwa słowa i zapisywane do rejestrów danych. Pierwsze słowo jest przechowywane w rejestrze SPRxDATA a drugie w SPRxDATB.

Odczyt i zapis odbywa się w czasie wygaszenia poziomego aż do osiągnięcia lewej krawędzi ekranu. Aktywuje to komparatory poziomej pozycji sprite’a pozwalając im na rozpoczęcie wyświetlania danych sprite’a na ekranie jak tylko licznik poziomej pozycji promienia wizji osiągnie wartość przechowywaną w HSTART (X value) w rejestrze SPRxPOS.

Jeśli wynik działania VSTOP - VSTART równy jest zero, proces wyświetlania sprite’a nie odbywa się. Kolejna para słów danych zostanie pobrana ale nie będzie zapisana w rejestrach danych sprite’a. Zamiast tego stanie się następną parą słów danych zapisanych w SPRxPOS i SPRxCTL.

Kiedy sprite jest użyty tylko raz na klatkę, ostatnie dwa słowa danych, które następują po słowach odpowiedzialnych za rejestry kolorów są automatycznie ładowane jako następna zawartość rejestrów SPRxPOS i SPRxCTL. Aby zakończyć sprite’a po pierwszym zestawie danych ostatnia para słów powinna mieć wartość zerową (wszystkie bity wygaszone).

Tak więc, jeśli utworzyłeś wzór sprite’a w pamięci, ten sam wzór zostanie automatycznie użyty przrez DMA do odwzorowania sprite’a jako punkty na ekranie linia po linii.

Podsumowanie informacji o rejestrach sprite’ów

Mamy osiem zestawów rejestrów, które służą do definiowania sprite’ów. Każdy zestaw składa się z pięciu rejestrów. Dla uproszczenia opiszemy tu rejestry dla prite’a o numerzez 0. Pozostałe są identyczne, jedyną różnicą jest nazwa, zawierająca odpowiedni numer sprite’a.

WSKAŹNIKI

Wskaźniki to rejestry używane przez system do wskazania aktualnie używanych danych. W czasie wyświetlania ekranu, rejestry te są zwiększane w celu wskazania aktualnie wyświetlanych danych. Dlatego rejestry wskaźników muszą być ponownie zainicjalizowane w czasie wygaszenia pionowego.

SPR0PTH oraz SPR0PTL

Para rejestrów zawierająca 32-bitowy adres danych DMA dla sprite’a 0.

Nazwy rejestrów wskaźnikowych dla pozostałych sprite’ów wyglądają następująco:

SPR1PTH	SPR1PTL
SPR2PTH	SPR2PTL
SPR3PTH	SPR3PTL
SPR4PTH	SPR4PTL
SPR5PTH	SPR5PTL
SPR6PTH	SPR6PTL
SPR7PTH	SPR7PTL

REJESTRY KONTROLI

SPR0POS

To rejestr pozycji sprite’a 0. Słowo zapisane w tym rejestrze odpowiada za pozycję sprite’a na ekranie. Określa ono pozycję gdzie górny lewy punkt sprite’a zostanie umieszczony. Najbardziej znaczący bit pierwszego słowa danych sprite’a pojawi się na ekranie w miejscu zapisanym w tym rejestrze.

> *Rozdzielczość położenia sprite’ów*. Rozdzielczosć według której sprite’y mogą być rozmieszczane na pełnym ekranie wynosi 320 x 200 pikseli w trybie NTSC (320 x 256 punktów w trybie PAL). Rozdzielczosć prite’ów jest niezależna od rozdzielczości bitplanu.

Pozycje bitów:

- Bity 15 – 8 określają startową pozycję pionową, bity V7 – V0.
- Bity 7 – 0 określają startową pozycję poziomą, bity H8 – H1.

> *Uwaga*: Rejestr ten normalnie jest zapisywany wyłącznie przez kanał DMA sprite’a.

SPR0CTL

Rejestr ten normalnie jest używany wyłącznie przez kanał DMA sprite’a. Zawiera dane kontrolujące proces pobierania danych dla sprite’a. Pozycje bitów:

- Bity 15 – 8 określają końcową pozycję obrazu sprite’a, bity V7 – V0.
- Bit 7 to bit złączenia. Jego ustawienie jest poprawne tylko dla sprite’ów o numerach nieparzystych. Jego ustawienie oznacza, że sprite’y 0, 1 (albo 2, 3 lub 4, 5 lub 6, 7) będą, w celu optymalizacji kolorów, traktowane jako złączona para, i jako taka będą miały 4 bitową głębię (nie złączone są dwubitowe). Parzysty numer (większy) sprite’a zawiera bity o większej ważności binarnej.

  Podczas trybu łączonego, wszystkie złączone sprite’y są przemieszczane w pionie i poziomie razem przez procesor. Pozwala to na zwiększenie ilości możliwych kolorów w ramach jednego (złożonego z dwóch) sprite’a. Sprite’y, chociaż złączone, mają cały czas możliwość poruszania się niezależnie. Zwiększona ilość kolorów pozostaje widoczna jedynie gdy oba sprite’y nakładają się na siebie i poruszają się synchronicznie.

- Bity 6 – 3 są zarezerwowane dla przyszłego użycia (dla kompatybilności powinny być skasowane, czyli zawierać 0)
- Bit 2 jest ósmym bitem wartości początkowej pionowej, bit V8.
- Bit 1 to ósmy bit końcowej pozycji pionowej, bit V8.
- Bit 0 to zerowy bit początku poziomego, bit H0.

Renestry pozycji i kontroli dla pozostałych sprite'ów działają w identyczny sposób jak opisane wyżej rejestry dla sprite'a o numerze 0. Nazwy tych rejestrów podane są poniżej:

SPR1POS	SPR1CTL
SPR2POS	SPR2CTL
SPR3POS	SPR3CTL
SPR4POS	SPR4CTL
SPR5POS	SPR5CTL
SPR6POS	SPR6CTL
SPR7POS	SPR7CTL

REJESTRY DANYCH

Poniższe rejestry, choć zdefiniowane w przestrzeni adresowej procesora głównego, normalnie używane są prez procesor graficzny. Przechowują rejestry dla danych otrzymanych w cyklach DMA.

SPR0DATA,	SPR0DATB	rejestry danych sprite'a 0
SPR1DATA,	SPR1DATB	rejestry danych sprite'a 1
SPR2DATA,	SPR2DATB	rejestry danych sprite'a 2
SPR3DATA,	SPR3DATB	rejestry danych sprite'a 3
SPR4DATA,	SPR4DATB	rejestry danych sprite'a 4
SPR5DATA,	SPR5DATB	rejestry danych sprite'a 5
SPR6DATA,	SPR6DATB	rejestry danych sprite'a 6
SPR7DATA,	SPR7DATB	rejestry danych sprite'a 7

Podsumowanie informacji o rejestrach kolorów sprite'a

Słowa danych sprite'a są używane do wyboru koloru poszczególnych jego punktów przy użyciu rejestrów kolorów w sporób przedstawiony w poniższych tabelach.

W tabeli 4-6, dla odpowiednich wartości bitowych pojedynczych sprite'ów umieszczonych w kolumnie „wartość", kolor będzie pobrany z odpowiadającego tej wartości rejestru umieszczonego w kolumnie „rejestr koloru".

Pojedyncze sprite’y		Rejestr koloru
Sprite	**Wartość**	
0 lub 1	00	Nie używany *
	01	17
	10	18
	11	19
2 lub 3	00	Nie używany *
	01	21
	10	22
	11	23
4 lub 5	00	Nie używany *
	01	25
	10	26
	11	27
6 lub 7	00	Nie używany *
	01	29
	10	30
	11	31

* wybiera kolor „przeźroczysty”

Tabela 4-6: Rejestry kolorów dla pojedynczych sprite’ów

W tabeli 4-7 poniżej pokazano kombinacje bitów i odpowiadające im numery rejestrów dla złączonych sprite’ów.

Złączone sprite’y	
Wartość	**Rejestr koloru**
0000	Kolor „przeźroczysty”
0001	17
0010	18
0011	19
0100	20
0101	21
0110	22
0111	23
1000	24
1001	25
1010	26
1011	27
1100	28
1101	29
1110	30
1111	31

Tabela 4-7: Rejestry kolorów dla złączonych sprite’ów

ODDZIAŁYWANIE SPRITE'ÓW Z INNYMI OBIEKTAMI

Playfieldy i sprite'y współdzielą jeden ekran. Rozdział 7 p.t. "Układy kontroli systemu," opisuje w jaki sposób można playfieldom ustawić różne priorytety w stosunku do prite'ów oraz jak playfieldy mogą „kolidować" (nachodzić) ze sprite'ami lub ze sobą.

> *Sprite'y na układach ECS*. Informacje dotyczące sprite'ów w rozszerzonych układach specjalizowanych Amiga (ECS) takie jak sprite'y w rozdzielczości SuperHires oraz pozycjonowanie sprite'ów w rozdzielczości SuperHires znajdują się w „Dodatku C".