A T A R I
I N T E R N

I D E A A T A R I

Mikrokomputery Atari są projektowane z przeznaczeniem dla amatorów. Właściwości jego budowy i oprogramowania umożliwiają różnorodne zastosowania. Dzięki właściwościom obrazu i dźwięku urządzenia Atari nadają się szczególnie do gier telewizyjnych (video). Naturalnie, mogą one także służyć do nauki BASIC-u, ale jest to BASIC bardziej przystosowany do zastosowań rozrywkowych niż do naukowych. Jest również możliwe, z dodatkowymi modułami, nauczanie innych języków programowania, jak na przykład Asembler, Pilot, Pascal lub Action!, także dla języka Forth Atari nadaje się znakomicie. Dalszym zastosowaniem jest wykorzystanie Atari jako komputera uczącego. Można też wykorzystać Atari do planowania budżetu. Także krótkie listy dzięki Atari pisze się lekko i łatwo,

Jednakże Atari ma swoje ograniczenia. Duże ilości danych są przez niego trudno przetwarzane, gdyż stacja dysków ma niedużą pojemność i szybkość. Uciążliwe jest takie jego wykorzystanie jako systemu tekstowego, ponieważ Edytor Obrazu pozwala na umieszczenie tylko 80 bajtów w linii. Dla komercjalnych zastosowań nadaje się więc Atari tylko warunkowo. Także dla języków programowania "zorientowanych na dyski" jak PASCAL, FORTRAN lub COBOL praca na Atari jest prawie niemożliwa.

Aby zrozumieć Atari w szczegółach, powinno się mieć przed oczami jego ideę i podstawową strukturę. Dopiero wtedy, gdy się to zna, właściwe jest poznawania szczegółów układów wielkiej skali integracji (POKEY, ANTIC, GTIA i PIA) oraz systemu operacyjnego.

Komputery Atari są wyposażone w szeroko rozpowszechniony mikroprocesor 6502.

Można zapewne zapytać, wobec dostępności na rynku innych procesorów, dlaczego zastosowano jeszcze ten procesor. Oczywiście 16- względnie 32-bitowe procesory są znacznie wydajniejsze. Są też w tej rodzinie mikroprocesorów takie, które współpracują z 8-bitową szyną danych, potrzebują więc zaledwie takiego oprogramowania jak 6502. Jednakże w czasie, gdy został zaprojektowany pierwszy mikrokomputer Atari, nie były one dostępne na rynku w wystarczającej ilości. Są one także jeszcze dziś znacznie droższe niż 6502.

Inne będące w tym czasie da dyspozycji 8-bitowe mikroprocesory, jak na przykład Z80, 8080/8085, 6800/6802 czy też 6809, albo nie miały wyraźnej przewagi nad 6502, albo były relatywnie droższe.

Przy konstruowaniu modeli 600XL i 800XL zastosował Atari ponownie procesor 6502. Jest to rozwinięta wersja znanej CPU, oznaczona 6502C, w pełni kompatybilna programowo. Zrobiono tak przede wszystkim ze względu na możliwie największą kompatybilność ze starymi modelami. Pozwala to bezpośrednio zastosować wiele programów lub ich części.

Zaletą 6502 jest poza tym to, że jego takt zegarowy składa się z dwóch faz jednakowej długości. Procesor zawsze podczas drugiej fazy łączy się z pamięcią lub innymi układami. Połączenie następuje więc zawsze w tym samym momencie cyklu. Upraszcza to budowę innych elementów systemu, które mogą korzystać z pamięci bez pośrednictwa CPU. Tym szczegółem 6502 odróżnia się istotnie od innych mikroprocesorów, jak na przykład Z80, 8080 i 8085.

Procesor 6502 pracuje w Atari z częstotliwością 1, 77 MHz. Częstotliwość ta jest znacznie niższa od zwykłe stosowanej częstotliwości pracy procesorów tej klasy.

Normalnie wejście następuje przez klawiaturę. Ma ona formę przeważnie używaną w USA (QWERTY). Niektóre znaki specjalne znajdują się w Atari w innych miejscach niż w innych komputerach. Przegłosy i inne stosowane w

obcych językach litery i formy liter nie są uwidocznione na klawiaturze. W innych komputerach tej klasy jest to zwykle stosowane. Atari 400 posiadał klawiaturę foliową. Większy model – Atari 800 – miał klawiaturę z normalnymi klawiszami (podobnie jak maszyna do pisania). Nowe modele Atari – 600XL i 800XL – mają także normalne klawisze, Klawiatury tych modeli różnią się tylko nieznacznie, inaczej niż u innych producentów, gdzie każdy nowy komputer ma inną klawiaturę. Oprócz głównego pola klawiatury z prawej strony urządzenia znajdują się dodatkowe klawisze SELECT, OPTION i START.

Do obsługi klawiatury nie zastosowano żadnego zwykle używanego układu. Zamiast tego użyto w Atari układ POKEY, który oprócz klawiatury służy także do obsługi wyjścia/wejścia szeregowego i wytwarzania dźwięku, Wynik, wskazujący który klawisz został naciśnięty, przesyła on do specjalnego rejestru. Klawiatura, która jak i w innych systemach zbudowana jest jako matryca, nie jest jednak odczytywana rzędami przez CPU. Oszczędza to czas CPU. W momencie naciśnięcia klawisza POKEY wywołuje przerwanie (IRQ). W ten sposób jest możliwa obsługa klawiatury przez przerwania. Klawisze dodatkowe SELECT, OPTION i START są obsługiwane przez inny układ wielkiej skali integracji – GTIA.

Standardowym wyjściem Atari jest monitor. Jako monitor może służyć praktycznie każdy odbiornik telewizyjny. W komputerach Atari jest wbudowany modulator, wystarczy więc normalne wejście antenowe i nie jest wymagane w telewizorze wejście video. Pożądany jest telewizor kolorowy dysponujący paletą 128 kolorów, System operacyjny pozwala na uzyskanie do 320 x 192 punktów lub 24 wierszy po 40 znaków. Największą rozdzielczość uzyskuje się przy 384 punktach w poziomie i około 210 w pionie (na ekranie).

Tworzenie całego obrazu przebiega w układzie, który jest stosowany tylko przez Atari. Układ ten nazywany jest ANTIC (Alphanumeric Television Interface Controller). ANTIC jest drugim mikroprocesorem, który jest sterowany własnym zestawem rozkazów, Przez program ANTIC–u, który można umieścić w dowolnym miejscu pamięci, są definiowane potrzebne obrazy. ANTIC dysponuje ponad 14 różnymi stanami wyświetlania obrazu. Stany te można na ekranie plastycznie łączyć. Są to zarówno stany dla tekstu, jak i dla różnorodnych znaków (na przykład do gier), a także liczne stany dla przedstawiania pojedynczych punktów z wykorzystaniem wysokiej rozdzielczości.

Szerokość obrazu może być ustawiana na trzy różne sposoby, Odróżniają się one maksymalną poziomą rozdzielczością obrazu: 256, 320 lub 384 punkty. W obramowaniu obrazu występuje jeden (programowany) kolor.

ANTIC czyta wszystkie dane, których potrzebuje do utworzenia obrazu, bezpośrednio z pamięci z pominięciem CPU. Ten sposób dostępu zwany jest DMA (Direct Memory Access – bezpośredni dostęp do pamięci). Podczas korzystania przez ANTIC z pamięci w trybie DMA CPU jest zatrzymywana. W ten sposób zmniejsza się efektywna szybkość pracy CPU. W większości stanów przepustowość CPU jest pomimo to większa niż w systemach o częstotliwości 1 MHz z jednym taktem.

ANTIC nie tworzy sam informacji o kolorach w sygnale wizyjnym, który przez modulator jest przesyłany do telewizora. Przekazuje on informacje o obrazie, które przez DMA pobrał z pamięci, do innego układu Atari. Układ ten zwany jest GTIA (Graphic Television Interface Adapter). GTIA zawiera między innymi dziewięć rejestrów koloru, dzięki którym użytkownik może wybrać z palety 128 barw kolor, który chce otrzymać na ekranie. GTIA otrzymuje od ANTIC–a informacje, który kolor, tzn. kolor z którego rejestru, ma pokazać na ekranie. Dzięki tej idei! otrzymuje się paletę barw, którą przeważnie można uzyskać tylko w profesjonalnych systemach CAD i którą trudno porównać z innymi mikrokomputerami.

Szczególnie interesujące jest też, że budowa ANTIC–a umożliwia przesuwanie obrazu o pojedyncze punkty zarówno poziomo, jak i pionowo, Pamięć ekranu można bez żadnych tricków umieścić w dowolnym miejscu pamięci, Generator znaków zasadniczo też może się znajdować w dowolnym miejscu pamięci, musi jednakże, zależnie od stanu ANTIC–a (zależnie od

tego, czy ma on wielkość 512 czy 1024 bajty), zaczynać się na granicy 512 bajtów lub 1 KB.

Z takim rozwiązaniem tworzenia obrazu jest Atari sprawniejszy niż większość innych systemów tej klasy. Przede wszystkim idea drugiego mikroprocesora, idea ANTIC-u, umożliwia nieosiągalną w inny sposób różnorodność tworzenia obrazu, Zasada ta daje też, pomimo przerywania pracy CPU przez pracujący w trybie DMA ANTIC, relatywnie dużą szybkość pracy przy częstotliwości zegara 1, 77 MHz. Ponadto można podczas procedur zawierających dużą ilość obliczeń wybrać tryb pracy ANTIC-u, który powoduje małą ilość przerwań przez DMA lub nawet wyłączyć obraz. Podwyższa to znacznie wydajność CPU.

Atari posiada poza tym bardzo różnorodne możliwości dźwiękowe. Steruje tym układ, który oprócz tego służy do obsługi klawiatury - POKEY (POtentiometer and KEYboard Integrated Circuit), POKEY dysponuje czterema generatorami dźwięku, które mogą pracować zarówno razem, jak i oddzielnie. Ponadto istnieją różnorodne możliwości zniekształceń dźwięku, szumów itp. Można też programować filtr wysokich częstotliwości. POKEY pozwala na przykład dołączyć do Atari klawiaturę syntetyzera. Dźwięki z POKEY-a są przez modulator przesyłane do telewizora. Jako wyjście służy więc normalny głośnik telewizora. Jest też możliwe zmieszanie sygnału POKEY-a z zewnętrznym sygnałem doprowadzonym na wejście Atari.

Wielką zaletą jest fakt, że można programować dźwięki w BASIC-u bez używania instrukcji POKE, jak w konkurencyjnych modelach. Bezpośrednie programowanie dźwięku w kodzie maszynowym jest za to bardziej skomplikowane. Możliwości dźwiękowe, które ofiaruje Atari, są przez to porównywalne z możliwościami konkurencyjnych modeli. Tworzenie dźwięków jest oparte na automatycznym liczniku, który w momencie zerowania przerywa pracę CPU. W procedurze przerwania można zmienić częstotliwość, głośność lub inny parametr dźwięku.

Nowe modele Atari (600XL, 800XL) różnią się wieloma szczegółami od starych 400 i 800. Częściowo zostały zmienione adresy hardware'u. Programy w BASIC-u mogą być stosowane bezpośrednio lub wymagają tylko niewielkich modyfikacji.

W starych modelach Atari są 4 porty (gniazda) dżojstików. Każdy port pozwala na przyłączenie jednego dżojstika, dwóch potencjometrów (paddle) lub jednego pióra świetlnego. Gdy pióro świetlne zostanie zatrzymane w jakimś miejscu ekranu, możliwe jest określenie jego pozycji. Jest to wykonywane przez ANTIC. Obsługa dżojstika jest prowadzona przez kolejny układ Atari - PIA (Peripherial Interface Adapter). Przycisk strzelania dżojstika jest obsługiwany przez wejście GTIA zwane trigger input. Analogowy sygnał określający położenie potencjometru jest zamieniany na cyfrowy przez POKEY.

W nowych modelach Atari znajdują się tylko dwa gniazda dżojstików. Stanowi to tylko nieznaczną wadę, ponieważ bardzo mało gier i innych zastosowań wymaga czterech dżojstików. Uzyskane w ten sposób wolne wejścia PIA wykorzystano częściowo dla tzw. MMU (Memory Management Unit) - jednostki zarządzania pamięcią.

W przeciwieństwie do starych modeli przestrzeń pamięci w modelach 600XL i 800XL jest częściowo zdublowana. Pozwala to na przykład w Atari 800XL (lub w Atari 600XL z modułem 64 KB RAM) wyłączyć system operacyjny. W tym miejscu znajduje się wtedy pamięć RAM. Jest na przykład możliwe wczytanie do nowych modeli systemu operacyjnego Atari 400/800. Poza tym nowe modele dysponują programem testującym zaszytym w ROM, który przy normalnej pracy jest wyłączony i uruchamia się, gdy przy włączaniu komputera naciśnięty jest klawisz "OPTION". Program ten umożliwia przetestowanie pamięci, klawiatury, obrazu i dźwięku.

Różne obszary pamięci wybiera się przy pomocy portów PIA, które w starych modelach były przeznaczone dla trzeciego i czwartego dżojstika. MMU

włącza, zależnie od sytuacji, poszczególne elementy systemu sprzętowego. MMU jest to układ PAL (Programmable Array Logic) specjalnie programowany dla Atari. W swoich funkcjach ten układ jest podobny do zestawu zwykłych układów TTL.

Aby wyeliminować stratę czasu na ładowanie programu a dysku lub kasety. Atari posiada gniazdo dla modułów ROM (kartridży). W tym gnieździe można umieszczać moduły posiadające do 16 KB pamięci. Moduły te zapewniają doskonałe zabezpieczenie danych m.in. przez prawie nieograniczoną trwałość. Gdy moduł jest w szczelinie wyłączona jest pamięć RAM (w dwóch blokach po 8 KB) zasłonięta przez pamięć modułu. Oprócz tego system operacyjny automatycznie uruchamia program zawarty w module. Do tego jeszcze w gnieździe jest wprowadzenie do bloku 256 bajtów (jedna strona), znajdującego się w obszarze innych układów I/O. Możliwe jest również wyłączenie przez moduł różnych obszarów ROM. Oczywiście można też do gniazda przyłączyć płytę z układami peryferyjnymi. W Atari 400 i 800 jest to kłopotliwe, gdyż w czasie pracy klapka nad szczeliną musi być zamknięta. W nowych modelach nie jest to konieczne. Może przy tym nastąpić pogorszenie jakości obrazu spowodowane nienajlepszym ekranowaniem. Efekt ten jest różny w różnych egzemplarzach. Zależy on przede wszystkim od tego, jakie urządzenia pracują z komputerem.

Początkowo komputery Atari posiadały tylko system operacyjny. BASIC, który był zapisany w module, musiał być dodatkowo dołączany. Atari wychodziło oczywiście z założenia, że nie każdy chce mieć BASIC. Później moduł BASIC-u był dostarczany w komplecie. Każdy, kto kupował komputer Atari 400 lub 800, otrzymywał moduł BASIC-u.

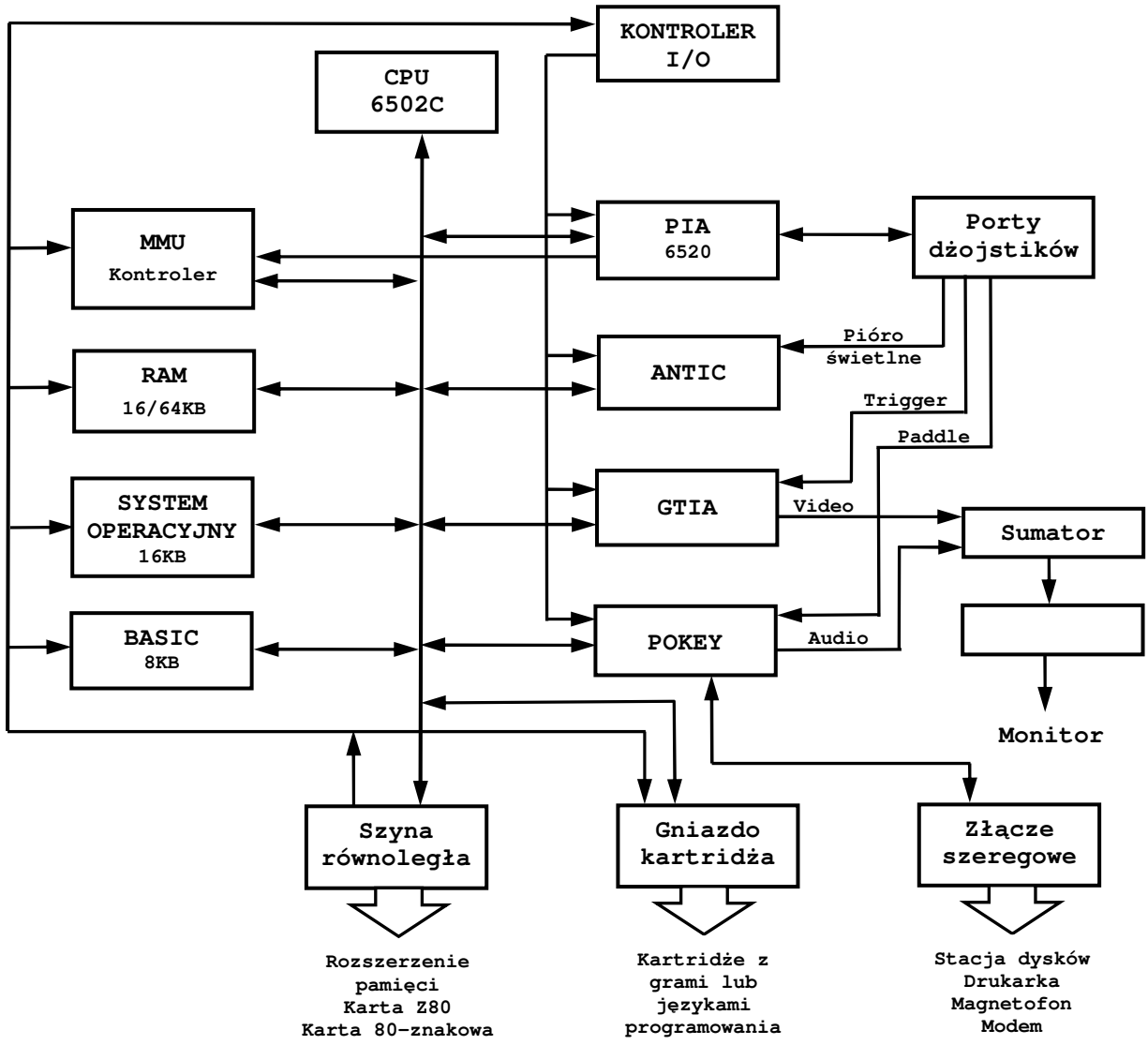
Nowe komputery Atari 600XL i 800XL posiadają wbudowany BASIC. Jest on całkowicie kompatybilny z wersją modułową. Programy mogą więc być używane bez zmian, o ile nie zawierają procedur systemu operacyjnego lub jego adresów. Jest też możliwe wyłączenie BASIC-u z pomocą MMU. Jeżeli moduł pokrywa obszar pamięci zawierający BASIC, to jego włączenie powoduje wyłączenie BASIC-u. W ten sposób możliwe jest użycie zarówno innego języka programowania, jak i rozszerzonej wersji BASIC-u w module.

Stacja dysków, drukarka, magnetofon kasetowy i inne urządzenia zewnętrzne są przyłączane do wyjścia szeregowego. Kanał ten jest obsługiwany przez POKEY. Możliwe jest przesyłanie danych z szybkością do 19200 bitów/sek. Zamiana danych równoległych na szeregowe jest przeprowadzana przez POKEY. Pozwala to na szybsze przesyłanie danych niż w systemach wykorzystujących do zamiany CPU. POKEY może wysyłać do CPU sygnał przerwania gdy przesłane zostaną wszystkie bity z rejestru wyjściowego lub zapełniony zostanie rejestr wejściowy. Te możliwości POKEY-a pozwalają na przekazywanie danych i jednoczesne wykonywanie przez CPU innych zadań. Złącze szeregowe nie odpowiada niestety żadnemu zwykłemu standardowi. Zasadniczo pozwala się ono programować jak zwykłe złącze V24, lecz na przykład nie jest możliwa quasi-jednoczesna praca dysków. Także gniazdo złącza szeregowego nie odpowiada żadnym amerykańskim ani europejskim normom, dlatego trudno jest innym producentom wykorzystać złącze dla własnych produktów.

W starych modelach szyna systemu nie była wyprowadzona na zewnątrz. Wszystkie zmiany i rozszerzenia musiały być robione w sprzęcie. W nowych modelach szyna jest wyprowadzona na 50-stykowe złącze PBI. Platynowe kontakty w odstępie 2,54mm (0,1") pozwalają na bezpośrednie przyłączenie urządzeń. Na PBI wyprowadzone są wszystkie sygnały danych i adresów, wiele sygnałów sterujących oraz sygnały dla przyłączenia pamięci dynamicznej RAM.

Dzięki temu możliwe jest rozszerzenie Atari 600XL lub, w ograniczonym zakresie, 800XL. Ograniczeniem jest bowiem brak wyprowadzenia napięcia +5V na PBI, ale jest to łatwe do naprawienia. Przykładami są: rozszerzenie pamięci RAM, karta z procesorem Z80 dla zastosowania CP/M, bezpośrednie dołączenie stacji dysków, karty 80-znakowej, różnych złączy, sterowań itd.

SCHEMAT BLOKOWY KOMPUTERA ATARI XL



Atari 400 i 800 dysponują pamięcią ROM 10 KB w kilku układach. Znajduje się w nich głównie system operacyjny i generator znaków. W nowych modelach zastosowano nowoczesną pamięć ROM 16 KB. Tak powiększona pamięć ROM zawiera przede wszystkim program testujący, drugi generator znaków i powiększony system operacyjny.

Już w pierwszych modelach Atari zastosowano prawie prawdziwy system operacyjny. W przeciwieństwie do wielu tak zwanych systemów operacyjnych, system Atari składa się nie tylko z elementarnych procedur, ale tworzy wydajny system procedur operacyjnych, mający uniwersalne zastosowanie,. Na przykład komunikacja z urządzeniami zewnętrznymi jest prowadzona przez jedną uniwersalną procedurę, wywoływany program przekazać musi do tzw. bloku kontroli I/O dane źródłowe względnie wynikowe i wszystkie inne parametry Idea ta najpierw została zastosowana w systemach zorientowanych na dyski jak CP/M, MSDOS, UNIX lub QNX. W systemach operacyjnych innych mikrokomputerów jest zwykle stosowana dla każdego urządzenia I/O (monitor,

klawiatura, drukarka, kaseta, dysk itd.) oddzielna procedura. Każda z tych procedur potrzebuje danych w innym formacie.

Wejścia i wyjścia Atari są zasadniczo zdolne do przerwań, jest to w pełni poparte w systemie operacyjnym. Możliwe jest we własnych programach maszynowych wprowadzanie lub wyprowadzanie danych podczas gdy procesor wykonuje inne zadania. Tych możliwości systemu operacyjnego nie można jednak wykorzystać w BASIC-u. Pozwala to na przykład programować system tekstowy, przy którym użytkownik może układać teksty praktycznie bez względu na operacje czytania i zapisywania dysków lub funkcje systemowe wyszukiwania i wymiany danych. Przesyłanie danych jest wykonywane przez system operacyjny w czasie przerwań. Aby nie pozwolić na zagubienie żadnego sygnału z klawiatury, POKEY musi zawsze, gdy zostanie naciśnięty klawisz, wywołać przerwanie. Podczas procedury przerwania wszystkie sygnały z klawiatury są przesyłane do bufora klawiatury. Według tego wzoru tworzone jest wiele programów.

Normalna praca Atari jest przerywana co 1/50 sekundy. Przerwanie jest wywoływane podczas pionowej synchronizacji obrazu. W procedurze tej jest wypełniane kilka zadań: zwiększana lub zmniejszana jest zawartość komórek pamięci tworzących liczniki programowe, przy pomocy których sterowany jest przebieg programu. Kilka z nich tworzy zegar czasu rzeczywistego RTC. Dalszym zadaniem procedury przerwania synchronizacji jest udostępnienie tzw. rejestru cieni. Ponieważ wiele rejestrów jest tylko zapisywanych lub odczytywanych, przez system operacyjny tworzone są w RAM rejestry cieni. Zawartość tych rejestrów jest aktualizowana 50 razy na sekundę. Rejestr cieni można w przeciwieństwie do większości rejestrów sprzętowych modyfikować rozkazami INC i DEC (zwiększ i zmniejsz). Jednakie trzeba przy tym zawsze uważać, gdyż zawartość rejestru niekoniecznie musi być aktualna.

System operacyjny komputerów Atari jest w zasadzie przejrzyście programowany. W Atari daremnie szukać, pomijając bardzo małe wyjątki, stosowanej w wielu innych urządzeniach tej klasy zasady "podstawa: działa". System operacyjny Atari zawiera relatywnie mało, lecz za to uniwersalnych podprogramów. Wszystkie ważne procedury, także wewnątrz systemu, osiągalne poprzez tzw. tabelę wektorów, w której znajdują się adresy startowe poszczególnych procedur. Dzięki temu jest relatywnie łatwo zarówno wykorzystywać system operacyjny w programach maszynowych, jak i zmieniać go.

W nowych modelach Atari system operacyjny jest częściowo zmieniony i ulepszony. Posiada kilka nowych procedur. Tabela wektorów zasadniczo nie jest zmieniona, ma jedynie dodane kilka wektorów, Naturalnie trzeba też uwzględnić w systemie operacyjnym zmianę sprzętu (na przykład MMU i wbudowany BASIC). Powoduje to, że bardzo wiele procedur w nowym systemie jest nieznacznie zmodyfikowanych i znajduje się w innych miejscach. Także wiele komórek pamięci wykorzystywanych przez system operacyjny leży w innych miejscach. Powoduje to, że niewiele starych programów działa na nowych modelach, bez większych problemów działają tylko programy, które wykorzystują procedury systemu operacyjnego poprzez tabelę wektorów lub podobną specyfikację systemu, Programy zawierające tricki najczęściej trudno dopasować do nowych modeli.

H A R D W A R E

W Atari 600XL/800XL obok ogólnie dobrej idei także hardware robi wrażenie dobrze przemyślanego i uporządkowanego, szczególnie, że w tej klasie cenowej nie wszystko może być zrobione całkiem dobrze. Główna część struktury w 600XL/800XL, tak jak w starych modelach 400/800, jest utworzona ze specjalnych układów wielkiej skali integracji ANTIC, POKEY i GTIA oraz standardowych układów CPU i PIA.

Jako pamięć wykorzystano w obu nowych modelach ROM 16 KB dla systemu operacyjnego i ROM 8 KB dla wbudowanego BASIC-u.

Jedyną dużą różnicą między 600XL a 800XL jest to, że 800XL ma powiększoną do 64 KB pamięć RAM. To powiększenie nie polega na dodaniu nowych układów, lecz na nieco innym rozwiązaniu.

Podczas, gdy 600XL dla swojej 16 KB RAM wykorzystuje 2 układy, każdy zawierający 16384 słowa 4-bitowe, to 800XL jest wyposażony w 8 znacznie konwencjonalniejszych i przez to tańszych układów, zawierających po 65536 słów 1-bitowych,

Sami już zauważyliście po włączeniu, że Atari nie działa natychmiast.

Prąd płynący z zewnętrznego zasilacza od razu za włącznikiem rozdziela się na trzy tory, które wzajemnie sprzężone są przez indukcyjności i kondensatory blokowe.

Pierwszy tor zasila głównie układy VLSI i bezpośrednio z nimi związane układy buforów. Drugi tor prowadzi do pozostałych układów logicznych i pamięci RAM. Trzeci tor jest przewidziany wyłącznie dla modulatora RF. Jest to też najlepiej odfiltrowana przez indukcyjność gałąź, co pozwala wyeliminować zakłócenia zarówno sygnałów cyfrowych przez sygnał w.cz., jak i odwrotnie.

Sercem komputera jest generator sygnału zegarowego. Generuje on stabilizowaną kwarcem częstotliwość podstawową 3.546894 MHz, która bezpośrednio wykorzystywana jest do wytwarzania taktu koloru.

Połowa częstotliwości podstawowej (1.773477 MHz) służy do taktowania pracy CPU i steruje wszystkimi innymi funkcjami. Drugi, niewykorzystany przez CPU sygnał służy do wytworzenia przez tzw.

podwajacz napięcia odniesienia dla CADJ w GTIA. Jest to w tych modelach niezbędne, gdyż istniejącego w starych modelach 400/800 napięcia 12V nie ma z powodu zastosowania nowocześniejszych pamięci RAM.

Aby ustawić system w stan uporządkowany, potrzebny jest impuls Reset. Jest on tworzony automatycznie przy włączaniu przez układ RC. Znajdujący się tam kondensator może być rozładowany do masy poprzez zewnętrzne przewody. Przewody te prowadzą do klawisza RESET na konsoli. Przez ten sygnał są ponownie ustawiane układy ANTIC, PIA i CPU. Pozostałych układów sygnał ten nie dotyczy - są one ustawiane programowo.

CPU steruje poprzez szyny adresową, danych i kontrolną pozostałymi układami VLSI. Pozwala się ona jednak zatrzymać przez wysłany przez ANTIC sygnał HALT tak, że ustawia swoje wyjścia w stan wysokiej impedancji. Powoduje to zwolnienie szyn systemu, dzięki czemu na przykład ANTIC ma bezpośredni dostęp do pamięci (DMA).

Wszystkie pamięci (zarówno RAM jak i ROM) są sterowane przez kolejny układ LSI. Jest to programowany układ logiczny (PAL) i wypełnia m. in. zadania dekodera logicznego pamięci, jak również manewrowania blokami pamięci. Z tego powodu jest on określany jako Memory Management Unit (MMU).

Zanim przejdziemy do opisu poszczególnych wejść i wyjść, jeszcze jedna mała uwaga do składni opisu wyprowadzeń; po każdym sygnale jest

umieszczony za dwukropkiem poziom, przy którym ten sygnał jest aktywny. Jeżeli na przykład sygnał XYZ jest aktywny przy niskim poziomie, to piszemy !XYZ, w przeciwnym razie XYZ.

MMU posiada jako wejścia 5 największych bitów adresowych i pochodzącą z ANTIC-a informację odświeżania, która mówi, że przy następnym dostępie do pamięci chodzi o jej odświeżenie (RAM jest zbudowana z układów dynamicznych, które wymagają okresowego odświeżania zawartości).

Mając takie informacje MMU jest już w stanie podzielić cały obszar 64 KB pamięci na 32 duże bloki po 2 KB. Bloki zasadniczo wyglądają następująco:

blok 0 od 0000h do 07FFh
blok 1 od 0800h do 0FFFh
blok 2 od 1000h do 17FFh
blok 3 od F000h do F7FFh
.
.
.
blok 31 od FB00h do FFFFh.

MMU łączy 2 KB bloki w większe grupy, na przykład obszar od 0000h do 4FFFh (najniższe 16 KB), tworzący zawsze jeden blok, który zarówno w 600XL, jak i w 800XL zajęty jest przez RAM. Dalej następuje blok, który zasługuje na szczególną uwagę, a dalej od 5800h do 7FFFh znowu normalny obszar RAM, niezależnie od tego czy istnieją fizycznie układy RAM, czy też nie.

Blok 10 (5000h-57FFh) ma, jak powiedziano, specjalny status. Obszar ten jest normalnie przewidziany dla RAM, można jednakże ten blok odłączyć. Służy do tego połączenie portu PB7 PIA z wejściem MAP MMU. Jeżeli na tym wejściu jest 1, to RAM pozostaje włączona. Jeżeli zmieni się na zero, to blok 10 RAM jest odłączany i odwzorowywany jest tu obszar ROM leżący poza układami I/O w bloku 26 (D000-D7FFh). Włączamy w ten sposób 2KB program z procedurą testowania, co następuje gdy przy włączaniu komputera naciskamy klawisz OPTION bez dołączonego kartridża lub dysku.

Jeszcze dwa sygnały przełączania prowadzą z portu B do MMU: port PB0 jest połączony bezpośrednio z ROM/!RAM w MMU i powoduje tam, że przy niskim stanie wyłącza system operacyjny leżący w obszarze C000-CFFFh i D800h-FFFFh oraz włącza leżące tam ewentualnie RAM. Przy używaniu tych adresów trzeba zwrócić uwagę na to, że system runie, jeżeli w tym miejscu nie ma RAM lub nie ma tam żadnego rozsądnego systemu.

Aby przy uruchomieniu zawsze został aktywny ROM, impuls RESET automatycznie ustawia PB0 w stan wysoki.

Trzecie i ostatnie połączenie między MMU a PIA jest wykonane poprzez port PB1, z którego sygnał dochodzi bezpośrednio do !BE. Gdy sygnał ten jest 0, to obszar 8 KB od A000h do BFFFh jest zajęty przez wewnętrzny BASIC, w przeciwnym razie pozostaje wolny do innych zastosowań.

To "inne" jest normalnie pamięcią RAM, która tym miejscu się znajduje w 800XL lub w 600XL z dołączoną płytą 64 KB.

MMU ma jednakże jeszcze dwa dalsze wejścia, którymi można przełączać obszary pamięci. Służą one do sprzętowego przyłączenia modułów.

Jeżeli w obszarze A000h-BFFFh znajduje się kartridż (np. stary kartridż BASIC-u lub Atari-Assembler lub gra jak STARRAIDER itp.), to powiadamia on MMU poprzez RD5. Jeżeli moduł znajduje się w obszarze 8000h-9FFFh, to informuje MMU poprzez RD5. MMU odpowiada na to w taki sposób, że wyłącza leżące tam obszary RAM,

Ostatnim wejściem jest !MPD. To zewnętrzne sterowanie powodujące, że wyłącza się obszar D800h-DFFFh (ROM procedur matematycznych Math Pack

Disable) w obszarze systemu operacyjnego. Jest to przewidziane dla dalszych uzupełnień hardware'u.

Oprócz tych wszystkich wejść MMU posiada oczywiście także wyjścia, które służą jako wyjścia selekcyjne.

Gdy RD4 lub RD5 jest w stanie wysokim, to na wyjściu !S4 względnie !S5 pojawia się stan niski, co powoduje aktywowanie odpowiedniego kartridża (8000h-9FFFh lub A000h-BFFFh).

ROM systemu operacyjnego jest wybierana przez wyjście !OS wg. następujących kryteriów:

MPD:L	ROM/!RAM	Obszar adresowy	Wyjście !OS
H	H	C000h-CFFFh	L
H	H	D800h-FFFFh	L
X	L	Ogólnie nieaktywne	H
L	H	C000h-CFFFh	L
L	H	D800h-DFFFh	H
L	H	E000h-FFFFh	L

Wyjście !I/O powoduje że przy odwołaniu się do obszaru D000h-DFFFh jest wywoływana mapa pamięci I/O. Mapa pamięci oznacza, że dla układów I/O nie jest ustanowiona oddzielna przestrzeń adresowa, lecz jest do tego wydzielona część przestrzeni pamięci.

Pamięć RAM może tam, gdzie to konieczne, być wyłączana przez sygnał z wyjścia !CasINH. Wyjście to powoduje przy odpowiednim dekodowaniu, że tzw. Column Adress Strobe (!CAS) powodujący ostateczny wybór komórki pamięci zostaje ukryty. Jest to wykonywane w ten sposób, ponieważ pamięć RAM musi być odświeżana pomimo jej odłączenia pamięci.

Na przejście sygnału z MMU do układów RAM potrzeba trochę czasu. Ogólnie można powiedzieć, że sterowanie pamięci dynamicznych jest skomplikowane z powodu dokładnego przestrzegania zależności czasowych wymaganych sygnałów.

W 600XL/800XL ten problem został rozwiązany przez linię opóźniającą, która opóźnia sygnał na wyjściach 25-260nsek.

Na całą logikę dekodowania działają dwa sygnały wyboru: !CasInh z MMU i doprowadzany z zewnątrz !ExtSel, który przy niskim poziomie uniemożliwia sterowanie wewnętrznej RAM. Obok sterowania pamięcią daje to jeszcze dwa inne duże obszary połączeń wewnątrz ATARI. Jednym jest wytwarzanie sygnałów koloru, obrazu i dźwięku, o których nie ma nic więcej szczególnego do powiedzenia, gdyż są zbudowane na standardowym sprzęcie, jak na przykład przetwornik cyfrowo-analogowy wartości luminancji GTIA lub 3-stopniowy wzmacniacz w.cz. dla kompletnego sygnału FBAS. Poza tym nie jest przewidywane w Atari wykorzystanie go dla zewnętrznego sprzętu.

Drugim obszarem jest I/O:

Tak więc istnieje multiplekser/demultiplekser do sterowania klawiatury, analogowe i cyfrowe wejścia POKEY-a, port A dla dżojstików i paddle'ów, a także dwa złącza szyny systemu.

Wejścia analogowe i cyfrowe są odkłócone kondensatorami włączonymi między masę a suwak potencjometru.

W przeciwieństwie do starych modeli 400/800 można zauważyć, że wejścia i wyjścia PIA nie są zabezpieczone szeregowymi rezystorami przed przeciążeniem. Ma to tę wadę, że łatwo moszna uszkodzić wejścia podczas eksperymentowania, ale za to zaletą jest brak problemów przy pracy z normalnym obciążeniem.

Wejścia dżojstików i paddle'ów są podłączone do 9-pinowych męskich gniazd D-SUB. Połączenia te są przedstawione w poniższej tabeli:

Pin #	JOY1/JOY2	Przeznaczenie
1	PA0/PA4	Naprzód
2	PA1/PA5	Wstecz
3	PA2/PA6	W lewo (dżoj) lewy skraj (pad)
4	PA3/PA7	W prawo (dżoj) prawy skraj (pad)
5	Pot1/Pot3	Suwak lewego potencjometru (pad)
6	Tr0/Tr1	Przycisk (dżoj) lub pióro świetlne
7	+5V	Zasilanie
8	GND	Masa
9	Pot0/Pot2	Suwak prawego potencjometru (pad)

Pióro świetlne można dołączyć do któregokolwiek z dwóch wejść. Przy dołączeniu dwóch piór świetlnych jednocześnie negacje sygnałów wejściowych są łączone przez operację OR (a więc sygnały oryginalne przez AND).

Na szczególną uwagę zasługują oba bezpośrednie złącza szyny CPU. Jedno to gniazdo kartridża, które zawiera 13 złącz adresowych (dla 8 KB), 8 złącz danych, masy, napięcia +5V, !S4, !S5, RD4 i RD5, !CCTL (!D5XX).

Dla właściwego transferu danych z układów znajdujących się w gnieździe kartridża są tam jeszcze sygnały CPU 6502 R/!W oraz Φ2.

Na złączu PBI jest znacznie więcej sygnałów. Obok wszystkich 16 bitów adresowych i 8 bitów danych jest kilka styków masy i dwa styki napięcia zasilania +5V. Tu mała uwaga: napięcie +5V wyprowadzone jest tylko w 600XL, do którego to modelu produkowany był moduł rozszerzenia pamięci Atari 1064. W 800XL styki te są wolne.

Sygnał !ExtSel w stanie aktywnym blokuje wewnętrzną RAM i służy głównie dla dołączenia RAM do 600XL.

!RESET jest asynchronicznym sygnałem wyjściowym, który ustawia wewnętrzne układy.

!IRQ jest wejściem połączonym z końcówką IRQ CPU. Należy zwrócić uwagę na to, że wszystkie inne układy korzystające z przerwań jak POKEY i PIA również mają wyjścia dołączone do tej linii i że to wejście jest wewnętrznie spolaryzowane do +5V przez rezystancję 3 kΩ.

R/!W jest buforowanym sygnałem, który odpowiada sygnałowi R/!W CPU. Przerzutnik bufora jest czasowo zależny od Φ2 aż do momentu, w którym nastąpi połączenie z !CasInh i !ExtSel i można ich użyć bezpośrednio do sterowania układów pamięci.

!RDY jest przyłączony do wyprowadzenia CPU o tej samej nazwie, ale trzeba zwrócić uwagę na to, że ANTIC musi mieć zawsze dostęp do CPU przez swoje wyjście także o tej samej nazwie. Przy używaniu tego łącza obowiązują takie same uwagi, jak dla łącza IRQ:L,

!CasInh jest, jak już to omówiono przy MMU, sygnałem wyjściowym do wyłączenia zewnętrznej pamięci dynamicznej RAM. Może on być także zastosowany do pamięci statycznych, wymaga wtedy jednak uprzedniej przemiany.

!CAS jest czystym sygnałem czasowym, kiedy Column-Adress-Strobe ma wystąpić, wtedy ani !CasInh ani !ExtSel nie jest aktywny.

RAS:L jest podobny do !CAS, tylko że ten sygnał generowany jest też w cyklu odświeżania i nie zależy od !CasInh.

!Ref jest sprzętową kopią odpowiedniego wyjścia ANTIC-u stosowany tylko wtedy, gdy chcemy uzyskać informację o chwilowym statusie szyny z

sygnałów !RAS/!CAS. Jest to na przykład konieczne, gdy chcemy odwołać się przez !CAS i !RAS do układów, które nie wymagają odświeżania. Jest to bardzo trafne, gdyż trudno uzyskać potrzebny sygnał z pozostałych wyjść CPU. Sygnały te są ważne, gdy chcemy mieć dostęp do systemu z zewnątrz.

!MPD służy do wyłączenia ROM z procedurami matematycznymi w obszarze D800h-DFFFh.

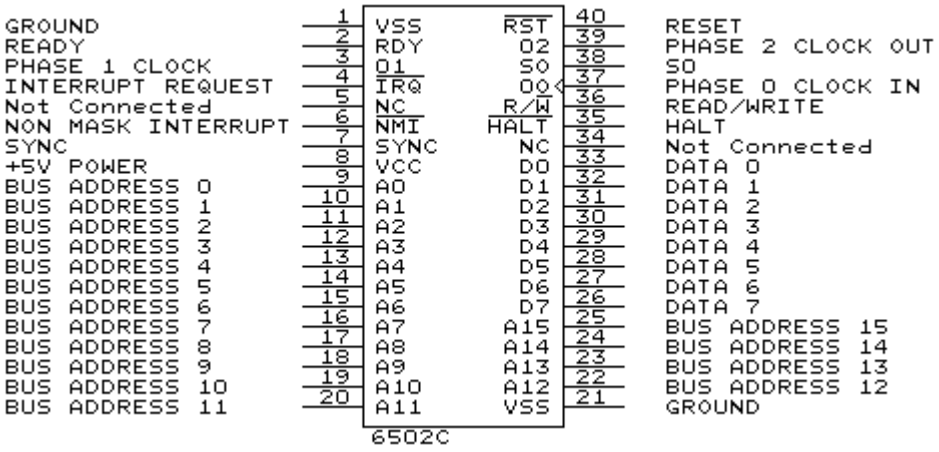
Ostatnim dającym się użyć sygnałem jest Audio In, który można zmieszać z sygnałem dźwiękowym tworzonym przez komputer. To wejście jest odseparowane od składowej stałej przez kondensator 4,7µF, można jednak wykorzystać je jako wyjście. Można przy tym liczyć na maksymalne napięcie wyjściowe 200 mV.

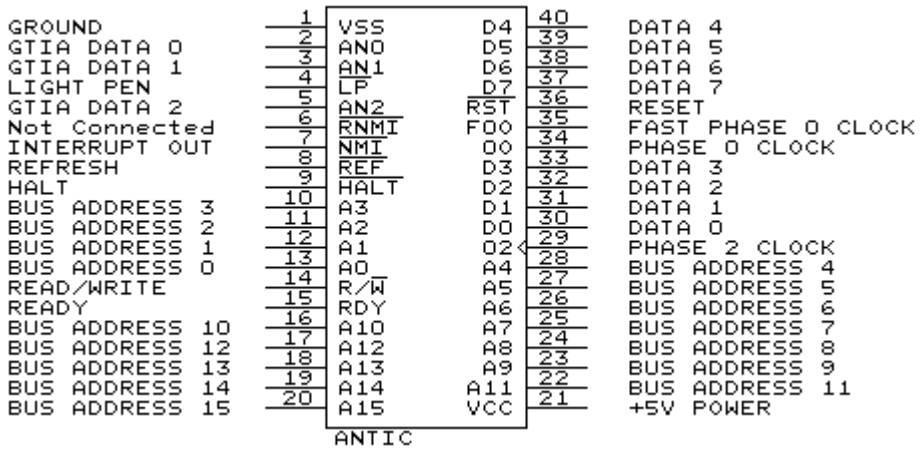
Całkowity przegląd tych złącz pozwala sądzić, że są otwarte przez nie wszystkie możliwości rozszerzeń zewnętrznych, przy czym uniemożliwiają one dokonywanie jakichkolwiek zmian wewnętrznej struktury, które nie były planowane przez Atari.

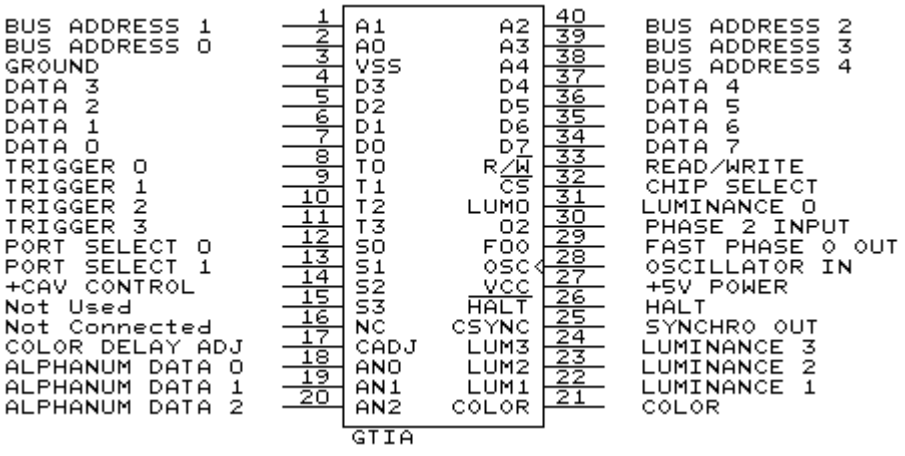
Ogólnie sygnały, które są wyprowadzone na zewnątrz, nie są buforowane, więc wyprowadzenia adresowe i danych można obciążyć jednym wejściem TTL LS, a pozostałe jak !RESET, !CAS i inne nie więcej jak pięcioma wejściami TTL LS. Przy !CAS i !RAS trzeba zauważyć, że pamięci dynamiczne mają bardzo dużą pojemność wejściową, która znacznie obciąża źródło sygnału.

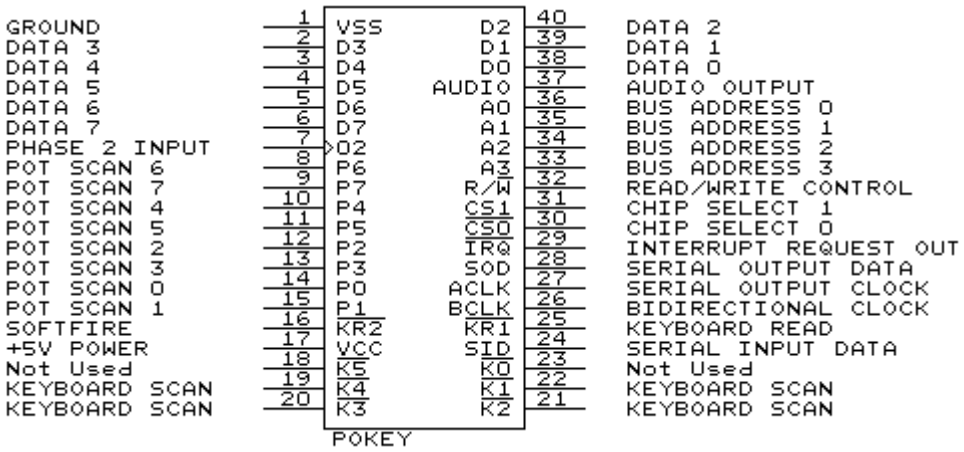
PODZIAŁ PAMIĘCI I PLANY PRZYŁĄCZEŃ

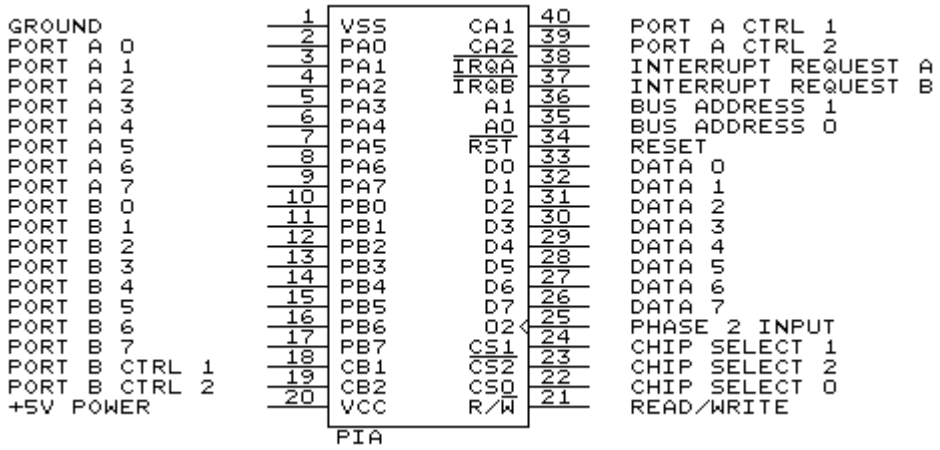
<table>
<tr><td>0</td><td>0000h</td><td rowspan="2">RAM
16KB</td><td></td><td></td></tr>
<tr><td>16384</td><td>4000h</td><td></td><td></td></tr>
<tr><td>20480
22527</td><td>5000h
57FFh</td><td rowspan="2">RAM
800/800XL
16KB</td><td></td><td>Odwzorowanie
ROM SELF TEST</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>32768</td><td>8000h</td><td>RAM
800/800XL
8KB</td><td></td><td>Obszar
Kartridża
Sel4</td></tr>
<tr><td>40960</td><td>C000h</td><td>RAM
800XL
8KB</td><td>BASIC
600XL/800XL
8KB</td><td>Obszar
Kartridża
Sel5</td></tr>
<tr><td>49152</td><td>C000h</td><td>RAM 800XL
4KB</td><td>ROM 4KB
600XL/800XL</td><td></td></tr>
<tr><td>53248</td><td>D000h</td><td rowspan="8">RAM
800XL
Nienadająca
się do użytku</td><td rowspan="8">SELF TEST
Odwzorowywany
w 5000h-7000h</td><td>GTIA 256B</td></tr>
<tr><td>53504</td><td>D100h</td><td>niewykorzystany</td></tr>
<tr><td>53760</td><td>D200h</td><td>POKEY 256B</td></tr>
<tr><td>54016</td><td>D300h</td><td>PIA 256B</td></tr>
<tr><td>54272</td><td>D400h</td><td>ANTIC 256B</td></tr>
<tr><td>54528</td><td>D500h</td><td>Kartridż CS</td></tr>
<tr><td>54784</td><td>D600h</td><td>niewykorzystany</td></tr>
<tr><td>55040</td><td>D700h</td><td>niewykorzystany</td></tr>
<tr><td>55296</td><td>D800h</td><td rowspan="2">RAM
800XL
10KB</td><td>Math Pack
2KB</td><td></td></tr>
<tr><td>61440
65535</td><td>F000h
FFFFh</td><td>ROM 8KB
System Oper.</td><td></td></tr>
</table>

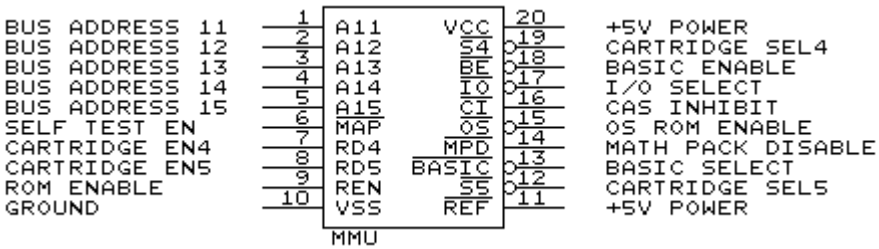

GROUND
READY
PHASE 1 CLOCK
INTERRUPT REQUEST
Not Connected
NON MASK INTERRUPT
SYNC
+5V POWER
BUS ADDRESS 0
BUS ADDRESS 1
BUS ADDRESS 2
BUS ADDRESS 3
BUS ADDRESS 4
BUS ADDRESS 5
BUS ADDRESS 6
BUS ADDRESS 7
BUS ADDRESS 8
BUS ADDRESS 9
BUS ADDRESS 10
BUS ADDRESS 11
1 VSS
2 RDY
3 Φ1
4 IRQ
5 NC
6 NMI
7 SYNC
8 VCC
9 A0
10 A1
11 A2
12 A3
13 A4
14 A5
15 A6
16 A7
17 A8
18 A9
19 A10
20 A11
RST 40
Φ2 39
SO 38
Φ0 37
R/W 36
HALT 35
NC 34
D0 33
D1 32
D2 31
D3 30
D4 29
D5 28
D6 27
D7 26
A15 25
A14 24
A13 23
A12 22
VSS 21
RESET
PHASE 2 CLOCK OUT
SO
PHASE 0 CLOCK IN
READ/WRITE
HALT
Not Connected
DATA 0
DATA 1
DATA 2
DATA 3
DATA 4
DATA 5
DATA 6
DATA 7
BUS ADDRESS 15
BUS ADDRESS 14
BUS ADDRESS 13
BUS ADDRESS 12
GROUND
6502C

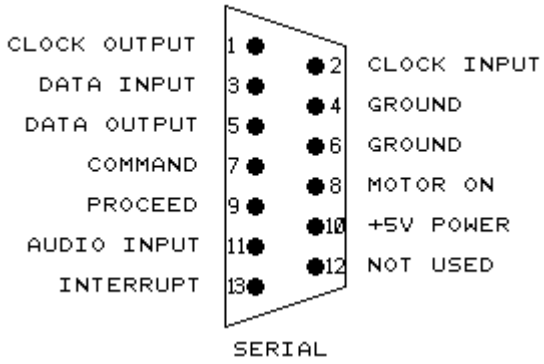

GROUND
GTIA DATA 0
GTIA DATA 1
LIGHT PEN
GTIA DATA 2
Not Connected
INTERRUPT OUT
REFRESH
HALT
BUS ADDRESS 3
BUS ADDRESS 2
BUS ADDRESS 1
BUS ADDRESS 0
READ/WRITE
READY
BUS ADDRESS 10
BUS ADDRESS 12
BUS ADDRESS 13
BUS ADDRESS 14
BUS ADDRESS 15
1 VSS
2 AN0
3 AN1
4 LP
5 AN2
6 RNMI
7 NMI
8 REF
9 HALT
10 A3
11 A2
12 A1
13 A0
14 R/W
15 RDY
16 A10
17 A12
18 A13
19 A14
20 A15
D4 40
D5 39
D6 38
D7 37
RST 36
F00 35
00 34
D3 33
D2 32
D1 31
D0 30
Φ2 29
A4 28
A5 27
A6 26
A7 25
A8 24
A9 23
A11 22
VCC 21
DATA 4
DATA 5
DATA 6
DATA 7
RESET
FAST PHASE 0 CLOCK
PHASE 0 CLOCK
DATA 3
DATA 2
DATA 1
DATA 0
PHASE 2 CLOCK
BUS ADDRESS 4
BUS ADDRESS 5
BUS ADDRESS 6
BUS ADDRESS 7
BUS ADDRESS 8
BUS ADDRESS 9
BUS ADDRESS 11
+5V POWER
ANTIC

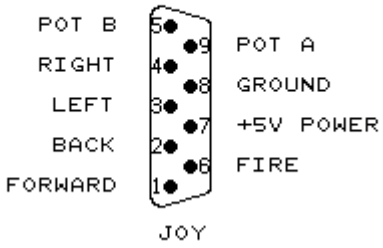

BUS ADDRESS 1
BUS ADDRESS 0
GROUND
DATA 3
DATA 2
DATA 1
DATA 0
TRIGGER 0
TRIGGER 1
TRIGGER 2
TRIGGER 3
PORT SELECT 0
PORT SELECT 1
+CAV CONTROL
Not Used
Not Connected
COLOR DELAY ADJ
ALPHANUM DATA 0
ALPHANUM DATA 1
ALPHANUM DATA 2
1 A1
2 A0
3 VSS
4 D3
5 D2
6 D1
7 D0
8 T0
9 T1
10 T2
11 T3
12 S0
13 S1
14 S2
15 S3
16 NC
17 CADJ
18 AN0
19 AN1
20 AN2
A2 40
A3 39
A4 38
D4 37
D5 36
D6 35
D7 34
R/W 33
CS 32
LUM0 31
Φ2 30
F00 29
OSC 28
VCC 27
HALT 26
CSYNC 25
LUM3 24
LUM2 23
LUM1 22
COLOR 21
BUS ADDRESS 2
BUS ADDRESS 3
BUS ADDRESS 4
DATA 4
DATA 5
DATA 6
DATA 7
READ/WRITE
CHIP SELECT
LUMINANCE 0
PHASE 2 INPUT
FAST PHASE 0 OUT
OSCILLATOR IN
+5V POWER
HALT
SYNCHRO OUT
LUMINANCE 3
LUMINANCE 2
LUMINANCE 1
COLOR
GTIA

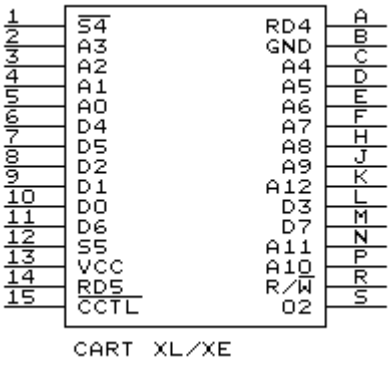

GROUND
DATA 3
DATA 4
DATA 5
DATA 6
DATA 7
PHASE 2 INPUT
POT SCAN 6
POT SCAN 7
POT SCAN 4
POT SCAN 5
POT SCAN 2
POT SCAN 3
POT SCAN 0
POT SCAN 1
SOFTFIRE
+5V POWER
Not Used
KEYBOARD SCAN
KEYBOARD SCAN
1 VSS
2 D3
3 D4
4 D5
5 D6
6 D7
7 O2
8 P6
9 P7
10 P4
11 P5
12 P2
13 P3
14 P0
15 P1
16 KR2
17 VCC
18 K5
19 K4
20 K3
40 D2
39 D1
38 D0
37 AUDIO
36 A0
35 A1
34 A2
33 A3
32 R/W
31 CS1
30 CS0
29 IRQ
28 SOD
27 ACLK
26 BCLK
25 KR1
24 SID
23 K0
22 K1
21 K2
POKEY
DATA 2
DATA 1
DATA 0
AUDIO OUTPUT
BUS ADDRESS 0
BUS ADDRESS 1
BUS ADDRESS 2
BUS ADDRESS 3
READ/WRITE CONTROL
CHIP SELECT 1
CHIP SELECT 0
INTERRUPT REQUEST OUT
SERIAL OUTPUT DATA
SERIAL OUTPUT CLOCK
BIDIRECTIONAL CLOCK
KEYBOARD READ
SERIAL INPUT DATA
Not Used
KEYBOARD SCAN
KEYBOARD SCAN

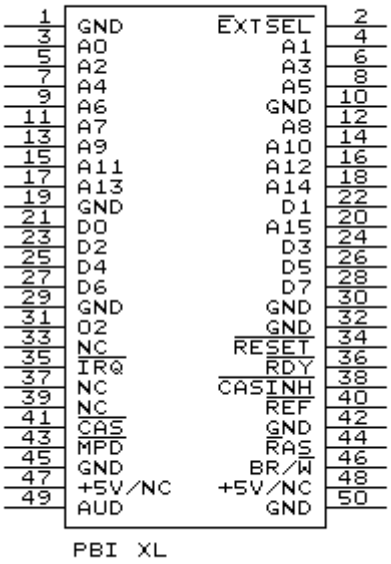

GROUND
PORT A 0
PORT A 1
PORT A 2
PORT A 3
PORT A 4
PORT A 5
PORT A 6
PORT A 7
PORT B 0
PORT B 1
PORT B 2
PORT B 3
PORT B 4
PORT B 5
PORT B 6
PORT B 7
PORT B CTRL 1
PORT B CTRL 2
+5V POWER
1 VSS
2 PA0
3 PA1
4 PA2
5 PA3
6 PA4
7 PA5
8 PA6
9 PA7
10 PB0
11 PB1
12 PB2
13 PB3
14 PB4
15 PB5
16 PB6
17 PB7
18 CB1
19 CB2
20 VCC
40 CA1
39 CA2
38 IRQA
37 IRQB
36 A1
35 A0
34 RST
33 D0
32 D1
31 D2
30 D3
29 D4
28 D5
27 D6
26 D7
25 O2
24 CS1
23 CS2
22 CS0
21 R/W
PIA
PORT A CTRL 1
PORT A CTRL 2
INTERRUPT REQUEST A
INTERRUPT REQUEST B
BUS ADDRESS 1
BUS ADDRESS 0
RESET
DATA 0
DATA 1
DATA 2
DATA 3
DATA 4
DATA 5
DATA 6
DATA 7
PHASE 2 INPUT
CHIP SELECT 1
CHIP SELECT 2
CHIP SELECT 0
READ/WRITE

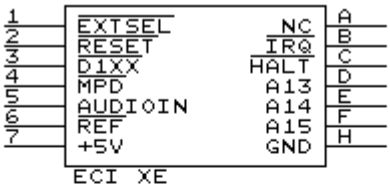

BUS ADDRESS 11
BUS ADDRESS 12
BUS ADDRESS 13
BUS ADDRESS 14
BUS ADDRESS 15
SELF TEST EN
CARTRIDGE EN4
CARTRIDGE EN5
ROM ENABLE
GROUND
1 A11
2 A12
3 A13
4 A14
5 A15
6 MAP
7 RD4
8 RD5
9 REN
10 VSS
20 VCC
19 S4
18 BE
17 IO
16 CI
15 OS
14 MPD
13 BASIC
12 S5
11 REF
MMU
+5V POWER
CARTRIDGE SEL4
BASIC ENABLE
I/O SELECT
CAS INHIBIT
OS ROM ENABLE
MATH PACK DISABLE
BASIC SELECT
CARTRIDGE SEL5
+5V POWER

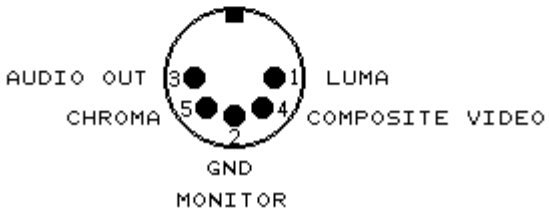

CLOCK OUTPUT
1
DATA INPUT
3
DATA OUTPUT
5
COMMAND
7
PROCEED
9
AUDIO INPUT
11
INTERRUPT
13
2
CLOCK INPUT
4
GROUND
6
GROUND
8
MOTOR ON
10
+5V POWER
12
NOT USED
SERIAL


POT B
5
RIGHT
4
LEFT
3
BACK
2
FORWARD
1
9
POT A
8
GROUND
7
+5V POWER
6
FIRE
JOY


1 S4
2 A3
3 A2
4 A1
5 A0
6 D4
7 D5
8 D2
9 D1
10 D0
11 D6
12 S5
13 VCC
14 RD5
15 CCTL
RD4 A
GND B
A4 C
A5 D
A6 E
A7 F
A8 H
A9 J
A12 K
D3 L
D7 M
A11 N
A10 P
R/W R
02 S
CART XL/XE


1 GND
3 A0
5 A2
7 A4
9 A6
11 A7
13 A9
15 A11
17 A13
19 GND
21 D0
23 D2
25 D4
27 D6
29 GND
31 O2
33 NC
35 IRQ
37 NC
39 NC
41 CAS
43 MPD
45 GND
47 +5V/NC
49 AUD
EXTSEL 2
A1 4
A3 6
A5 8
GND 10
A8 12
A10 14
A12 16
A14 18
D1 22
A15 20
D3 24
D5 26
D7 28
GND 30
GND 32
RESET 34
RDY 36
CASINH 38
REF 40
GND 42
RAS 44
BR/W 46
+5V/NC 48
GND 50
PBI XL


1 EXTSEL
2 RESET
3 D1XX
4 MPD
5 AUDIOIN
6 REF
7 +5V
NC A
IRQ B
HALT C
A13 D
A14 E
A15 F
GND H
ECI XE


AUDIO OUT
3
1
LUMA
CHROMA
5
4
COMPOSITE VIDEO
2
GND
MONITOR

ANTIC

ANTIC nie jest standardowym układem scalonym, lecz układem specjalizowanym. "ANTIC" jest skrótem nazwy "Alphanumeric Television Interface Controller" i jest to specjalnie zaprojektowany mikroprocesor, który jest zastosowany jako koprocesor graficzny. Może on "kraść" cykle pracy CPU, tzn., że zatrzymuje on CPU zawsze wtedy, gdy potrzebuje danych z pamięci operacyjnej. Tak więc ANTIC czyta dane poprzez bezpośredni dostęp do pamięci (DMA - Direct Memory Access). ANTIC może adresować całe 64 KB pamięci i nie musi, jak układy graficzne w innych systemach komputerowych, przekazywać danych przez rejestr w określonym miejscu pamięci.

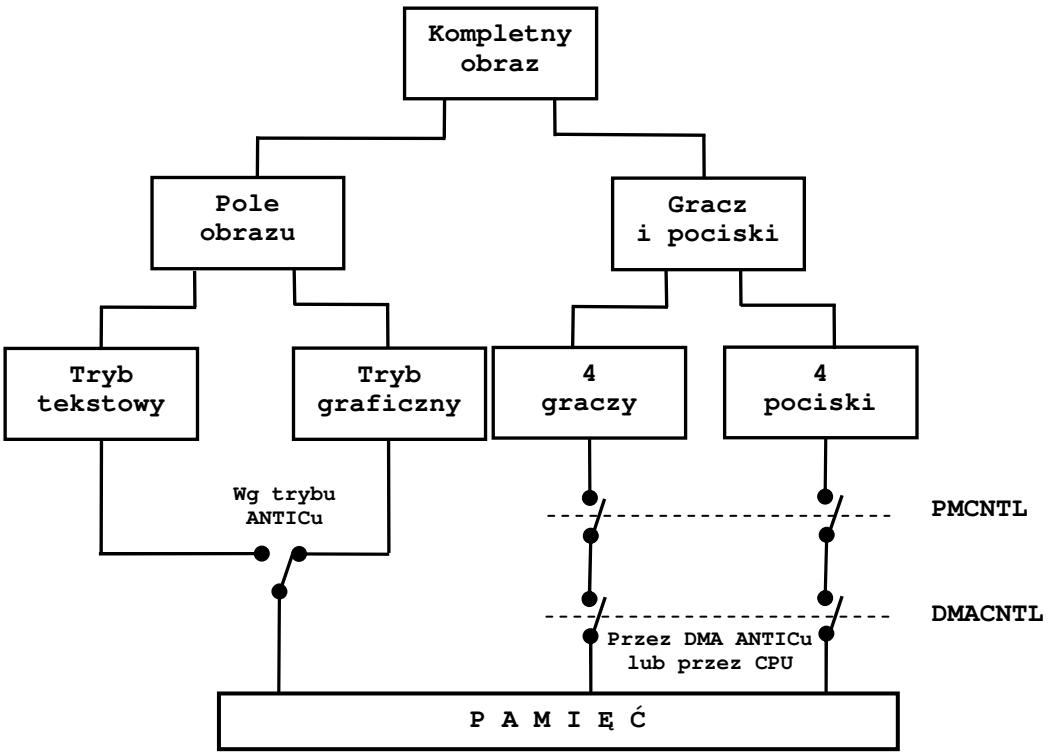
ANTIC posiada własny zestaw rozkazów, dzięki któremu poszczególne lub też kilka jednakowych punktów obrazu umieszcza na monitorze. Dzięki niemu jest możliwe dowolne kombinowanie na ekranie różnych trybów graficznych. Można więc uzyskać jednocześnie tryb graficzny i tekstowy. Jest to dotychczas niemożliwe w żadnym innym komputerze tej klasy cenowej.

ANTIC nie jest jednak wystarczający do otrzymania obrazu, gdyż nie tworzy on sam gotowego sygnału wizji z informacjami o kolorze, lecz przesyła informacje potrzebne do utworzenia obrazu, które otrzymał z pamięci poprzez rozkazy swojego programu (program ANTIC-u), do układu GTIA.

GTIA dodaje do informacji o obrazie, które otrzymał z ANTIC-a, informacje o kolorze. Poza tym GTIA służy do tzw. Player-Missile Graphic (PMG), tzn. do tworzenia na ekranie ruchomych obiektów. ANTIC może pomóc GTIA w tworzeniu PMG, w ten sposób, że pobiera z pamięci dane do utworzenia PMG i przenosi je do rejestru GTIA.

Dla wielu rejestrów ANTIC-u są przewidziane tzw., rejestry cieni. Zawartości rejestrów i rejestrów cieni są uzgadniane wzajemnie podczas cyklu wygaszania pionowego obrazu. Dalsze informacje na ten temat znajdują się w rozdziałach o GTIA i systemie operacyjnym.

Przedstawiony poniżej schemat pozwala dokładniej zrozumieć sposób tworzenia i przetwarzania danych obrazu.



ANTIC tworzy 50 półobrazów na sekundę. Półobrazy te nie są przesunięte względem siebie jak w normalnym obrazie telewizyjnym. W ten sposób otrzymuje się spokojnie stojący obraz. Nazywane jest to również "nieprzeplatanym rytmem" ("non interlaced scan"). Każdy z tych półobrazów składa się z 310 linii obrazu. Każda linia obrazu składa się z 228 tzw. cykli koloru (Color Clocks). Jeden taki cykl koloru ma szerokość około dwóch wysokości linii obrazu. W jednej sekundzie daje to około 3,5 miliona cykli koloru (50 x 310 x 228). CPU pracuje z częstotliwością 1,77 MHz. Częstotliwość ta została dobrana tak, aby na jeden cykl maszynowy wypadały dokładnie dwa cykle koloru.

W trybie graficznym z największą rozdzielczością można podać poszczególne małe pola zwane "pixelami", z rozdzielczością poziomą pół cyklu koloru i pionową jednej linii obrazu.

Ponieważ telewizor normalnie nie pokazuje wyraźnej krawędzi obrazu, to system operacyjny tworzy po synchronizacji pionowej przed początkiem obrazu 24 puste linie. Pozwala to oglądać cały obraz na ekranie. Jeżeli pomimo to nie ma krawędzi obrazu na ekranie, to znaczy, że tzw. "Overscan" monitora lub telewizora jest za duży.

Wielkość poszczególnych pól obrazu na ekranie określana jest przez liczbę i rodzaj rozkazów w programie ANTIC-u.

Szerokość poszczególnych pól obrazu może przyjmować trzy różne wartości. Wąski obraz ma 128 cykli koloru, normalny obraz ma 160 cykli, a szeroki 192 cykle. Stąd otrzymujemy maksymalną rozdzielczość poziomą 256, 320 i 384 punkty obrazu (pixele).

System operacyjny zasadniczo ustawia program ANTIC-u, który ma 24 puste linie na górnej krawędzi obrazu i 192 właściwe linie obrazu, Zasadniczo nie można już nigdy przedstawić więcej linii niż system operacyjny Gdy próbuje się uzyskać więcej niż 24 linie puste i 192 linie obrazu, to jest możliwe, że obraz zacznie się zwijać.

REJESTRY KONTROLI I PIÓRA ŚWIETLNEGO

Rejestr (wzgl. rejestr cieni)

DMACNTL 54272 D400h .

DMACNTL$ 559 022Fh ,

kontroluje bezpośredni dostęp ANTIC-u do pamięci. Poszczególne bity rejestrów mają przy tym następujące funkcje:

Bit 0-1: Określa wielkość obrazu.

Bit 2: Gdy ten bit jest "1", to jest włączony DMA dla pocisków.

Bit 3: Gdy ten bit jest "1", to jest włączony DMA dla graczy.

Bit 4: Gdy ten bit jest "1", to jest wybrana jednowierszowa rozdzielczość dla graczy i pocisków, gdy bit zmienia się na "0", to ustawiana jest rozdzielczość dwuwierszowa,

Bit 5: Gdy ten bit jest "1", to jest włączony DMA dla czytania programu ANTIC-u. Gdy jest "0", ta nie pojawia się żaden obraz na monitorze.

Bit 6-7: niewykorzystane

Bity 0 i 1 wybierają szerokość pola obrazu następująco:

bit 1	bit 0	Funkcja
0	0	Nie ma cyklu DMA dla obrazu, tzn. brak obrazu.
0	1	Wąskie pole gry. Obraz ma 128 cykli koloru.
1	0	Normalne pole gry. Obraz ma 160 cykli koloru. Jest to zasadnicze ustawienie systemu operacyjnego.
1	1	Szerokie pole. Obraz ma 192 cykle koloru.

Standardową zawartością (tzw. default value) tego rejestru jest 34 lub 22h (0010 0010).

Często konieczne jest przerwanie normalnej pracy CPU, gdy trzeba wykonać określone zadania, na przykład odpowiedzieć na określone pytania urządzeń zewnętrznych. Służą do tego zasadniczo dwa różne przerwania. Przerwanie maskowane (IRQ-Interrupt Request - przerwanie na żądanie) jest używane przez POKEY i w danym przypadku przez PIA. Przerwanie niemaskowane (NMI-Non Maskable Interrupt) jest wywoływane przez ANTIC.

W starych modelach 400/800 także klawisz RESET wywoływał poprzez ANTIC NMI, w nowych modelach metoda ta jest nieużyteczna, gdyż na przykład po włożeniu kartridża koniecznie musi nastąpić sprzętowy Reset, gdyż przy tym wyłączany jest system. W starych modelach komputer automatycznie wyłącza się przy zmianie kartridża.

W ANTIC-u daje to rejestr NMIEN (54286 D40Eh)dzięki któremu można już w ANTIC-u wstrzymać tzw. przerwania programu ANTIC-u i przerwania synchronizacji pionowej (w których system operacyjny np. aktualizuje rejestry cieni). Maskowanie NMI przez (tu nieprzewidziany) sygnał RNMI jest niemożliwe w NMIEN. Bity w NMIEN mają następujące funkcje:

Bity 0-5: **niewykorzystane.**

Bit 6: **Gdy ten bit jest "1", to jest włączone przerwanie synchronizacji pionowej.**

Bit 7: **Gdy ten bit jest "1", to są możliwe przerwania programu ANTICu.**

System operacyjny ustawia w NMIEN wartość 64 wzg1. 40h (0100 0000). Gdy wejście NMI w CPU przechodzi na "0", to przerywana jest normalna praca i wykonuje się procedura przerwania, której adres początkowy jest umieszczony w określonym miejscu pamięci (FFFAh i FFFBh). Procedura ta musi na początku stwierdzić, czy przerwanie jest wyłączone. Może ona to sprawdzić w rejestrze:

NMIST 54287 D40Fh

Każde źródło przerwania ma do dyspozycji jeden bit. Przez ustawienie odpowiednich bitów ANTIC wskazuje, że występuje określony warunek przerwania. Bity w NMIST są przyporządkowane następująco:

Bit 0-4: **niewykorzystane**

Bit 5: **wejście RNMI**

Bit 6: **przerwanie synchronizacji pionowej**

Bit 7: **przerwanie programu ANTIC-u**

Dla przestawienia bitów w tym rejestrze trzeba jedynie wpisać jakąś wartość w rejestrze

NMIRES 54287 D40Fh

Rejestr ten ma taki sam adres jak NMIST. Podczas czytania występuje pod tym adresem NMIST, natomiast podczas wpisywania NMIRES.

Dzięki rejestrowi

VCOUNT 54283 D40Bh

możliwe jest stwierdzenie, którą linię obrazu ANTIC właśnie przedstawia. Bity rejestru VCOUNT zawierają stan wewnętrznego licznika linii obrazu. W VCOUNT nie jest jednak zawarty najniższy bit licznika linii. VCOUNT zawiera więc numer aktualnej linii obrazu podzielony przez dwa. Przy jednym przebiegu obrazu VCOUNT zlicza od 0 do 155.

W niektórych wypadkach dla uzyskania szczególnych efektów konieczne jest pozwolenie na pracę procesora równolegle do linii. Ponieważ tworzenie linii

jest wtedy zbyt szybkie dla zwykłego zapytania, kiedy zaczyna się następna linia, to umożliwia to rejestr

WAITSYNC 54282 D40Ah

Gdy zostanie wpisana w tym rejestrze jakaś wartość, to po prostu CPU jest zatrzymywana do początku następnej linii. Wyjście READY CPU jest przy tym ustawiane na "0". Skoro tylko nadejdzie następny sygnał synchronizacji poziomej, a więc linia została zakończona, CPU znowu podejmuje pracę.

W komputerze Atari do złącza dżojstika można włączyć pióro świetlne. Gdy takie pióro świetlne zatrzyma się na ekranie, jego położenie jest zapisywane w dwóch rejestrach i rejestrach cieni:

LPENH 54284 D40Ch Tu jest umieszczana pozioma pozycja pióra świetlnego.

LPENH$ 564 0234h Rejestr zawiera numer cyklu koloru, w którym znajduje się pióro świetlne.

LPENV 54285 D40Dh Tu jest umieszczona pionowa pozycja pióra świetlnego.

LPENV$ 565 0235h Rejestr zawiera numer linii obrazu, w której znajduje się pióro świetlne, podzielony przez dwa (jak VCOUNT).

Powoduje to wszakże częste problemy z podłączeniem pióra świetlnego. Pióro świetlne jest właściwie niczym innym jak fototranzystorem. Zasadą pióra świetlnego jest podanie do ANTIC-u sygnału, kiedy strumień tworzący obraz trafi do pióra. W tym celu fototranzystor musi bardzo szybko reagować. Tego rodzaju szybkie elementy są bardzo drogie. Czasami otrzymuje się akceptowalne wyniki z tanimi fototranzystorami, przeważnie jednak występują problemy z określeniem pozycji poziomej. W wielu zastosowaniach wystarczające jest ustalenie pionowej pozycji pióra świetlnego, np. gdy chcemy wybierać punkty z listy przy pomocy pióra.

DMA DLA PLAYER-MISSILE GRAPHICS

Player-Missile Graphics (PMG) jest tworzona przez GTIA. Jednak GTIA wspiera przy tym tylko jedną linię obrazu. Musi on więc dla każdej linii obrazu wczytać do tzw. rejestru grafiki nowe dane. Może to być także zrobione przez assembler, wymaga jednak bardzo dużego nakładu pracy. Najprościej jest wykorzystać do Player-Missile Graphics DMA ANTIC-u.

Trzeba jedynie w pamięci zbudować pole pamięci z danymi graczy i pocisków. Dla każdej linii obrazu (lub dla każdej co drugiej linii, gdy jest wybrana rozdzielczość dwuwierszowa) pobiera ANTIC dane z pamięci i przesyła je do rejestru grafiki GTIA. Poziome położenie obiektów wybierane jest wtedy w GTIA, a pionowe jest ustalane przez miejsce obiektu w polu pamięci Player-Missile.

W ten sposób gracz może mieć dowolny pionowy wymiar, a więc może też wypełnić całą wysokość ekranu. Do poziomego przedstawienia wykorzystuje się 8 bitów dla graczy i 2 bity dla pocisków. Przez rejestr SIZE w GTIA wybiera się, ile cykli koloru wypada na jeden bit gracza lub pocisku, a więc jaki jest poziomy wymiar gracza lub pocisku.

DMA dla graczy i pocisków, jak również rozdzielczość pionowa, są włączane lub wybierane za pomocą rejestru DMACNTL lub DMACNTL$.

Bazowy adres pola pamięci Player-Missile jest umieszczony w rejestrze

PMBASE 54279 D407h

0	0h	PMBASE X 100h	0h	0
768	300h	3 2 1 0 pociski	180h	384
1024	400h	Gracz 0	200h	512
1280	500h	Gracz 1	280h	640
1536	600h	Gracz 2	300h	768
1792	700h	Gracz 3	380h	896
2047	7FFh		3FFh	1023
Rozdzielczość jednowierszowa		7 6 5 4 3 2 1 0 bity	Rozdzielczość dwuwierszowa	

Właściwy adres, z którego ANTIC pobiera dane dla PMG, zestawia się przy rozdzielczości jednowierszowej w następujący sposób:

Bit 0-7: Licznik linii obrazu dla graczy i pocisków.

Bit 8-10: Te bity wybierają, dla którego gracza/pocisku występuje DMA.

Bit 11: PMBASE, bit 3

Bit 12: PMBASE, bit 4

Bit 13: PMBASE, bit 5

Bit 14: PMBASE, bit 6

Bit 15: PMBASE, bit 7

PMBASE wybiera więc numer 256-bajtowego bloku, w którym rozpoczyna się pole pamięci Player-Missile. Ten blok musi jednak leżeć na początku 2 KB bloku pamięci. Można też powiedzieć, że PMBASE wybiera jeden z 32 możliwych bloków pamięci (wykorzystując 5 bitów). Bity 8-10 określają, jak już powiedziano, dla którego gracza/pocisku występuje DMA. Dzieje się to w następujący sposób:

Bit 10	bit 9	bit 8	Opis
0	0	0	Te trzy kombinacje nie mogą wystąpić, dlatego
0	0	1	też przy rozdzielczości jednowierszowej
0	1	0	pierwsze 3x256 bitów jest niewykorzystane. Zasadniczo jest 8 kombinacji bitów 8-10, ale ponieważ potrzeba tylko 4 pola dla graczy i 1 dla pocisków, więc 3 kombinacje są są niewykorzystane.
0	1	1	DMA jest na pole pamięci dla pocisków,
1	0	0	DMA jest na pole pamięci dla gracza 0.
1	0	1	DMA jest na pole pamięci dla gracza 1.
1	1	0	DMA jest na pole pamięci dla gracza 2.
1	1	1	DMA jest na pole pamięci dla gracza 3.

Przy dwuwierszowej rozdzielczości graczy i pocisków właściwy adres dla DMA (tzn. adres, pod którym znajdują się dane przepisywane do rejestru grafiki) zestawiany jest następująco:

Bit 4-6: Licznik linii obrazu dla graczy i pocisków,.

Bit 7-9: Te bity wybierają, dla którego gracza/pocisku występuje DMA, Przy rozdzielczości jednowierszowej dają to bity 8-10. Jednak trzeba zauważyć, że każdy gracz 1 pociski zajmują tylko po 128

bitów, Na początku pola pamięci Player-Missile pozostaje więc tylko 3x128 niewykorzystanych bitów,

Bit 10: PMBASE, bit 2.

Bit 11: PMBASE, bit 3.

Bit 12: PMBASE, bit 4.

Bit 13: PMBASE, bit 5.

Bit 14: PMBASE, bit 6.

Bit 15: PMBASE, bit 7.

PMBASE wybiera więc numer 256-bajtowego bloku, w którym rozpoczyna się pole pamięci Player-Missile. Ten blok musi jednak leżeć na początku 1KB bloku pamięci. Można też powiedzieć, że PMBASE wybiera jeden z 64 możliwych bloków pamięci.

ZESTAW ROZKAZÓW I PROGRAM ANTIC-u

Jak już stwierdzono, ANTIC jest mikroprocesorem. Jego rozkazy powodują utworzenie od 1 do 16 linii obrazu. Poza tym każdy rozkaz ma określoną formę, która powoduje, że ładowany jest tzw. licznik pamięci obrazu, aby ANTIC zaznaczył sobie początek pamięci obrazu. Jak "normalne" mikroprocesory ANTIC ma też rozkazy skoków, które powodują czytanie programu ANTIC-u z innej komórki pamięci.

Naturalnie trzeba powiadomić ANTIC, w której komórce pamięci powinien skończyć swój program. Adres ten jest mu podawany w dwóch rejestrach wzgl. rejestrach cieni (tzw. licznik programu ANTIC-u):

DLPTRL	54274	D402h	mniejszy bajt adresu
DLPTRL$	560	0230h	" " " (rejestr cień)
DLPTRH	54275	D403h	większy bajt adresu
DLPTRH$	561	0231h	" " " (rejestr cień)

Skrót DLPTR oznacza "Display-List-Pointer", tj. licznik programu ANTIC-u. ANTIC używa zawartości tego rejestru jako wartości swojego licznika programu. Licznik programu ANTIC-u liczy każdy wykonany rozkaz. Ponieważ najwyższe 6 bitów licznika programu jest tylko w rejestrze i nie można więcej zliczyć, to program ANTIC-u może jedynie przez skok przekroczyć granicę 1 KB.

Przez zmianę zawartości rejestrów DLPTRL i DLPTRH można zmienić cały obraz. Trzeba przedtem wprowadzić jedynie nowy program ANTIC-u i pamięć obrazu.

Powinno się zmieniać ten rejestr tylko podczas synchronizacji pionowej, inaczej obraz będzie krótko wyświetlany. Zawartość rejestru cieni jest przepisywana do rejestru zawsze podczas synchronizacji pionowej, toteż nie trzeba tu zwracać uwagi na to, że przepisywanie może być tylko podczas synchronizacji. Aby ANTIC mógł czytać program, musi być ustawiony 5 bit w rejestrze DMACNTL.

Wszystkie rozkazy ANTIC-u można zasadniczo podzielić na trzy grupy:

1) Rozkazy tworzące puste linie '

2) Rozkazy skoków

3) Rozkazy tworzące linie obrazu

Każdy rozkaz ANTIC-u jest ładowany przez DMA do rejestru rozkazów. Przy rozkazach skoków ANTIC czyta przez DMA dwa dalsze bajty. Bajty te są ładowane do licznika programu ANTIC-u. Tak jak w rozkazach CPU tu też najpierw ładowany jest młodszy bajt, a później starszy.

Przy rozkazach, które powodują ładowanie licznika pamięci obrazu, ANTIC pobiera także z pamięci dwa dalsze bajty i przesyła je do rejestru licznika pamięci obrazu (tu też młodszy bajt najpierw). Przy normalnych

rozkazach wyświetlania ANTIC pobiera następny bajt danych obrazu z adresu, który następuje bezpośrednio po ostatnim bajcie poprzedniej linii. Licznik pamięci obrazu nie jest przy tym na nowo ładowany. Dzięki temu dane każdej następnej linii są czytane bezpośrednio z komórki pamięci, która następuje po poprzedniej linii. Dane poszczególnych linii są uszeregowane bez przerw. Otrzymuje się przez to blok pamięci obrazu.

Licznik pamięci obrazu składa się z 4 bitów rejestru i 12 bitów licznika. Dlatego łączny obszar pamięci obrazu nie może być większy niż 4 KB. W przypadku przejścia do następnego 4 KB bloku licznik pamięci obrazu musi być ponownie załadowany przez odpowiedni rozkaz ANTIC-u. Trzeba przy tym szczególnie uważać, żeby 4 KB blok nie był przekraczany podczas tworzenia linii. Może to mianowicie prowadzić do tego, że nagle wewnątrz bieżącej linii zostaną wczytane dane z komórki pamięci znajdującej się na początku 4 KB bloku, określonego przez 4 najwyższe bity licznika pamięci obrazu.

Każdy rozkaz ma wersję, która powoduje, że przy wykonywaniu ostatniej, przez ten rozkaz tworzonej linii obrazu, przerywana jest normalna praca CPU i wykonywana jest zamiast tego procedura, której adres początkowy jest umieszczony w DLIVKT (mniejszy bajt w 512/200h większy w 513/201h). Warunkiem dla tego rodzaju przerwania programu ANTIC-u jest ustawienie bitu w NMIEN na "1". Przerwanie programu ANTIC-u nie jest przerwaniem programu wykonywanego przez ANTIC, lecz przerwaniem pracy CPU, które jest wywołane przez program ANTIC-u.

Dzięki przerwaniu programu ANTIC-u można na przykład podczas tworzenia obrazu zmienić kolor w określonej linii. Jest też możliwe stworzenie więcej niż czterech graczy i pocisków przez przestawienie PMBASE na inne pole pamięci Player-Missile i, jeśli to konieczne, zmianę koloru i pozycji na ekranie. Przerwania programu ANTIC-u dają różnorodne możliwości, gdyż jest możliwe w bardzo prosty sposób, wykonanie części programu podczas tworzenia określonej pozycji obrazu.

Rozkazy tworzące puste linie

ANTIC posiada rozkazy, które tworzą na ekranie puste linie. Gdy tego rodzaju rozkaz jest wykonywany, to nie są pobierane z pamięci żadne dane obrazu. Zamiast tego wyświetla się na ekranie tylko kolor tła. Rozkazy pustych linii mają zasadniczo długość jednego bajta. Jednym rozkazem można utworzyć od jednej do ośmiu pustych linii. Rozkaz tworzący puste linie jest zbudowany następująco:

Bit 0-3: Te bity muszą mieć wartość 0000 (bin.).

Bit 4-6: Wartość tych trzech bitów określa liczbę linii tworzoną rozkazem, zmniejszoną o jeden. Na przykład wartość binarna 010 powoduje utworzenie 3 pustych linii.

Bit 7: Gdy ten bit jest "1", to po utworzeniu ostatniej z linii określonych tym rozkazem jest wywoływane przerwanie programu ANTIC-u.

Rozkazy skoków

Rozkazy te ładują na nowo licznik programu ANTIC-u. Gdy tego rodzaju rozkaz wystąpi w bloku obrazu, to spowoduje wyświetlenie pustej linii. Dlatego rozkazy skoków powinno się umieszczać tylko na końcu obrazu. Dwa bajty następujące w programie ANTIC-u po rozkazie skoku zawierają nowy adres (najpierw młodszy bajt). Rozkaz skoku jest zbudowany następująco:

Bit 0-3: Te bity muszą mieć wartość 0001 (bin.).

Bit 4: Gdy ten bit jest "0", to występuje prosty skok, przy którym jest wyświetlana pusta linia. Gdy jest on "1", to również następuje skok do podanego adresu. ANTIC czeka z wykonaniem następnego rozkazu do końca obrazu.

Bit 5-6: Te bity powinny być "0".

Bit 7: Gdy ten bit jest "1", to jest wywoływane przerwanie programu ANTIC-u.

W poniższych tabelach są zebrane rozkazy ANTIC-u podane w kodzie heksadecymalnym:

Rozkaz	Przerwanie programu ANTIC-u	
	jest	brak
1 pusta linia	00	80
2 puste linie	10	90
3 puste linie	20	A0
4 puste linie	30	B0
5 pustych linii	40	C0
6 pustych linii	50	D0
7 pustych linii	60	E0
8 pustych linii	70	F0
Skok	01	81
Skok i oczekiwanie	41	C1

Przewijanie poziome	--	XX	--	XX	--	XX	--	XX
Przewijanie pionowe	--	--	XX	XX	--	--	XX	XX
Ładowanie licznika	--	--	--	--	XX	XX	XX	XX
BEZ PRZERWANIA PROGRAMU ANTIC-u								
Tryb 2	02	12	22	32	42	52	62	72
Tryb 3	03	13	23	33	43	53	63	73
Tryb 4	04	14	24	34	44	54	64	74
Tryb 5	05	15	25	35	45	55	65	75
Tryb 6	06	16	26	36	46	56	66	76
Tryb 7	07	17	27	37	47	57	67	77
Tryb 8	08	18	28	38	48	58	68	78
Tryb 9	09	19	29	39	49	59	69	79
Tryb A	0A	1A	2A	3A	4A	5A	6A	7A
Tryb B	0B	1B	2B	3B	4B	5B	6B	7B
Tryb C	0C	1C	2C	3C	4C	5C	6C	7C
Tryb D	0D	1D	2D	3D	4D	5D	6D	7D
Tryb E	0E	1E	2E	3E	4E	5E	6E	7E
Tryb F	0F	1F	2F	3F	4F	5F	6F	7F
Z PRZERWANIEM PROGRAMU ANTIC-u								
Tryb 2	82	92	A2	B2	C2	D2	E2	F2
Tryb 3	83	93	A3	B3	C3	D3	E3	F3
Tryb 4	84	94	A4	B4	C4	D4	E4	F4
Tryb 5	85	95	A5	B5	C5	D5	E5	F5
Tryb 6	86	96	A6	B6	C6	D6	E6	F6
Tryb 7	87	97	A7	B7	C7	D7	E7	F7
Tryb 8	88	98	A8	B8	C8	D8	E8	F8
Tryb 9	89	99	A9	B9	C9	D9	E9	F9
Tryb A	8A	9A	AA	BA	CA	DA	EA	FA
Tryb B	8B	9B	AB	BB	CB	DB	EB	FB
Tryb C	8C	9C	AC	BC	CC	DC	EC	FC
Tryb D	8D	9D	AD	BD	CD	DD	ED	FD
Tryb E	8E	9E	AE	BE	CE	DE	EE	FE
Tryb F	8F	9F	AF	BF	CF	DF	EF	FF

Przykład programu ANTIC-u

Program ANTIC-u i pamięć obrazu trybie BASIC-u "0" /GRAPHICS 0/:

```
Adres       dane
 (hex)      (hex)
 7C20       70          24 puste linie na górnej krawędzi ekranu
            70
            70
            42          ładowanie licznika pamięci obrazu liczbą 7C40h
            40
            7C
            02
            02
            02
            02
            .
            .
            .
            02
            41          skok do 7C20h i oczekiwanie na koniec obrazu
            20          (ponowne ładowanie licznika programu ANTIC-u)
            7C
7C40                    1 linia obrazu
7C68                    2 linia obrazu
7C90                    3 linia obrazu
  .                           .
  .                           .
7F60                    23 linia obrazu
7F88                    24 linia obrazu
```

Pamięć obrazu ma razem 960 (3C0h) bajtów i kończy się na 32687 (7FAFh).

TRYBY GRAFICZNE

ANTIC dysponuje ośmioma różnymi trybami graficznymi. W trybie graficznym można na ekranie opisać każdy pojedynczy pixel. Nie wszystkie tryby są dostępne w BASIC-u. Poszczególne tryby różnią się między sobą stopniem rozdzielczości i liczbą kolorów dostępnych dla każdego pixela, a więc też potrzebną pamięcią.

Pod pojęciem pixela rozumie się prostokątne pola na ekranie, które mają wysokość od 1 do 8 linii obrazu i szerokość pół do 4 cykli koloru. W programie ANTIC-u ze 192 liniami i 160 cyklami koloru w linii otrzymuje się rozdzielczość od 24 X 40 do 192 X 320 pixeli. Przy 192 cyklach koloru otrzymuje się maksymalną poziomą rozdzielczość do 384 pixeli.

Tryby ANTIC-u z niewielką rozdzielczością mają tę zaletę, że potrzebują małej przestrzeni pamięci, a także pozwalają się szybko całkowicie zmieniać. Poza tym potrzebują one niewiele cykli DMA, więc spowalniają pracę CPU mniej niż tryby ANTIC-u o wysokiej rozdzielczości, Adresy pamięci obrazu są umieszczone w programie ANTIC-u.

ANTIC pobiera dane z pamięci przez DMA, natomiast przesyła dane obrazu do GTIA a także do wewnętrznego rejestru przesuwnego. Dane z tego rejestru wykorzystywane są w przypadku, gdy więcej linii jest jednakowych (gdy np.

pixel zajmuje kilka linii obrazu). W ten sposób ANTIC czyta dane obrazu jeszcze raz i oszczędza przez to cykle DMA.

Poszczególne tryby:

Opisy odwołują się, jeśli nie podano inaczej, do normalnej szerokości linii (160 cykli koloru na linię obrazu).

<table>
<tr><th>Tryb ANTIC-u</th><th>OPIS</th></tr>
<tr><td>8</td><td>Ten tryb ANTIC-u odpowiada trybowi BASIC-u "3". Przedstawia on 40 pixeli w normalnej linii, przy czym dysponuje czterema różnymi kolorami. Każdy pixel ma wysokość 8 linii obrazu. Dla normalnej linii konieczne jest 10 bajtów pamięci obrazu. Każdy pixel zajmuje 4 cykle koloru, więc przy normalnym tworzeniu obrazu pixel jest kwadratowy. Każde dwa bity danych obrazu wybierają rejestr koloru według poniższej tabeli:
<table>
<tr><th>Dane</th><th>Rejestr koloru</th></tr>
<tr><td>00</td><td>COLBAK</td></tr>
<tr><td>01</td><td>COLPF0</td></tr>
<tr><td>10</td><td>COLPP1</td></tr>
<tr><td>11</td><td>COLPF2</td></tr>
</table></td></tr>
<tr><td>9</td><td>Ten tryb ANTIC-u odpowiada trybowi BASIC-u "4". Przedstawia on 80 pixeli w normalnej linii, przy czym każdy pixel ma szerokość 2 cykli koloru. Dysponuje on dwoma różnymi kolorami do wyboru. Dla linii konieczne jest również 10 bajtów pamięci obrazu. W tym trybie każdy pixel także jest kwadratowy. Gdy bit danych obrazu jest "0", to wyświetlany jest kolor z COLBAK, w przeciwnym razie z COLPF0.</td></tr>
<tr><td>A</td><td>Ten tryb ANTIC-u odpowiada trybowi BASIC-u "5". Tu też pixele są kwadratowe. Linia zawiera 80 pixeli, każdy o szerokości po 2 cykle koloru. Dla każdego pixela są do dyspozycji 4 różne kolory. Potrzeba 20 bajtów pamięci dla linii. Każdy pixel ma wysokość 4 linii obrazu. Każde dwa bity danych obrazu wybierają jeden z czterech rejestrów koloru według następującej tabeli:
<table>
<tr><th>Dane</th><th>Rejestr koloru</th></tr>
<tr><td>00</td><td>COLBAK</td></tr>
<tr><td>01</td><td>COLPF0</td></tr>
<tr><td>10</td><td>COLPP1</td></tr>
<tr><td>11</td><td>COLPF2</td></tr>
</table></td></tr>
<tr><td>B</td><td>Ten tryb ANTIC-u odpowiada trybowi BASIC-u "6". Przedstawia on w linii 160 kwadratowych pixeli, z których każdy ma szerokość jednego cyklu koloru i wysokość dwóch linii obrazu, Dla każdego pixela są do dyspozycji 2 kolory. Na każdą linię konieczne jest więc 20 bajtów pamięci obrazu. Do wybrania koloru pixela służy jeden bit danych obrazu. Gdy ten bit jest "0", to wybiera kolor z rejestru COLBAK, w przeciwnym razie z rejestru COLPF0.</td></tr>
<tr><td></td><td>Ten tryb ANTIC-u nie jest stosowany w systemie operacyjnym. Szerokość pixela jest dwa razy większa od wysokości. W linii przedstawione jest 160 pixeli, każdy po jednym cyklu koloru</td></tr>
</table>

<table>
<tr><td>C</td><td>szerokości i jednej linii obrazu wysokości. Na każdą linię konieczne jest 20 bajtów pamięci obrazu. Do wybrania koloru pixela służy jeden bit danych obrazu. Gdy ten bit jest "0", to wybiera kolor z rejestru COLBAK, w przeciwnym razie z rejestru COLPF0.</td></tr>
<tr><td>D</td><td>Ten tryb ANTIC–u odpowiada trybowi BASIC–u "7". Pixel jest tu kwadratowy. W linii jest 160 pixeli, dla których można wybrać 4 kolory. Każdy pixel ma szerokość jednego cyklu koloru i wysokość 2 linii obrazu. Na każdą linię konieczne jest więc 40 bajtów pamięci obrazu. Każde dwa bity danych obrazu wybiera jeden z czterech rejestrów koloru według poniższej tabeli:
Dane Rejestr koloru
00 COLBAK
01 COLPF0
10 COLPP1
11 COLPF2</td></tr>
<tr><td>E</td><td>Ten tryb ANTIC–u nie jest wykorzystywany przez system operacyjny ani przez BASIC. Pixel jest dwa razy szerszy niż wysoki. W linii jest przedstawione 160 pixeli, które dysponują czterema kolorami. Każdy pixel ma szerokość 1 cyklu koloru i wysokość 1 linii obrazu. Dla każdej linii konieczne jest 40 bajtów pamięci obrazu. W tym trybie należy zachować szczególną ostrożność przy tworzeniu programu ANTIC–u: gdy stosuje się, ten tryb ze 192 liniami obrazu, to pamięć obrazu jest większa niż 4 KB, mniej więcej w środku programu ANTIC–u trzeba więc ponownie załadować licznik pamięci obrazu odpowiednim rozkazem ANTIC–u.
Każde dwa bity danych obrazu wybiera jeden z czterech rejestrów koloru według następującej tabeli:
Dane Rejestr koloru
00 COLBAK
01 COLPF0
10 COLPP1
11 COLPF2</td></tr>
<tr><td>F</td><td>Ten tryb ANTIC–u odpowiada trybowi BASIC–u "9". W każdej 1inii przedstawione jest 320 kwadratowych pixeli o wysokości jednej linii obrazu i szerokości pół cyklu koloru. Ponieważ pixele mają szerokość tylko pół cyklu koloru, to mogą się różnić tylko jasnością, a nie mogą mieć różnych kolorów. Dla każdej linii konieczne jest tu również 40 bajtów pamięci obrazu. Także w tym trybie przy tworzeniu programu ANTIC–u należy zachować szczególną ostrożność: gdy stosuje się ten tryb ze 192 liniami obrazu, to pamięć obrazu jest większa niż 4 KB, mniej więcej w środku programu ANTIC–u trzeba więc ponownie załadować licznik pamięci obrazu odpowiednim rozkazem ANTIC–u. Kolorem pixela jest zawsze kolor umieszczony w rejestrze COLPF2. Każdemu pixelowi odpowiada jeden bit danych obrazu. Gdy ten bit jest "0", to jasność pixela jest określana przez rejestr COLPF2, gdy jest on "1", to pixel posiada jasność pobraną z rejestru COLPF1.</td></tr>
</table>

W poniższej tabeli są jeszcze raz zestawione tryby graficzne ANTIC–u:

Tryb ANTIC–u	8	9	A	B	C	D	E	F
Tryb BASIC–u	3	4	5	6	–	7	–	8

Pixeli w linii (obraz wąski)	32	64	64	128	128	128	128	256
Pixeli w linii (obraz szeroki)	48	96	96	192	192	192	192	384
Pixeli w linii (obraz normalny)	40	80	80	160	160	160	160	320
Bajtów na linię (normalna linia)	10	10	20	20	20	40	40	40
Linii na pixel	8	4	4	2	1	2	1	1
Cykli kolorów na pixel	4	2	2	1	1	1	1	½
Liczba kolorów	4	2	4	2	2	4	4	1½
Bitów na pixel	2	1	2	1	1	2	1	1
STOSOWANE REJESTRY KOLORU								
COLPF0	X	X	X	X	X	X	X	
COLPF1	X		X			X	X	X
COLPF2	X		X			X	X	X
COLPF3								
COLBAK	X	X	X	X	X	X	X	

Tryby tekstowe

ANTIC dysponuje nie tylko trybami tworzenia grafiki punktowej, lecz posiada również 6 trybów tworzenia znaków pisarskich. Pojęcia znaków pisarskich nie należy brać zbyt dosłownie, można też tworzyć inne obiekty na ekranie przy użyciu trybów tekstowych, ale trzeba wtedy zdefiniować poszczególne znaki. Kilka trybów tekstowych nie nadaje się bezpośrednio do tworzenia liter, ponieważ mają one szerokość tylko 4 pixeli. Dlatego nadają się one do innych celów (np. w grach), ponieważ dysponują różnorodnymi możliwościami barw, W trybach tekstowych można utworzyć obrazy o wysokiej rozdzielczości w małej przestrzeni pamięci obrazu.

Najistotniejsza różnica pomiędzy trybami tekstowymi i graficznymi polega na źródle danych, które tworzą obraz. W trybach graficznych dane są bezpośrednio z pamięci obrazu wyświetlane na monitorze. W trybach tekstowych dane są rozumiane jako numery znaków (nazwy znaków) z zestawu. Dane nie są więc przedstawiane na monitorze bezpośrednio, lecz przez zestaw znaków umieszczony w pamięci operacyjnej (tzw. generator znaków). ANTIC czyta przy tym dane z pamięci obrazu i przesyła je do rejestru przesuwnego. Zależnie od tego, która linia obrazu jest tworzona, ANTIC czyta przez DMA dane obrazu z generatora znaków z pomocą danych z pamięci obrazu, które są umieszczone w rejestrze przesuwającym.

W generatorze znaków dla każdego znaku jest przewidziane 8 bajtów. Każdy bajt znaku określa wygląd znaku w jednej linii pixeli (a więc w jednej lub dwóch liniach obrazu zależnie od trybu). Generator znaków jest listą wszystkich danych znaków. Generator znaków nie zawiera żadnych przerw, znaki są umieszczone bezpośrednio jeden za drugim.

Po wyświetleniu jednej linii pixeli licznik linii w ANTIC-u jest zwiększany o jeden, co powoduje, że z generatora znaków są czytane dane dla następnej linii obrazu.

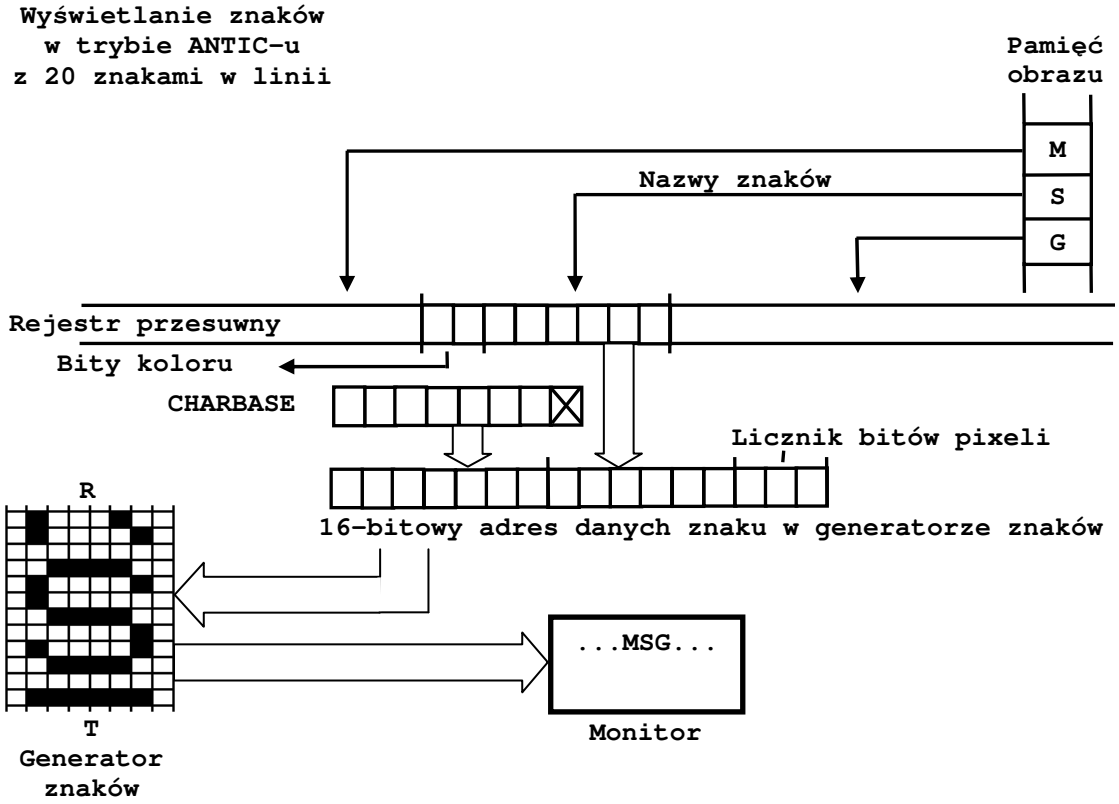
Naturalnie trzeba powiadomić ANTIC, gdzie znajduje się generator znaków. Numer 256-bajtowego bloku, w którym zaczyna się generator znaków, zapisuje się w rejestrze wzgl. w rejestrze cieni

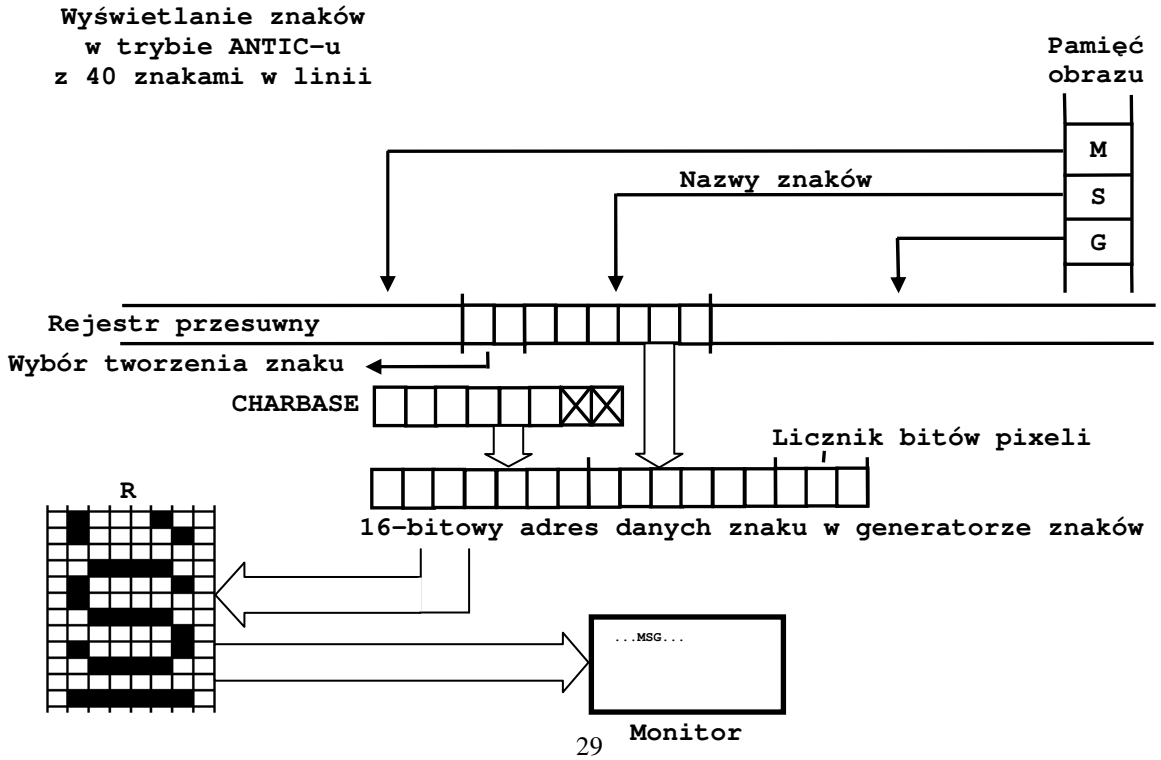
CHARBASE 54281 D409h

CHARBASE$ 756 02F4h

W trybie z 40 znakami w linii bity 0-6 danych pamięci obrazu są brane jako nazwa tworzonego znaku. Generator znaków obejmuję więc 128 znaków, Ponieważ każdy znak zajmuje 8 bajtów, to potrzeba 1024 bajty na generator znaków, W tym trybie są wykorzystywane bity 2-7 CHARBASE, dlatego generator znaków musi zaczynać się na granicy 1 KB w systemie.

W trybie z 20 znakami w linii brane są jako nazwa tworzonego znaku bity 0-5 danych pamięci obrazu. Generator znaków obejmuje więc 64 znaki. Ponieważ każdy znak zajmuje 8 bajtów, to potrzeba 512 bajtów na generator znaków. W tym trybie są wykorzystywane bity 1-7 CHARBASE. Generator znaków musi się więc zaczynać na granicy 512 bajtów w systemie.





Dolne 6 (w trybie z 64 znakami w zestawie) lub 7 (w trybie ze 128 znakami) bitów bajta pamięci obrazu określa więc, który znak z zestawu zostanie wyświetlony. Najwyższy lub dwa najwyższe bity określają w kilku trybach kolor. W dwóch trybach (tryby ANTIC-u "2" i "3") są przez bit 7 bajta pamięci obrazu sterowane dodatkowe funkcje. Dla tych dodatkowych funkcji i sterowania trybów tekstowych w ogóle wykorzystuje się rejestr lub rejestr-cień:

CHARCNTL 54273 D401h

CHARCNTL$ 755 02F3h

Bity tych rejestrów mają następujące funkcje:

Bit 0 **Wytłumianie znaków:**
Ten bit ma funkcję tylko w trybach "2" i "3". Gdy jest "1", to powoduje, że wszystkie znaki, których bit 7 jest "1", nie są wyświetlane na ekranie. Gdy bit 7 jest "1" a bit 0 CHARCNTL jest okresowo przełączany, to znak okresowo błyska.

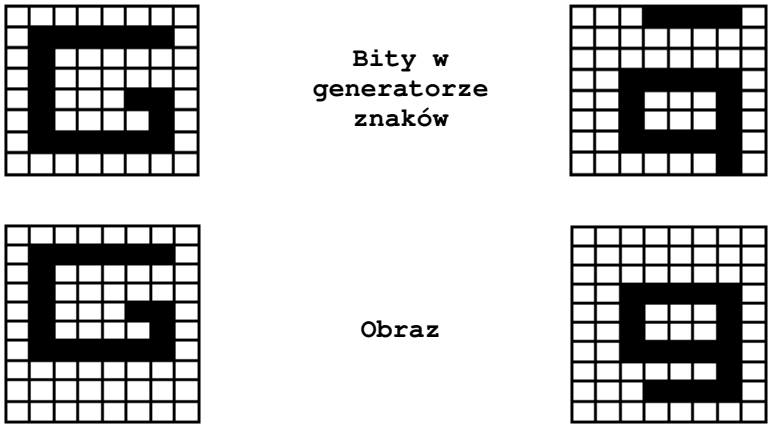
Bit 1 **Odwracanie znaków:**
Ten bit ma funkcję tylko w trybach "2" i "3". Gdy jest "1", to powoduje, że wszystkie znaki, których bit 7 jest "1", mają zamienione kolory znaku i tła.

Bit 2 **Ten bit jest sprawdzany na początku każdej linii znaków. Gdy jest on "1", to linia znaków jest odwrócona „do góry nogami”.**

Bit 3-7 **Niewykorzystane.**

Poszczególne tryby

Tryb ANTIC-u	OPIS
2	Ten tryb ANTIC-u odpowiada trybowi BASIC-u "0", który jest uaktywniany po włączeniu komputera. Ma on 40 znaków w normalnej linii. Każdy z tych znaków składa się z 8 linii pixeli, z których każdy ma szerokość pół cyklu koloru. Dlatego poszczególne pixele nie mogą mieć różnych kolorów, a tylko stopnie jasności. Znak ma wysokość 8 pixeli, przy czym każdy pixel zajmuje jedną linię obrazu. Każdy bit danych obrazu określa jasność jednego pixela, natomiast kolor jest zawsze określony przez COLPF2. Gdy bit jest "0" to jasność jest pobierana także z COLPF2, w przeciwnym przypadku z COLPF1. Zestaw znaków dla tego trybu składa się ze 128 znaków, więc generator znaków zajmuje 1024 bajty. Bit 7 nazwy znaku(danej w pamięci obrazu)steruje wyświetlaniem znaku przez CHARCNTL($).
3	Ten tryb ANTIC-u nie ma odpowiednika w systemie operacyjnym. Jak tryb "2" jest on przewidziany szczególnie do tworzenia znaków pisarskich. Istotną różnicą w porównaniu z trybem "2" jest, że można tym trybie tworzyć litery z przedłużeniem w dół. Przedłużenie jest na przykład konieczne dla litery "g": dolna część "g" musi leżeć poniżej innych liter. W normalnej linii znajduje się 40znaków po 8 pixeli poziomo, przy czym każdy pixel ma szerokość pół cyklu koloru. Także w tym trybie pixele mogą

	się różnić tylko jasnością. Każdy bit generatora znaków określa przy tym jasność pixela, "0" wybiera jasność z COLPF2, "1" – z COLPF1. Kolor jest określany zawsze przez COLPF2. Linia w tym trybie ma wysokość 10 pixeli, przy czym jeden pixel zajmuje jedną linię. Zestaw znaków zawiera 128 znaków, bit 7 nazwy znaku (bajta w pamięci obrazu) steruje wyglądem znaku poprzez rejestr CHARCNTL/CHARCNTL$. Poniższy szkic przedstawia tworzenie znaków z generatora: 
4	Ten tryb także nie jest wykorzystywany w BASIC–u. W tym trybie jest wyświetlane 40 znaków w normalnej linii. Poszczególny znak ma szerokość tylko 4 pixeli przy czym każdy pixel zajmuje jeden cykl koloru. Dzięki temu ten tryb nadaje się szczególnie do gier, które tworzone są w wielu kolorach, ale potrzebują tylko niewielką przestrzeń pamięci. Na przykład bardzo dobrze można w tym trybie tworzyć mapy. Znaki mają wysokość 8 pixeli, przy czym każdy pixel ma wysokość jednej linii obrazu. Znak ma więc wysokość 8 linii obrazu, Hity 0–6 nazwy znaku wybierają, który znak z zestawu będzie tworzony. Zestaw znaków zawiera więc w tym trybie 128 znaków. Bit 7 nazwy znaku (danych pamięci obrazu) służy do wyboru koloru. Kolor pixela jest poza tym wybierany przez bity generatora znaków. Każde dwa leżące obok siebie Bity w generatorze znaków wybierają rejestr koloru. Sposób wyboru koloru pixela pokazuje poniższa tabela: Bit 7 nazwy Znaku / Bity generatora znaków / Rejestr koloru 0 / 00 / COLBAK 0 / 01 / COLPF0 0 / 10 / COLPF1 0 / 11 / COLPF2 1 / 11 / COLPF3
5	Tryby ANTIC–u "4" i "5" są prawie jednakowe. Różnią się one jedynie wysokością linii. Podczas gdy w trybie "4" linia ma wysokość 8 linii obrazu, to w trybie "5" ma wysokość 16 linii obrazu. Lecz w trybie "5" znak ma także wysokość 8 pixeli, ale każdy pixel ma wysokość 2 linii obrazu. Również tryb "5" nie ma odpowiednika w BASIC–u. Wybór koloru jest identyczny jak w trybie "4".

6	Ten tryb ANTIC–u odpowiada trybowi BASIC–u "1". W tym trybie w normalnej linii jest wyświetlane 20 znaków. Każdy znak ma szerokość 8 pixeli, a każdy pixel zajmuje jeden cykl koloru. Linia ma wysokość 8 pixeli, a każdy pixel ma wysokość 2 linii obrazu. Dla znaków można wykorzystać 5 różnych kolorów. Bitami 0–5 danych pamięci obrazu (nazwy znaku) wybiera się znak. W tym trybie zestaw ma tylko 64 znaki. Bity 6 i 7 wybierają rejestr koloru: Bit 7 Bit 6 Rejestr koloru 0 0 COLPF0 0 1 COLPF1 1 0 COLPF2 1 1 COLPF3 Dla każdego pixela jest do wykorzystania jeden bit w generatorze znaków. Gdy ten bit jest "0", pixel ma kolor z rejestru COLBAK. Kolor z rejestru wybranego przez bity 6 1 7 jest używany w znaku tam, gdzie odpowiedni bit generatora znaków jest "1". Ten tryb nadaje się szczególnie do tworzenia dużych liter. Sposób wyboru koloru umożliwia otrzymywanie różnokolorowych znaków. Jednakże, gdy konieczne jest więcej niż dwa kolory w znaku, to trzeba zastosować tryb "4" lub "5". Dla normalnego pisma można zastosować bezpośrednio normalny zestaw znaków systemu operacyjnego.
7	Ten tryb ANTIC–u odpowiada trybowi BASIC–u "2". Tryby ANTIC–u "6" i "7" są prawie identyczne. Różnią się one jedynie wysokością linii. Podczas gdy w trybie "6" linia ma wysokość 8 linii obrazu, a każdy pixel 1 linii obrazu, to w trybie "7" linia ma wysokość 16 linii obrazu, a każdy pixel 2 linii obrazu. Wybór kolorów znaków przebiega tak samo jak w trybie "6".

W poniższej tabeli jeszcze raz przedstawione zostały tryby tekstowe ANTIC–u:

Tryb ANTIC–u	2	3	4	5	6	7
Tryb BASIC–u	0	–	–	–	1	2
Liczba znaków w linii	40	40	40	40	20	20
Liczba pixeli w znaku (poziomo)	8	8	4	4	8	8
Liczba cykli koloru na pixel (poziomo)	½	½	1	1	1	1
Liczba pikseli na znak (pionowo)	8	8	8	8	8	8
Liczba linii obrazu w linii znaków	8	10	8	16	8	16
Liczba linii obrazu na pixel	1	1	1	2	1	2
Liczba kolorów	1½	1½	5	5	5	5
Liczba znaków w zestawie	128	128	128	128	64	64
Liczba bajtów na zestaw	1K	1K	1K	1K	512	512
Liczba bajtów na linię	40	40	40	40	20	20
Liczba bitów generatora znaków na pixel	1	1	2	2	1	1

PRZEWIJANIE OBRAZU

Często występuje problem jak znaleźć miejsce na ekranie, gdy trzeba stworzyć więcej niż pojemność monitora. Jedną możliwością jest zmiana całego obrazu przy pomocy określonej komendy z klawiatury, inną automatyczne przełączenie obrazu podczas wykonywania programu.

Mikrokomputery przeważnie dają możliwość wyboru między dwoma pamięciami obrazu, a więc dwoma obrazami. Chcąc przełączać kilka obrazów lub, gdy jest przewidziana tylko jedna pamięć obrazu, w ogóle uzyskać drugi obraz, trzeba całą zawartość pamięci obrazu przesunąć do CPU. Ponieważ jest to dla CPU bardzo pracochłonne, więc dzieje się to zbyt wolno i zakłócenia obrazu są prawie nie do uniknięcia. Atari daje wobec tego komfortowe warunki zmiany obrazu. Wystarczy zmienić jedynie adres programu ANTIC-u. Gdy program ANTIC-u jest jednakowy dla obu obrazów, wystarczy zmienić w programie rozkaz, który zapisuje w liczniku pamięci obrazu adres tej pamięci.

Pomimo prostoty zmiana obrazu przy opracowywaniu dużych tabel lub pisaniu tekstów z przeszło 40 znakami w linii jest jednak kłopotliwa.

Znacznie lepszą możliwością pokazania więcej na ekranie niż się na nim mieści, jest przesuwanie obrazu małymi krokami poziomo lub pionowo. Ekran służy wtedy jako okno, przez które ogląda się większe pole. W ten sposób powstaje wrażenie pracy na większym obrazie.

To także łatwiej uzyskać w Atari niż w innych mikrokomputerach. W wielu innych urządzeniach trzeba przesuwać całą zawartość pamięci obrazu. Nakład pracy CPU jest przy tym ogromny. Następstwem jest powoli zmieniający się obraz. Atari umożliwia dla każdej linii ponowne ładowanie licznika pamięci obrazu przez program ANTIC-u. Można też przygotować wcześniej więcej niż mieści się na ekranie. Aby umożliwić przesuwanie obrazu, trzeba tylko nieznacznie zmienić program ANTIC-u. Zmiana programu ANTIC-u wymaga znacznie mniejszego nakładu pracy niż przesuwanie całej zawartości pamięci obrazu.

PRZYKŁAD:

Ekran jest podzielony na dwie części. Górna część ma 16 linii po 40 normalnych znaków i porusza się po polu 32 x 256 znaków. Dolna część ekranu zawiera 8 stałych linii tekstu. Program, który jest konieczny do otrzymania takiego obrazu, umieścimy w pamięci operacyjnej od adresu 5000h.

Następujący program tworzy wyżej opisany obraz (wszystkie dane i adresy podane są w kodzie szesnastkowym):

```
ADRES   DANE            KOMENTARZ
5000     70
         70             24 puste linie
         70
5003     42             1 linia obrazu
         XPOS           ładowanie licznika pamięci obrazu
         YPOS+60+0
         42             2 linia obrazu
         XPOS           ładowanie licznika pamięci obrazu
         YPOS+60+1
         42             3 linia obrazu
         XPOS           ładowanie licznika pamięci obrazu
         YPOS+60+1
         42             4 linia obrazu
         XPOS           ładowanie licznika pamięci obrazu
         YPOS+60+1
           .
           .
           .
           .
         42             15 linia obrazu
         XPOS           ładowanie licznika pamięci obrazu
         YPOS+60+E
         42             16 linia obrazu
         XPOS           ładowanie licznika pamięci obrazu
```

```
          YPOS+60+F
5030      42             1 linia stałej pamięci obrazu
          00             ładowanie licznika pamięci obrazu
          55
          02             2 linia
          02             3 linia
           .
           .
           .
          02             7 linia
          02             8 linia
503A      41             skok do adresu 5000 i oczekiwanie na synchronizację
          00             pionową
          50
Tu kończy się program ANTIC-u

5500                     Początek pamięci stałej części obrazu (1 linia)
5528                     2 linia stałej części obrazu
5550                     3 linia stałej części obrazu
5578                     4 linia stałej części obrazu
55A0                     5 linia stałej części obrazu
55C8                     6 linia stałej części obrazu
55F0                     7 linia stałej części obrazu
5618                     8 linia stałej części obrazu
563F                     koniec pamięci stałej części obrazu
6000                     Początek pola ruchomej części obrazu - 1 linia pola
6100                     2 linia pola
6200                     3 linia pola
6300                     4 linia pola
6400                     5 linia pola
  .
  .
  .
7D00                     31 linia pola
7E00                     32 linia pola
7FFF                     Koniec pola ruchomej części obrazu
```

Ten przykład świadomie jest zbudowany bardzo prosto. Przy każdej zmianie pozycji obrazu na polu pamięci trzeba zmienić program ANTIC-u. Może to być zrobione według próbki podanej w przykładzie. Za XPOS trzeba podstawić poziomą pozycję na polu pamięci, a za YPOS pionową. Trzeba przy tym uważać, żeby XPOS nie było większe niż D8h, a YPOS niż 10h. W przeciwnym razie wystąpi błąd, ponieważ koniec linii obrazu przekroczy koniec linii pola pamięci. Przykład także niezbyt dobrze wykorzystuje pamięć, gdyż poszczególne linie zaczynają się od adresów o okrągłych wartościach.

Naturalnie można ten przykład przenieść także na tryby ANTIC-u o wysokiej rozdzielczości. Jednakże dla pola pamięci obrazu potrzebna jest wtedy bardzo duża przestrzeń pamięci operacyjnej. Przesuwanie części obrazu opłaca się najczęściej w trybach ANTIC-u, które potrzebują mniejszej przestrzeni pamięci.

Opisana metoda ma jednak wadę. Nie można otrzymać obrazu przesuwanego po jednym pixelu, lecz obraz skacze zawsze poziomo o jeden znak i/lub pionowo o jedną linię.

ANTIC daje jednakże bardzo komfortowe możliwości przesuwania obrazu lub jego części. Możliwe jest przesuwanie obrazu poziomo o pojedyncze cykle koloru i/lub pionowo o poszczególne linie obrazu. Gracze i pociski pozostają przy tym jednak na swoich starych miejscach. Przesuwanie obrazu nie ma na nie wpływu.

W ANTIC–u są do dyspozycji dwa rejestry, które sterują przesuwaniem obrazu lub jego części w kierunku poziomym i pionowym. Do przesuwu poziomego służy Rejestr:

HSCROL 54276 D404h

Sterowanie przesuwem pionowym jest prowadzone poprzez rejestr

VSCROL 54277 D405h

Przez te rejestry nie można jednak bezwarunkowo wpływać na wszystkie linie obrazu. W każdym rozkazie ANTIC–u, który służy do tworzenia linii (oprócz rozkazów pustych linii), są do dyspozycji dwa bity, które włączają poziome wzgl., pionowe przesuwanie linii tworzonej przez ten rozkaz.

W ten sposób jest możliwe przesuwanie całego obrazu lub jego poszczególnych części. Przesuw poziomy jest włączany przez bit 4 rozkazu ANTIC–u, zaś przesuw pionowy przez bit 5. Część obrazu przesuwana poziomo nie może być jednak identyczna z częścią przesuwaną pionowo.

Przesuw pionowy

W trybie graficznym, w którym każdy rozkaz ANTIC–u tworzy tylko jedną linię obrazu, można przesuwać obraz o poszczególne linie obrazu w sposób użyty w przykładzie. W innych trybach, np. tekstowych, każdy rozkaz ANTIC–u tworzy kilka linii obrazu. Pojawia się tu taki problem, że nie można metodą pokazaną w przykładzie przesuwać obrazu o poszczególne linie.

ANTIC daje poprzez rejestr VSCROL możliwość przesuwania obrazu o poszczególne linie także w trybach ANTIC–u, które tworzą kilka linii. Przesuwa to wszystkie linie obrazu, przy których bit 5 w rozkazie ANTIC–u jest "1", w górę o liczbę linii ustawioną w VSCROL. Ostatnią w ten sposób przesuwaną linią jest pierwsza linia, przy której bit 5 w rozkazie ANTIC–u jest "0". Dla zrozumienia, jak ANTIC wykonuje przesuw, trzeba przypomnieć sobie normalny sposób tworzenia linii obrazu.

ANTIC posiada wewnątrz licznik, który określa, którą linię obrazu wyświetla właśnie linia programu ANTIC–u. Ten licznik jest w literaturze najczęściej nazywany DCTR (Deltacontrol – kontroler różnic). Licznik ten liczy linie obrazu od 0 do najwyższej liczby, jaka jest określona przez tryb ANTIC–u. Ta najwyższa liczba jest zawsze o jeden mniejsza od liczby linii obrazu, które są tworzone w tym trybie ANTIC–u. Poniższa tabela podaje maksymalny stan licznika dla poszczególnych trybów ANTIC–u:

Tryb ANTIC–u	Maksymalny stan licznika
2	7
3	9
4	7
5	15
6	7
7	15
8	7
9	3
A	3
B	1
C	0
D	1
E	0
F	0

W trybach ANTIC–u, w których tylko jedna linia obrazu jest tworzona, DCTR nie liczy. Dlatego w tych trybach maksymalny stan licznika jest "0". W tych trybach jest możliwe przesuwanie obrazu bez wykorzystania VSCROL.

Zwykle przez program ANTIC-u jest definiowany blok znaków, który jest ruchomy. Przy tworzeniu pierwszej linii takiego ruchomego bloku (pierwsza linia, przy której bit 5 rozkazu ANTIC-u jest "1"), DCTR rozpoczyna liczenie nie od 0, lecz od wartości znajdującej się w VSCROL. Przez to brakuje linii obrazu w górnej części linii. Linia jest przesuwana do góry.

Wszystkie dalsze linie, przy których bit 5 rozkazu ANTIC-u jest "1", są rozpoczynane bezpośrednio po ostatniej linii obrazu poprzedzającej linii. Są one tworzone normalnie. Ponieważ przy pierwszej linii ruchomego bloku brakuje górnych linii obrazu, to cały blok jest przesunięty do góry.

Przy pierwszej linii, przy której bit 5 rozkazu ANTIC-u jest "0", DCTR rozpoczyna liczenie od 0, ale liczy tylko do wartości zawartej w VSCROL. Przez to linia ta wychodzi jakby skrócona od dołu. Pierwsza linia, przy której bit 5 rozkazu ANTIC-u jest "0", jest więc ostatnią linią przesuniętą do góry.

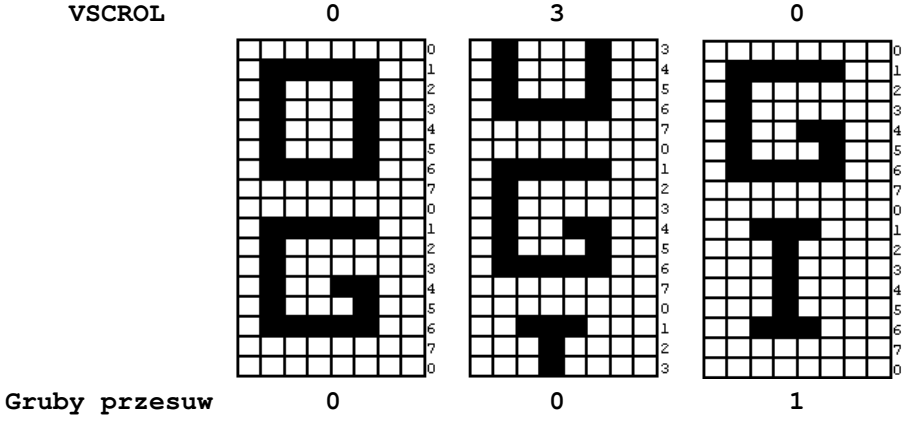
Byłoby to niemądre, wybierać wartość w rejestrze VSCROL (a więc liczbę linii obrazu, o którą blok obrazu jest przesuwany do góry) większą niż maksymalny stan licznika DCTR. Przeważnie nadmiarowe bity są ignorowane, ale w trybie ANTIC-u "3" można ich użyć do innych zastosowań.

Chcąc zrobić ruchomy cały obraz lub tylko dolną część musisz pamiętać jeszcze o dalszych rzeczach: Gdy przy ostatniej linii obrazu bit 5 odpowiedniego rozkazu ANTIC-u jest "1", linia ta nie zwija się na obrazie, lecz wewnątrz. ANTIC nie może wiedzieć, że przy tej linii chodzi o ostatnia przesuwana linię. Aby więc ANTIC prawidłowo wyświetlił tę linię, bit 5 rozkazu ANTIC-u musi być "0". Teraz ANTIC pozna, że ta linia jest ostatnią ruchomą i zwinie ją na ekranie.

Z pomocą rejestru VSCROL można przesunąć obraz tylko o jedna linię, tzn. o 1-15 linii obrazu. Zwykle chce się poruszać obraz po znacznie większej powierzchni. Aby to osiągnąć trzeba zastosować kombinację techniki grubego przesuwania, opisanej wcześniej w przykładzie, z przesuwem precyzyjnym poprzez rejestr VSCROL. Przy przesuwaniu normalnego tekstu dla każdego rozkazu ANTIC-u tworzącego 8 linii obrazu (np. w trybie "2") powinno się najpierw przesunąć do góry 7 linii obrazu. Zamiast przesunięcia ósmej linii (które jest niemożliwe) należy wykonać grube przesunięcie o jedną linię tekstu i ustawić VSCROL ponownie na 0. Przy przesuwaniu w dół należy najpierw wykonać grube przesunięcie całej linii i jednocześnie ustawić wartość 7 w VSCROL. Przesunięcie o następne 7 linii obrazu jest wykonywane poprzez VSCROL.

W ten sposób otrzymujemy obraz, który można o poszczególne linie przesuwać po powierzchni ograniczonej tylko wielkością dostępnej przestrzeni pamięci.

Poniższy rysunek demonstruje zasadę przesuwu pionowego:



Przesuw poziomy

W przykładzie było pokazane, jak przesuwać obraz poziomo o całe bajty pamięci obrazu. Przy przesuwaniu o całe bajty obraz jednak skacze, gdyż każdy bajt tworzy kilka pixeli w jednej linii obrazu, a więc też kilka cykli koloru.

Poniższa tabela pokazuje jeszcze raz, ile cykli koloru i pixeli zawiera jeden bajt pamięci obrazu w poszczególnych trybach ANTIC-u:

Tryb ANTIC-u	Liczba pixeli	Liczba cykli koloru
2	8	4
3	8	4
4	4	4
5	4	4
6	8	8
7	8	8
8	4	16
9	8	16
A	4	8
B	8	8
C	8	8
D	4	4
E	4	4
F	8	4

Na przykład w trybie ANTIC-u "B" każdy bajt pamięci obrazu tworzy w 2 liniach obrazu po 8 pixeli. Mają one razem szerokość 8 cykli koloru. Inaczej mówiąc każdy bajt tworzy w tym trybie 8 cykli koloru w każdej z obu linii obrazu.

ANTIC daje, poprzez rejestr HSCROL, możliwość przesuwania obrazu o poszczególne cykle koloru. Przesuwa on linie obrazu, przy których bit 4 rozkazu ANTIC-u jest "1", o taką liczbę cykli koloru w prawo, jaka wartość znajduje się w rejestrze HSCROL. Znaczące są przy tym jedynie najmłodsze bity rejestru HSCROL. Pozwala to przesuwać obraz tylko o 15 cykli koloru. Dla uzyskania większego przesunięcia trzeba, jak przy przesuwie pionowym, użyć kombinacji przesuwu precyzyjnego i zgrubnego.

Gdy ANTIC przeczyta rozkaz, w którym bit 4 jest "1", opuszcza on liczbę cykli koloru, odpowiadającą zawartości HSCROL, na początku każdej linii obrazu należącej do tego rozkazu ANTIC-u.

Dla tworzenia linii przesuwanych poziomo ANTIC wymaga więcej danych, niż dla tworzenia normalnych linii, gdyż na przykład w trybie z 40 znakami w linii z lewej strony pojawia się 41 znak.

Gdy w rejestrze DMACNTL jest wybrane wąskie pole gry, ANTIC dostaje taką liczbę bajtów pamięci na jedną linię, jaka jest wykorzystywana przy normalnym polu gry.

Przy normalnej szerokości pola gry ANTIC czyta tyle bajtów, ile zwykle potrzeba dla szerokiego pola.

Przy szerokim polu gry nie zmienia się liczba bajtów i jest wstawiany kolor tła.

Przy wykorzystaniu poziomego przesuwu dokładnego trzeba przy każdym rozkazie ANTIC-u ponownie ładować licznik pamięci obrazu (jak w przykładzie). Dlatego tylko warunkowo jest interesujące, ile bajtów czyta ANTIC dla jednej linii. Trzeba jedynie uważać, aby ANTIC nie czytał żadnych danych z miejsca pamięci leżącego za końcem linii. Poza tym ważne jest, aby przy przesuwie poziomym na końcu linii pojawił się następny znak (w trybie z 40 znakami w linii - 41 znak).

GTIA

GTIA jest następnym specjalizowanym układem scalonym Atari. Jego nazwa jest skrótem "Graphic Television Interface Adaptor". GTIA jest głównym ogniwem łączącym ANT1C z telewizorem.

Najważniejszym zadaniem GTIA jest przesyłanie do telewizora danych otrzymanych z ANTIC-u. GTIA dysponuje do tego rejestrami koloru. W każdym rejestrze umieszczony jest zwykle jeden określony kolor z jedną określoną jasnością. Gdy ANTIC tworzy obraz nie podaje on sam koloru, lecz wybiera tylko rejestr koloru GTIA. Jest to normalny sposób tworzenia obrazu, ale jest też możliwe inne interpretowanie przez GTIA danych otrzymanych z ANTIC-u.

Dalszym zadaniem GTIA jest tworzenie na ekranie ruchomych obiektów, tzw. graczy i pocisków (Players and Missiles). Ta forma jest też znana jako Player-Missile Graphic (PMG). Jest możliwe jednoczesne utworzenie 4 graczy i 4 pocisków. Warunkowo jest także możliwe uzyskanie większej ich liczby. GTIA wspiera przez PMG w każdym przypadku tylko jedną linię obrazu, tzn. że dla każdej linii GTIA potrzebuje nowych danych o formie graczy i pocisków, Jak już opisano w rozdziale o ANTIC-u, jest to możliwe automatycznie przez DMA.

W GTIA znajdują się rejestry do wykorzystania dla PMG. Są to, tak jak dla danych przekazanych przez ANTIC, rejestry koloru dla PMG. Poza tym ma on rejestry dla poziomego położenia graczy i pocisków oraz ich wyglądu (tzn. jak wygląda gracz lub pocisk w JEDNEJ linii). Rejestry, w których zapisywany jest wygląd graczy i pocisków, nazywają się rejestrami grafiki graczy lub pocisków. Często są też te rejestry zwane w literaturze rejestrami konturów. Nazwa ta jednak jest błędna. Także pozioma wielkość graczy i pocisków jest zapisywana w rejestrach, a więc programowalna.

Ponieważ obraz z ANTIC-u i dane z rejestrów grafiki graczy i pocisków są łączone w GTIA, to przy obiektach i częściach obrazu zajmujących tą samą pozycję powstaje pytanie, co powinno być utworzone na ekranie, co leży na pierwszym planie, a co w tle. Jest to sterowane za pomocą rejestru, w którym jest zawarta lista priorytetów obiektów wzgl. kolorów ekranu na poszczególnych miejscach.

W grach szczególnie interesujące jest, czy zderzają się ze sobą gracze, pociski z graczami, pociskami lub częściami obrazu. To też wskazują odpowiednie rejestry w GTIA. Przez te funkcje obniża się znacznie wykorzystanie czasu obliczeniowego CPU, ponieważ nie musi ona sama obliczać i sprawdzać czy pozycje obiektów są kolizyjne.

Dodatkowo do opisanych funkcji GTIA obsługuje także wejście triggera (przycisku strzelania). Stan tego wejścia jest sprawdzany przez rejestr GTIA. W starych modelach Atari są wykorzystywane 4 wejścia triggerów GTIA, ponieważ można wykorzystać 4 porty dżojstików. W nowych modelach dwa wejścia triggerów są połączone z portami dżojstików i 1 z gniazdem kartridża. Czwarte wejście triggera jest niewykorzystane.

Z GTIA są połączone także dodatkowe klawisze klawiatury, tzw. klawisze konsoli. Oznacza to, że klawisze START, SELECT i OPTION są sprawdzane przez GTIA.

Dźwięk towarzyszący naciśnięciu klawisza, jest również tworzony przez GTIA. W starych modelach Atari jest on przekazywany przez wbudowany głośnik, nowych modelach dźwięk ten jest teraz mieszany z sygnałem dźwiękowym generowanym przez POKEY. Sygnał wynikowy jest przesyłany poprzez modulator do telewizora.

Wiele adresów jest położonych w dwóch rejestrach GTIA. Jeden z nich używany jest tylko do zapisu, a drugi tylko do odczytu. Tak więc oba rejestry mogą mieć ten sam adres i mimo to nie przeszkadzają sobie.

Dla wielu rejestrów GTIA system operacyjny tworzy rejestry cieni. Zawsze podczas synchronizacji pionowej rejestry i rejestry cieni są

aktualizowane (wzgl. wzajemnie upodabniane). Można to zauważyć na przykład, gdy chcemy zmienić kolor pola gry. Gdy nowa informacja zostanie zapisana do rejestru, który posiada rejestr cieni, jest przepisywana przy następnej synchronizacji pionowej, a więc ukazuje się na ekranie maksymalnie po 1/50 sekundy. Aby podać trwale nowy kolor ekranu trzeba go zapisać do rejestru cieni, ponieważ system operacyjny czyta stąd dane, które zapisuje do rejestru GTIA. Częstym źródłem błędów jest fakt, że zawartość rejestru cieni jest prawidłowa dopiero z opóźnieniem 1/50 sekundy. Czasami występuje więc przypadek, że rejestr GTIA i rejestr cieni mają razem różną zawartość. Prowadzi to do błędów w programie. Użytkownik powinien czytać informacje albo tylko z rejestru GTIA, albo tylko z rejestru cieni.

TWORZENIE DANYCH OBRAZU

Jak już wspomniano, ANTIC przesyła dane, które wskazują GTIA, z którego rejestru ma wyświetlić kolor. W rozdziale o ANTIC-u było już także podane przy opisie różnych trybów graficznych, które bity w danych obrazu muszą być ustawiane, aby uzyskać na ekranie kolor i jasność z określonego rejestru koloru. W trybach graficznych, które mają rozdzielczość dwóch punktów obrazu na cykl koloru, można sąsiednie punkty pokazać tylko z różną jasnością, a nie w różnych kolorach. Na jeden cykl koloru może być utworzony tylko jeden kolor.

Wszystkie rejestry koloru w GTIA są zbudowane jednakowo:

Bit 0: niewykorzystany
Bit 1-3: jasność
Bit 4-7: kolor

Jasność można więc ustawić w 8 stopniach (ponieważ wykorzystuje ona 3 bity). Określa je poniższa tabela:

Bit 3	Bit 2	Bit 1	Jasność
0	0	0	0 minimalna (czarny)
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7 maksymalna (biały)

Wybór odpowiedniego koloru określony jest w poniższej tabeli:

Bit 7	Bit 6	Bit 5	Bit 4	Kolor
0	0	0	0	Szary
0	0	0	1	Złoty
0	0	1	0	Pomarańczowy
0	0	1	1	Czerwono-pomarańczowy
0	1	0	0	Różowy
0	1	0	1	Purpurowy
0	1	1	0	Fioletowy
0	1	1	1	Niebieski 1
1	0	0	0	Niebieski 2
1	0	0	1	Jasnoniebieski
1	0	1	0	Turkusowy
1	0	1	1	Niebiesko-zielony
1	1	0	0	Zielony
1	1	0	1	Żółto-zielony
1	1	1	0	Zielono-pomarańczowy

1	1	1	1	Jasnopomarańczowy

Jak widać do wykorzystania jest 16 różnych kolorów podstawowych. Jednakże bardzo zależy od telewizora, jak który kolor wygląda. Przy źle ustawionym telewizorze mogą być widoczne zamiast białych punktów gęsto koło siebie leżące kolorowe punkty. Tego rodzaju błąd, który polega na złej zbieżności wiązki telewizora, powinien usunąć fachowiec.

Dla kolorów pola gry przeznaczone są 4 rejestry: COLPF0, COLPF1, COLPF2 i COLPF3 (COLor PlayField - kolor pola gry). Poza tym jest jeszcze rejestr COLBAK (COLor BacKground - kolor tła). Kolor z tego rejestru jest zawsze wyświetlany tam, gdzie nie znajduje się żadna inna część obrazu, a praktycznie na krawędziach obrazu. W niektórych trybach graficznych jest ten rejestr używany także do określenia koloru ekranu (np. w trybie ANTIC-u "A" lub "B"). Dla wszystkich tych rejestrów system operacyjny tworzy rejestry cieni.

Rejestry i rejestry-cienie znajdują się w systemie Atari pod następującymi adresami pamięci:

Rejestr			Rejestr-cień		
COLPF0	53270	D016h	COLPF0$	708	02C4h
COLPF0	53271	D017h	COLPF0$	709	02C5h
COLPF0	53272	D018h	COLPF0$	710	02C6h
COLPF0	53273	D019h	COLPF0$	711	02C7h
COLPF0	53274	D01Ah	COLPF0$	712	02C8h

Jak już na początku tego rozdziału wspomniano, jest możliwe, aby GTIA inaczej niż normalnie utworzył dane, które otrzymał z ANTIC-u. Także BASIC zawiera tryby graficzne, które nie pasują do zwykłych trybów pracy ANTIC-u. Są to tryby oznaczone jako "GRAPHICS 9", "GRAPHICS 10" i "GRAPHICS 11".

Początkowo Atari produkował swoje urządzenia z układem nazwanym CTIA, który był przystosowany szczególnie do amerykańskiego systemu telewizji kolorowej (system NTSC). Jasne jest, że Atari nie sprzedałby w Europie żadnego urządzenia, gdyby nie wyposażył ich w układ GTIA. GTIA jest szczególnie przystosowany do europejskiego systemu telewizji kolorowej (PAL). Prawie wszystkie systemy Atari w Europie zawierają GTIA (wszystkie nowe modele i większość starych). Najnowsze urządzenia w USA posiadają bardziej rozwinięty układ zastępujący stary CTIA.

Między GTIA i CTIA jest istotna różnica. Przy grafice z CTIA i ANTIC jest wykorzystywany tylko rejestr grafiki, który jest dostępny poprzez rozkaz "SETCOLOR". Bez PMG pozostają więc niewykorzystane rejestry koloru. Układ GTIA może używać wszystkich dziewięciu rejestrów koloru bez korzystania z PMG, umożliwiając użytkownikowi lepsze wykorzystanie systemu Atari.

Ponieważ w nowych systemach zastosowano tylko GTIA zamiast CTIA, a nie zmieniono ANTIC-u, przymusowo sam GTIA musi być "właściwy" dla nowych trybów (GRAPHICS 9, 10 i 11). Rzeczywiście ANTIC pracuje w tych trybach normalnie, jak np. w trybie tekstowym lub też w trybie graficznym o wysokiej rozdzielczości, jedynie GTIA przetwarza inaczej informację z ANTIC-u.

Nowy GTIA jest kompatybilny z CTIA, tzn. że wszystkie programy, które działały z CTIA działają też z GTIA. Odwrotna droga nie jest jednak możliwa, gdyż CTIA ma mniejsze możliwości.

GTIA interpretuje dane obrazu ANTIC-u inaczej niż w zwykły sposób. Zbiera on dane w czterobitowe kawałki. Te kawałki określają wtedy, które kolory są umieszczane na ekranie. Trzeba jednakże uważać, bo przy użyciu przesuwu poziomego daje to często nieprzewidziane efekty, gdy korzysta się z tych możliwości GTIA.

Różne sposoby pracy są wybierane przez bity 6 i 7 rejestru GTIACNTL. Bity 0-5 tego rejestru są wykorzystywane z PMG. Przy programowaniu należy więc uważać, by nie doszło do konfliktu z PMG. GTIACNTL ma też swój rejestr-cień (GTIACNNI$). Rejestry te znajdują się w następujących miejscach pamięci:

GTIACNTL 53275 D01Bh

GTTACNTL$ 623 026Fh

Możliwe kombinacje bitów 6 i 7 dają następujące funkcje:

Bit 6	Bit 7	Opis
0	0	Ta kombinacja bitów jest zawsze ustawiona przy CTIA. Gdy chcesz uruchomić program z Atari z GTIA na Atari z CTIA, musisz ustawić tę kombinację bitów. W tym przypadku Atari pracuje tak jak omówiono.
0	1	Teraz GTIA interpretuje wspomniane kawałki jako stopnie jasności. Bierze przy tym kolor z rejestru COLBAK wzgl. COLBAK$ i jako jasność kombinację bitów kawałka. Gdy ANTIC przykładowo korzysta z trybu "F" (BASIC: tryb "8"), GTIA łączy po cztery punkty obrazu w pasek wysokości jednej linii obrazu, który ma kolor z rejestru COLBAK. Kombinacja bitów normalnie tworząca 4 punkty określa jasność paska. Ten przykład odpowiada trybowi BASIC-u "9".
1	0	Teraz GTIA interpretuje kawałki jako numery rejestrów koloru. Ale ponieważ GTIA dysponuje tylko dziewięcioma rejestrami koloru, to liczba jednocześnie dostępnych kolorów jest ograniczona do 9. Gdy ANTIC korzysta przykładowo z trybu "F", GTIA łączy po cztery punkty obrazu w pasek wysokości jednej linii obrazu. Kombinacja bitów kawałka określa, z którego rejestru koloru ustawia się kolor i jasność paska. Ten przykład odpowiada trybowi BASIC-u "10".
1	1	Teraz GTIA interpretuje kawałki jako stopnie koloru. Bierze przy tym jasność z rejestru COLBAK wzgl. COLBAK$. Kolor jest wybierany przez kombinację bitów kawałka. Gdy ANTIC korzysta przykładowo z trybu "F", GTIA łączy po cztery punkty obrazu w pasek wysokości jednej linii obrazu. Ten pasek ma wtedy jasność z rejestru COLBAK i kolor określony przez kombinację bitów kawałka. Ten przykład odpowiada trybowi BASIC-u "11".

Dzięki tym możliwościom GTIA zwiększa się liczba kolorów tworzonych jednocześnie na ekranie. Także rozdzielczość jest bardzo wysoka w stosunku do liczby jednocześnie tworzonych kolorów. Proste tworzenie dla statystyki lub gier jest często tak budowane, gdy nie przeszkadza, że obraz składa się z krótkich, poszczególnych linii.

Naturalnie można praktycznie każdy obraz, który jest tworzony przez ANTIC, przetworzyć w GTIA w opisany sposób. Prosto tworzone w trybie tekstowym ANTIC-u obrazy, które potrzebują mało miejsca w pamięci, pozwalają się więc szybko zmieniać przez CPU. Do tego jest jednak konieczne zdefiniowanie osobnego zestawu znaków.

Dalsze możliwości otrzymania interesujących efektów i większej niż zwykle liczby kolorów daje zmiana zawartości rejestrów koloru podczas przerwania programu ANTIC-u. Podczas przerwania umieszcza się w rejestrach koloru nowe wartości. W ten sposób można uzyskać na ekranie więcej kolorów jednocześnie, niż jest rejestrów koloru. Można również zmieniać tryb pracy GTIA podczas przerwań programu ANTIC-u.

Rozkazem BASIC-u "SETCOLOR R,F,H" zapisuje się do rejestrów cieni koloru określoną wartość, która zawiera zakodowany kolor i jasność (kodowanie było już opisane). Możliwym wartościom "R" w rozkazie odpowiadają następujące rejestry-cienie:

dla R=0: COLPF0$

dla R=1: COLPF1$

dla R=2: COLPF2$

dla R=3: COLPF3$

dla R=4: COLBAK$

Zamiast "SETCOLOR R,F,H" można też podać rozkaz "POKE (adres rejestru-cienia),F,16+H". Z pomocą tego rozkazu jest również możliwy wybór koloru dla PMG, czego rozkaz "SETCOLOR" nie pozwala zrobić.

PLAYER-MISSILE GRAPHIC

Do najważniejszych elementów w grach należą oczywiście ruchome obiekty. W tradycyjnych systemach obraz jest prostym obszarem pamięci, który jest wyświetlany na monitorze w jeden określony sposób.

Wyświetlane na ekranie pole, które przedstawia gracza lub pocisk, leży w pamięci przeważnie nie bezpośrednio obok. Między informacjami, które należą do gracza, znajdują się bajty należące do części obrazu, która następuje pomiędzy częścią gracza w jednej linii a częścią gracza znajdującą się w kolejnej linii. Przy każdej operacji z graczem trzeba na to zwracać uwagę. CPU musi więc, gdy na przykład chcesz zmienić wygląd gracza, zmienić tylko każde takie bajty. Daje to znaczne utrudnienie programu.

Może także leżeć gracz w połówce bajta lub bajtów, przez co CPU musi odróżniać pojedyncze bity należące i nienależące do gracza. Daje to dalsze utrudnienie.

Oprócz tego każdy gracz zasłania część obrazu lub jest przez część obrazu zasłaniany. Każda zasłonięta część obrazu lub każdy zasłonięty gracz musi po dalszym ruchu być ponownie utworzony. Przy każdym przepisaniu obrazu musi więc zostać zmagazynowane to, co zostało zasłonięte.

Całkowite utrudnienie daje się zmniejszyć przez to, że gracz jest poruszany zawsze o kilka punktów. Czyni to jednak ruch niezbyt płynnym, lecz skokowym. Rozwiązanie daje wprowadzenie graczy i pocisków, które są włączane nie programowo lecz sprzętowo. Także kontrolowanie spotkań (kolizji) jest łatwiejsze sprzętowo niż programowo.

W urządzeniach Atari obejmuje to tzw. Player-Missile Graphic. Pomimo, że wysiłek nie jest zmniejszony do zera i Atari nie daje najprostszej idei ruchomych obiektów, powinno się zawsze pamiętać, które operacje CPU się oszczędza. Poza tym PMG Atari jest zupełnie odmienna.

Z PMG jest możliwe włączenie do czterech graczy i czterech pocisków do każdego obrazu ANTIC-u. Liczbę tę można jednak zwiększyć, można też więcej obiektów tworzyć z jednego gracza lub pocisku.

W jednym wierszu można jednak utworzyć nie więcej niż 4 pociski i 4 graczy.

Gdy dwa obiekty wykorzystują wspólnie jednego gracza lub jeden pocisk, mogą być zapisane w rejestrze grafiki dane jednego lub drugiego obiektu, zależnie od tego, która linia obrazu jest wykonywana. Jest także możliwe tworzenie dwóch obiektów przez jednego gracza poprzez DMA. Trzeba

jedynie w linii obrazu spowodować przerwanie programu ANTIC-u i w procedurze przerwania ustawić rejestr PMBASE (rejestr ten jest opisany w rozdziale o ANTIC-u) na adresie bazowym innego pola pamięci.

PMG jest włączana i wyłączana przez rejestr PMCNTL, znajdujący się pod adresem 53277 (D01Dh). Poszczególne bity tego rejestru mają następujące funkcje:

Bit 0: Gdy ten bit jest "1", to jest włączone przeniesienie rejestru grafiki pocisku na ekran. Tak więc ten bit włącza pociski.

Bit 1: Gdy ten bit jest "1", to jest włączone przeniesienie rejestru grafiki gracza na ekran. Tak więc ten bit włącza graczy.

Bit 2: Gdy ten bit jest "1", to wejście przycisku dżojstika może być ustawione tylko na "0", więc położenie przycisku nie jest rejestrowane. Dzięki temu jest możliwe blokowanie naciśnięć klawiszy na przyciskach dżojstików.

Kolory graczy są zapisywane tak jak kolory pól gry w rejestrach koloru. Przy tym zawsze jeden gracz i jeden pocisk mają wspólny rejestr koloru. Rejestry te są w swoich funkcjach identyczne z innymi rejestrami kolorów. Rejestry znajdują się pod następującymi adresami:

COLPM0	53266	D012h	Gracz/pocisk 0
COLPM0$	704	02C0h	
COLPM1	53267	D013h	Gracz/pocisk 1
COLPM1$	705	02C1h	
COLPM2	53258	D014h	Gracz/pocisk 2
COLPM2$	706	02C2h	
COLPM3	53259	D015h	Gracz/pocisk 3
COLPM3$	707	02C3h	

Kształty graczy są umieszczone w rejestrach PMG. Lecz GTIA tworzy tylko jedną linię obrazu w danym czasie. Oznacza to, że po każdej linii, o ile obraz gracza w tej linii zmienia się, trzeba podać nowe dane do rejestrów grafiki graczy i pocisków. Można to wykonać zarówno programem assemblera jak i przez DMA ANTIC-u, którego użycie opisane jest w rozdziale o ANTIC-u.

Dla każdego gracza jest przewidziany jeden rejestr grafiki:

GRAFP0	53261	D00Dh
GRAFP1	53262	D00Eh
GRAFP2	53263	D00Fh
GRAFP3	53264	D010h

Dla pocisków jest przewidziany jeden wspólny rejestr grafiki, w którym są zapisane dane wszystkich pocisków:

GRAFPM	53265	D011h

Do każdego pocisku należą dwa bity w tym rejestrze:

Bity 0-1	pocisk 0
Bity 2-3	pocisk 1
Bity 4-5	pocisk 2
Bity 6-7	pocisk 3

Programowanie rejestrów pocisków jest nieco bardziej skomplikowane niż programowanie rejestrów grafiki graczy, gdyż tu poszczególne bity pocisków muszą być złożone w jeden bajt. Nawet DMA nie ułatwia tej pracy. Ale pomimo tego programowanie gier przy użyciu PMG jest łatwiejsze niż w ogóle bez sprzętowej pomocy dla ruchomych obiektów.

Pozioma pozycja pocisków i graczy jest również zapisywana w rejestrach. Zawartość rejestru określa w każdym przypadku, przy którym cyklu koloru linii obrazu rozpoczyna się podawanie zawartości rejestru grafiki (a więc obiektu) na ekran.

Trzeba jednak zauważyć, że lewa krawędź pola gry zależnie od liczby cykli koloru w linii obrazu, która jest tworzona w ANTIC-u, leży w różnych cyklach koloru. Cykle koloru przed właściwym obrazem należą do ramki obrazu, która ma kolor z rejestru COLBAK. Pierwsze cykle koloru leżą normalnie poza ekranem telewizora. Aby uczynić obiekty niewidocznymi trzeba rejestr poziomych pozycji graczy ustawić na zero.

Dla każdego gracza i pocisku jest przewidziany rejestr poziomej pozycji:

```
HPOSP0 53248 D000h gracz 0          HPOSM0 53252 D004h pocisk 0
HPOSP1 53249 D001h gracz 1          HPOSM1 53253 D005h pocisk 1
HPOSP2 53250 D002h gracz 2          HPOSM2 53254 D006h pocisk 2
HPOSP3 53251 D003h gracz 3          HPOSM3 53255 D007h pocisk 3
```

Poza tym mamy następny rejestr, który wpływa na pionową pozycję graczy i pocisków:

```
VDELAY      53276       D01Ch
```

VDELAY jest używany, gdy została wybrana rozdzielczość dwuwierszowa, ale pomimo to chcemy podać obiekt w jednej określonej linii. Przez ustawienie bitu na "1" powoduje się, że odpowiedni obiekt jest± przesunięty w dół o jedną linię. Bity tego rejestru mają następujące znaczenia:

```
Bit 0       pocisk 0
Bit 1       pocisk 1
Bit 2       pocisk 2
Bit 3       pocisk 3
Bit 4       gracz 0
Bit 5       gracz 1
Bit 6       gracz 2
Bit 7       gracz 3
```

Pozioma wielkość obiektów także jest zapisana w rejestrach. Pozioma wielkość jest to liczba cykli koloru, którą obiekt zajmuje w jednej linii.

Dla każdego gracza jest jeden taki rejestr:

```
SIZEP0      53256       D008h       gracz 0
SIZEP1      50257       D009h       gracz 1
SIZEP2      53258       D00Ah       gracz 2
SIZEP3      53259       D00Bh       gracz 3
```

Wielkość pozioma jest określana przez bity 0 i 1 tych rejestrów (pozostałe bity są niewykorzystane):

Bit 1	Bit 0	Wielkość
0	0	Normalna wielkość, każdy bit ma szerokość jednego cyklu koloru (gracz ma więc szerokość 8 cykli koloru).
0	1	Podwójna wielkość, każdy bit ma szerokość dwóch cykli koloru (gracz ma więc szerokość 16 cykli koloru).
1	0	Normalna wielkość, każdy bit ma szerokość jednego cyklu koloru (gracz ma więc szerokość 8 cykli koloru).
1	1	Poczwórna wielkość, każdy bit ma szerokość czterech cykli koloru (gracz ma więc szerokość 32 cykli koloru).

Wielkość pocisków jest zapisana w rejestrze:

SIZEM 53260 D00Ch

Po dwa bity tego rejestru określają wielkość pocisków:

Bity 0-1 pocisk 0
Bity 2-3 pocisk 1
Bity 4-5 pocisk 2
Bity 6-7 pocisk 3

Możliwe kombinacje bitów działają następująco:

0	0	Normalna wielkość, każdy bit ma szerokość jednego cyklu koloru (pocisk ma więc szerokość 2 cykli koloru).
0	1	Podwójna wielkość, każdy bit ma szerokość dwóch cykli koloru (pocisk ma więc szerokość 4 cykli koloru).
1	0	Normalna wielkość, każdy bit ma szerokość jednego cyklu koloru (pocisk ma więc szerokość 2 cykli koloru).
1	1	Poczwórna wielkość, każdy bit ma szerokość czterech cykli koloru (pocisk ma więc szerokość 8 cykli koloru).

Powstaje pytanie, co będzie utworzone z przodu a co z tyłu, gdy część obrazu ma taką samą pozycję jak gracz lub pocisk. Określane jest to przez rejestr GTIACNTL (wzgl. GTIACNTL$). Bity 6 i 7 określają sposób pracy GTIA. Cztery pierwsze bity służą do kontroli priorytetów. Przez ustawienie jednego z czterech bitów wybiera się kolejność priorytetów między kolorami pola gry a graczami i pociskami :

Bit 0: P0, P1, P2, P3, PF0, PF1, PF2, PF3/P5, BAK
Bit 1: P0, P1, PF0, PF1, PF2, PF3/P5, P2, P3, BAK
Bit 2: PF0, PF1, PF2, PF3/P5, P0, P1, P2, P3, BAK
Bit 3: PF0, PF1, P0, P1, P2, P3, PF2, PF3/P5, BAC

Priorytet zmniejsza się od lewej do prawej!

Gdy jest ustawiony więcej niż jeden z czterech bitów, priorytety nie są jednoznacznie określone. Prowadzi to do konfliktów priorytetów między poszczególnymi obiektami obrazu. Pokrywające się strefy takich, będących w konflikcie priorytetów obiektów są wyświetlane w kolorze czarnym. Gdy na przykład ustawimy bity 0 i 2 w GTIACNTL, to P0 jest wyświetlane w kolorze czarnym, gdy wypadnie na PF0. To samo dotyczy też P1, P2 i P3, gdy wypadną one na PF1, PF2 i PF3/P5.

W trybie ANTIC-u z jednym kolorem i 40 znakami w linii jest określana jasność znaku bez względu na priorytet PF1. Gdy gracz lub pocisk z wyższym priorytetem pokryje taki znak, to kolor tej strefy jest określony przez kolor gracza.

Pozostałe bity GTIACNTL mają następujące funkcje:

Bit 4: Gdy ten bit jest "1", to wszystkie pociski mają kolor określony przez COLPF3. Dzięki temu można połączyć pociski w

piątego gracza /P5/.

Bit 5: Ustawienie tego bitu umożliwia wyświetlanie wielokolorowych graczy. Przy tym gracze 0 i 1 jak również 2 i 3 są związane ze sobą, o ile się pokrywają. Priorytety są ważne tylko warunkowo. Pozwala to wyświetlać kolory obu graczy. Zależnie od tego, który bit którego gracza jest ustawiony, wyświetlany jest jeden z trzech kolorów (dwa kolory graczy i kolor pod graczami).

Szczególnie interesujące w grach jest ustalenie, czy występują "zderzenia" pomiędzy graczami i pociskami a innymi graczami, pociskami i częściami obrazu. GTIA posiada 16 rejestrów, które mogą być odczytywane przez program w celu sprawdzenia, czy występują takie zderzenia.

Każdy pocisk ma rejestr, który zawiera dane o kolizjach pocisku z kolorami pola gry:

KOLM0PF:	53219	D000h
KOLM1PF:	53249	D001h
KOLM2PF	53250	D002h
KOLM3PF	53251	D003h

Bit tego rejestru sygnalizuje ustawieniem na "1", że występuje kolizja między pociskiem, któremu służy ten rejestr, a danym kolorem pola gry:

Bit 0: kolizja z PF0
Bit 1: kolizja z PF1
Bit 2: kolizja z PF2
Bit 3: kolizja z PF3
Bit 4-7: niewykorzystane (ustawione na "0")

Także każdy gracz ma rejestr, który sygnalizuje kolizje między danym graczem a kolorem pola gry:

KOLP0PF	53252	D004h
KOLP1PF	53253	D005h
KOLP2PF	53254	D006h
KOLP3PF	53255	D007h

Bit tego rejestru sygnalizuje ustawieniem na "1", że występuje kolizja między graczem, któremu służy ten rejestr, a danym kolorem pola gry:

Bit 0: kolizja z PF0
Bit 1: kolizja z PF1
Bit 2: kolizja z PF2
Bit 3: kolizja z PF3
Bit 4-7: niewykorzystane (ustawione na "0")

Każdy pocisk ma rejestr, który sygnalizuje kolizje między danym pociskiem a graczem:

KOLM0P	53256	D008h
KOLM1P	53257	D009h
KOLM2P	53258	D00Ah
KOLM3P	53259	D00Bh

Bit tego rejestru sygnalizuje ustawieniem na "1", że występuje kolizja między pociskiem, któremu służy ten rejestr, a danym graczem:

Bit 0: kolizja z graczem 0 (PO)
Bit 1: kolizja z graczem 1 (P1)
Bit 2: kolizja z graczem 2 (P2)
Bit 3: kolizja z graczem 3 (P3)
Bit 4-7: niewykorzystane (ustawione na "0")

Wreszcie każdy gracz ma rejestr, który sygnalizuje kolizje z innymi graczami:

KOLP0P	53260	D00Ch
KOLP1P	53261	D00Dh

KOLP2P 53262 D00Eh
KOLP3F 53263 D00Fh

Bit tego rejestru sygnalizuje, że występuje kolizja między graczem, któremu służy ten rejestr, a innym graczem. "Własny" bit w danym rejestrze jest zawsze "0".

Bit 0: kolizja z graczem 0 (P0)
Bit 1: kolizja z graczem 1 (P1)
Bit 2: kolizja z graczem 2 (P2)
Bit 3: kolizja z graczem 3 (P3)
Bit 4-7: niewykorzystane (ustawione na "0")

Kolizja jest zaznaczana w chwili, w której pozycja, na której ma miejsce kolizja, jest wyświetlana na ekranie. Bezpośrednio po wpisaniu w rejestrze pozycji ściśle biorąc kolizja już ma miejsce, jednak ze względu a RD5na wewnętrzną strukturę GTIA niemożliwe jest wskazanie kolizji w tym momencie.

Bity w rejestrach kolizji pozostają ustawione tak długo, aż jakakolwiek wartość zostanie zapisana w rejestrze "kasującym". Poszczególne bity tego rejestru nie mają żadnych innych funkcji. Jest więc zupełnie obojętne jaka wartość zostanie zapisana.

HITCLR 53278 D01Eh

Kolizja jest więc ewentualnie zapisana jeszcze w chwili, w której obiekty przesunęły się dalej i znajdują się już w różnych miejscach. W ten sposób można też wykonywać ruchy obiektów w procedurach przerwania, a sprawdzanie kolizji robić dopiero później w głównym programie.

Jak konkretnie używać Player-Missile Graphic?

Najpierw trzeba zdecydować się, czy obiekty będą tworzone przez DMA ANTIC-u, czy też zapisywane w rejestrach grafiki GTIA bezpośrednio z programu assemblera. Zazwyczaj używa się DMA, gdyż inaczej trzeba bardzo uważać, która linia jest właśnie wyświetlana. Wysiłek, który wynika, gdy pracuje się z PMG bez DMA, tylko bardzo rzadko jest usprawiedliwiony.

Jeżeli używa się PMG z DMA, trzeba znaleźć wolny obszar pamięci. Zazwyczaj całkiem "u góry" wolnej pamięci operacyjnej leży pamięć obrazu. Bezpośrednio pod nią umieszczony jest zwykle program ANTIC-u. Zaleca się ulokowanie pola pamięci dla PMG pod programem ANTIC-u. W razie umieszczania dalszych pól pamięci dla PMG powinno się wypełniać pamięć dalej z góry do dołu. Adres pola pamięci dla PMG trzeba podać ANTIC-owi przez rejestr PMBASE. Przy wyborze obszaru pamięci PMG trzeba jednak uważać, aby - jak już mówiono - zawsze rozpoczynał się na granicy 1KB wzgl. 2KB.

Poziomy ruch gracza można osiągnąć bezpośrednio przez zmianę rejestru pozycji GTIA. Dla uzyskania pionowego ruchu gracza trzeba przesuwać bajty gracza lub poszczególne bity pocisku przez pole pamięci dla PMG-DMA. Do tego powinno się tworzyć podprogramy. Podprogramy te są względnie proste, gdyż obszar pamięci dla jednego gracza lub pocisku ma nie więcej niż 256 bajtów, więc nie jest większy niż wielkość wartości dającej się przedstawić w rejestrze mikroprocesora 6502C (rejestr A, X lub Y). Odpadają więc operacje 16-bitowe, które są konieczne bez PMG.

Teraz trzeba skasować obszar pamięci dla PMG. Poza tym trzeba ustalić poziome pozycje i wielkości graczy, jak również stopień rozdzielczości pionowej. Trzeba także umieścić w rejestrach COLPM0$-COLPM3$ kolory graczy.

Na samym końcu należy włączyć PMG-DMA (przez rejestr DMACNTL) i PMG (przez rejestr PMCNTL); tak aby nie wystąpiły żadne przeszkody na ekranie w momencie rozpoczęcia.

WEJŚCIA PRZYCISKÓW I KLAWISZE KONSOLI

Wejścia przycisków GTIA są w starych modelach Atari związane z czterema wejściami dżojstików. Przez wejścia przycisków są sprawdzane "przyciski ognia" dżojstików.

Ponieważ nowe modele Atari mają tylko dwa porty dżojstików, to jedno z wolnych wejść przycisków służy do sprawdzania zezwolenia dla kartridża. Czwarte wejście przycisku jest niewykorzystane, ale można je zaprogramować np.jako dodatkowe wyjście dźwięku.

Każde wejście przycisku ma do dyspozycji rejestr, którego najmłodszy bit (bit 0) podaje status wejścia. "1" w tym bicie wskazuje na przykład, że przycisk w dżojstiku nie jest naciśnięty, a "0", że jest naciśnięty. Jak już powiedziano, można ukryć przełączenie na "1" przez ustawienie bitu 2 w rejestrze PMCNTL. Gdy w jednym z rejestrów wejść przycisków ustawione jest "0", to pozostaje zapisane aż do skasowania bitu 2 w PMCNTL. Bity 1-7 tych rejestrów są zawsze "0". Każdy z tych rejestrów ma rejestr-cień.

Rejestry i rejestry-cienie znajdują się pod następującymi adresami:

```
TRIG0   532E4  D010h     DŻOJSTIK 1
TRIG0$   E44   0284h
TRIG1          532E5     D011h    DŻOJSTIK 2
TRIG1$           E45     0285h
TRIG2          532EE     D012h    Zezwolenie kartridża RD5
TRIG2$           E4E     0286h    (dżojstik 3)
TRIG3          532E7     D013h    niewykorzystane
TRIG3$           E47     0287h (dżojstik 4)
```

(w nawiasach: Atari 400/800)

Klawisze konsoli "START", "SELECT" i "OPTION" są również sprawdzane przez GTIA. Służy do tego rejestr:

```
CONSOL      53279 D01Fh
```

Klawisze konsoli są przyporządkowane bitom rejestru następująco:

```
Bit 0:      START
Bit 1:      SELECT
Bit 2:      OPTION
```

"0" w jednym z tych bitów sygnalizuje, że naciśnięty jest odpowiedni klawisz. Przez wpisanie "1" w jeden lub kilka bitów blokuje się działanie odpowiadających im klawiszy.

Bit 3 rejestru CONSOL steruje klikiem klawiatury. Podczas cyklu wygaszania pionowego system operacyjny wpisuje "1" w ten bit, przez wpisanie "0" w ten bit wywołuje się klik. Bit ten może być tylko krótkotrwale ustawiony na "0", szczególnie gdy wyłącza się procedura przerwania wygaszania pionowego systemu operacyjnego.

Przed odczytem klawiszy konsoli w rejestr CONSOL trzeba wpisać wartość 08h, aby mieć pewność, że żaden z klawiszy nie jest zablokowany.

W systemach PAL i NTSC wygląd kolorów nie jest niestety identyczny. Gdy gra jest napisana na system PAL, bywa czasami konieczna zmiana kolorów na system NTSC przed uruchomieniem programu. Zmiana ta może być także dokonana w programie. Sprawdza się wtedy podczas programu w rejestrze:

```
NTSCPAL     532F8       D014h
```

czy urządzenie pracuje w systemie PAL, czy w NTSC. Przy systemie PAL bity 1-3 są "0", a przy NTSC - "1".

Poza tym system NTSC ma inną liczbę linii obrazu na ekranie niż system PAL.

POKEY

Układ kontroli potencjometrów i klawiatury (Potentiometer-KEYboard controller chip) jest trzecim specjalizowanym układem w Atari 600XL/800XL. Bierze on na siebie przede wszystkim prawie całe sterowanie dźwiękiem, po drugie służy do komunikacji z urządzeniami zewnętrznymi i w końcu odczytuje kanały potencjometrów (paddle).

Jako połączenie z CPU wykorzystuje on 8 linii danych, linię zegara, linię zapisu/odczytu, linię przerwań i 4 linie adresowe. POKEY nie jest obsługiwany przez sprzętową linię resetowania.

Dzięki czterem liniom adresowym można w POKEY-a wskazać 16 rejestrów danych wzgl. sterowania.

Te szesnaście rejestrów dzieli się na 9 rejestrów dźwięku, 4 rejestry przeniesienia, 1 rejestr przerwań i jeden nie zajęty.

Tworzenie dźwięku

POKEY posiada możliwość zarządzania maksymalnie czterema kanałami dźwięku jednocześnie.

Każdy dźwięk, który możemy usłyszeć, jest złożony z wielu składników. Jest to przede wszystkim wysokość dźwięku, która technicznie jest zapisywana jako częstotliwość i mierzona w Hz (Hertz od nazwiska znanego fizyka Heiricha Hertza). Jeden Hz jest więc jednym pełnym drganiem na sekundę, przy czym "pełny" oznacza, że na obrazie drgania wybiera się określony punkt i czeka tak długo, aż ten punkt zostanie znowu osiągnięty. Okres między tymi dwoma punktami jest wtedy pełnym drganiem. Gdy jest 1000 drgań na sekundę, to oznacza, że częstotliwość jest 1000 Hz lub też 1kHz.

Im większa jest częstotliwość sygnału, tym wyższy dźwięk wzgl. szum słychać.

Dalszą charakterystyką drgań jest kształt drgania. Skrajnościami są tu n.p, czyste drganie sinusoidalne lub, jako jego przeciwieństwo, czyste drganie prostokątne, Są to jednak przypadki specjalne, które wprawdzie obecnie łatwo tworzyć dzięki elektronicznym środkom, jednak ze względu na ich wyrazistość są "nerwowe" i nie dają się długo słuchać. Jeżeli ktoś nie wierzy, powinien posłuchać przez pięć minut dźwięku 440 Hz (kamertonu) w telefonie.

Wszystkie szumy i tony, które obecnie słyszymy, są mieszaninami różnych sygnałów. Każdy złożony sygnał możemy, biorąc pod uwagę również tzw. krzywą obwiedni, dokładnie odróżnić ze względu na odmienne źródła sygnałów. Tak więc dość rzadko może nastąpić pomyłka między waltornią i fortepianem.

Krzywa obwiedni opisuje, jak zmienia się natężenie (głośność) jakiegoś dźwięku w czasie. Daje to np. sygnał, który bardzo szybko staje się głośny, a następnie powoli i równomiernie zanika (fortepian), natomiast skrzypce znacznie wolniej osiągają maksymalną głośność.

Jako ostatnią istotną własność każdego dźwięku omówimy mniej lub bardziej nieświadome zakresy częstotliwości, w których znajduje się w przybliżeniu mieszanina sygnałów. Słyszy się zwykle inaczej, gdy muzyka dobiega bezpośrednio z urządzenia Hi-Fi, niż kiedy to czyni przez telefon. Nie ze względu na ewentualne zakłócenia i trzaski, lecz z powodu znacznie mniejszego zakresu przenoszonych częstotliwości. Jest to przez konstrukcję sieci tak określone, że z przenoszonych dźwięków są przepuszczane głównie dźwięki o wysokości koniecznej do zrozumienia mowy.

W POKEY-u zawarte są 4 rejestry, które służą tylko do generowania określonej częstotliwości sygnału poprzez podział określonej częstotliwości wejściowej. Przez to właśnie powstaje najniższy ton. Ponieważ jest to jednak zbyt prymitywne, a po tym, co wcześniej powiedziano, nie pozwala osiągnąć rozsądnych dźwięków, są jeszcze dalsze możliwości. Pierwszą jest np. związanie dwóch takich liczników, to znaczy dołączyć wyjście jednego licznika do wejścia drugiego. Można tak związać licznik 2 z 1, jak również 4 z 3. Połączenie to ma następujący cel:

Pojedynczy licznik ośmiobitowy pozwala podzielić częstotliwość wejściową w zakresie od 1 do 256 (wewnętrznie jest dodawane 1 do zawartości licznika). Łącząc razem dwa liczniki otrzymujemy możliwość podziału od 1 do 65536, ponieważ teraz mamy do dyspozycji 16 bitów.

Mamy zatem możliwość wykorzystania czterech kanałów 8-bitowych, lub jednego kanału 16-bitowego i dwóch 8-bitowych, albo w końcu dwóch kanałów 16-bitowych, Wszystko to jest ustalane przez odpowiednie bity statusu. Właściwe wartości stosunków podziału są zapisane rejestrach od AUDIFREQ1 do AUDIFREQ4.

Jako częstotliwość wejściową można wybrać dla każdego rejestru częstotliwość 64kHz, 15kHz lub (tylko dla kanałów 1 i 3) cykl 1,77 MHz. Im wyższa częstotliwość jest wybrana, tym wyższy jest przy takim samym stosunku podziału sygnał wyjściowy.

Z kombinacji częstotliwości wejściowych i stosunków podziału można otrzymać wszystkie zakresy częstotliwości od 0 do 1,77/2MHz = 887kHz. Ostatni podział przez 2 zachodzi zawsze i ma na celu uzyskanie wyłącznie symetrycznych sygnałów. Podzielenie sygnału cyfrowego przez 2 na końcu powoduje, że tak samo długo "rośnie" i "opada", a więc jest symetryczny. Warunek ten jest istotny, ponieważ sygnał niesymetryczny słychać zupełnie inaczej niż symetryczny, bowiem wraz ze zmianą częstotliwości zmienia się barwa dźwięku.

Obok częstotliwości można ustawić jeszcze rodzaj i kształt ewentualnych zniekształceń. Do tego normalny sygnał wyjściowy, otrzymywany z liczników może być łączony z cyklicznie występującymi zniekształceniami.

Zniekształcenia te są tworzone przez rejestr przesuwny, który działa jak długa rura. Sygnał dźwiękowy wpadający do jednego końca rury pojawia się na drugim końcu z opóźnieniem zależnym od długości rury jako sygnał wyjściowy. Podanie sygnału wyjściowego ponownie na wejście spowoduje zdudnienie. Aby nie tworzyć nudnych jednostajnych rytmów używa się jeszcze jednego tricku: rura jest w niektórych miejscach "przewiercona" i znajdujące się w tych miejscach sygnały są łączone z sygnałem wyjściowym. Dzięki temu jest zawsze zapewnione, że przejście jest cykliczne, ale jego okres powtórzeń jest znacznie zwiększony.

W POKEY-u są trzy takie rejestry przesuwne, które są określane jako liczniki poly. Jeden ma długość 4 bitów, drugi 5 bitów, a trzeci przełączany 17 albo 9 bitów.

Częstotliwość jest określana, jak wcześniej powiedziano, przez wartości w rejestrach od AUDIFREQ1 do AUDIFREQ4.

Zniekształcenia i szesnastostopniowa głośność jest programowana przez rejestry kontroli AUDICNTL1 do AUDICNTL4. Poszczególne bity tych rejestrów kontroli oznaczają:

Bity 0-3: **Te cztery bity kontrolują głośność. Mogą one zawierać wartości od 00h do 0Fh (0000-1111).**

Bit 4: **VOLUME_ONLY**

Bity 5-7: **Te trzy bity służą do wybrania jednej z trzech możliwości zniekształceń.**

Tabela wartości bitów 5, 6 i 7 rejestrów AUDICNTLn:

Bit 7	Bit 6	Bit 5	Mieszanie sygnałów
0	0	0	Pochodzący z rejestru liczącego sygnał jest najpierw łączony z wielomianem 5-bitowym, następnie z 17-bitowym, a w końcu dzielony przez dwa.
0	X	1	Zachodzi tu tylko połączenie z wielomianem 5-bitowym i podział przez dwa.
0	1	0	Pochodzący z rejestru liczącego sygnał jest najpierw łączony z wielomianem 5-bitowym, następnie z 4-bitowym, i na końcu dzielony przez dwa.
1	0	0	Częstotliwość wyjściowa jest łączona z sygnałem wielomianu 17-bitowego i na końcu

			dzielona przez dwa.
1	X	1	Te dwie kombinacje dają czysty ton, gdyż do sygnału wyjściowego nie jest dołączany żaden sygnał wielomianu, a występuje tylko końcowe dzielenie przez dwa.
1	1	0	Zachodzi tu tylko połączenie z wielomianem 4-bitowym. Potem, jak we wszystkich przypadkach, wartość jest dzielona przez dwa.

"X" w tabeli oznacza, że może tu być "0" lub "1".

Bit VOLUME_ONLY z każdego z czterech rejestrów AUDICNTLn wyłącza całą wewnętrzną "maszynerię" liczenia, zniekształcania i łączenia oraz określa, że jako sygnał wyjściowy powinno być wysyłane napięcie dokładnie odpowiadające głośności. Gdy więc jest ustawione VOLUME_ONLY i głośność na 16 (pełna głośność), to membrana głośnika znajdującego się w telewizorze przesuwa się z położenia spoczynkowego tak daleko, jak to jest możliwe. Ustawienie teraz głośności ponownie na zero powoduje powrót membrany do położenia spoczynkowego, co daje trzask w głośniku.

Zadaniem tego dodatkowego urządzenia jest naturalnie nie tylko tworzenie trzasków. Dzięki tej metodzie programista nie podlega żadnym ograniczeniom odnośnie kształtu i częstotliwości sygnału, ponieważ może programować każdy pojedynczy ruch membrany głośnika. Należy jednak przy zbyt szybkich zmianach amplitudy (amplituda - wielkość odchylenia przy drganiu) zwrócić uwagę na to, że wprawdzie Atari może taki sygnał wydać, jednak ostre piki sygnału będą znacznie stłumione z uwagi na ograniczone pasmo przenoszenia telewizora. Nie stwarza to jednak w normalnych przypadkach żadnych ograniczeń, gdyż obecnie telewizory najczęściej mają już jakość Hi-Fi.

Można powiedzieć, że z odpowiednio programując można otrzymać z POKEY-a wszystkie pożądane efekty akustyczne bez stosowania dodatkowych urządzeń.

Można np. dzięki programowi S.A.M., który był już stworzony dla modeli 400/800, bez dodatkowego osprzętu uzyskać dźwięki naśladujące mowę. Mowa ta jest jednak średnio zrozumiała.

Oprócz tych czterech rejestrów obsługujących poszczególne kanały jest jeszcze jeden rejestr istotny dla tworzenia dźwięków: AUDIOCOM.

Rejestr ten zawiera 8 następujących bitów sterujących:

BIT	Opis
0	Gdy jest ustawiony, to dla czterech rejestrów częstotliwości ponownie przełącza częstotliwość wejściową z 64kHz na 15kHz.
1	Gdy ten bit jest "1", to do wyjścia kanału 2 jest dołączony filtr wysokiej częstotliwości, którego charakterystykę określa kanał 4.
2	Dołącza do wyjścia kanału 1 filtr wysokiej częstotliwości określony przez kanał 2.
3	Przez ustawienie tego bitu łączone są kanały 4 i 3 tworząc rejestr 16-bitowy, .
4	Przez ustawienie tego bitu są łączone kanały 2 i 1.
5	Ten bit powoduje, że kanał 3 jest sterowany częstotliwością podstawową 1,77MHz,
6	Jak bit 5, lecz dla kanału 1.
7	Przez ustawienie tego bitu wielomian 17-bitowy jest zamieniany na wielomian o długości 9 bitów, ,

Stosunek częstotliwości wyjściowej do wejściowej opisuje równanie:

$$F_{out} = \frac{F_{in}}{2(AUDIFREQn + Off)}$$

przy czym Off otrzymuje wartość 4, gdy używany jest rejestr 8-bitowy, gdy jest 16-bitowy, to Off musi mieć wartość 7.

Te techniki, oprócz tworzenia dźwięków, mają jeszcze jedno zadanie.

Gdy jeden z liczników 1, 2 lub 4 doliczy ponownie do zera, automatycznie jest przez POKEY wywoływane przerwanie. Licznik może więc całkiem dobrze funkcjonować jako zegar (timer).

Aby we wszystkich rejestrach ustawić wartość początkową (jest to ważne przede wszystkim przy pracy jako timer!), najprościej jest wpisać tylko jedną wartość do rejestru STIMER.

Jako "produkt uboczny" liczników poly powstaje w rejestrze RANDOM ośmiobitowa liczba losowa. Składa się ona z 8 najwyższych bitów licznika poly 17-/9-bitowego.

Na zakończenie tego tematu przedstawiona jest jeszcze tabela, w której podane są stosunki zawartości rejestrów AUDIFREQn dla określonych wartości nut. Przy zawartości rejestrów AUDIOCOM=00h i AUDICNTL=AXh, gdzie X jest poziomem głośności od 0 do 15 otrzymuje się następujące wartości dla AUDIFREQn:

Nuta	AUDIFREQn	
	HEX	DEC
C	F3	243
Cis	E6	230
D	D9	217
Dis	CC	204
E	C1	193
F	B6	182
Fis	AD	173
G	A2	162
Gis	99	153
A	90	144
B	88	136
H	80	128
c	79	121
cis	72	114
d	6C	108
dis	66	102
e	60	96
f	5B	91
fis	55	85
g	51	81
gis	4C	76
a	48	72
b	44	68
h	40	64
c′	3C	60
cis′	39	57
d′	35	53
dis′	32	50
e′	2F	47
f′	2D	45
fis′	2A	42
g′	28	40
gis′	25	37

a′	23	35
b′	21	33
h′	1F	31
c″	1D	29

Jest też praktycznie możliwe otrzymanie innych częstotliwości. Atari dysponuje nie tylko zakresem częstotliwości 3,5 oktawy. Najwyższe możliwe do uzyskania częstotliwości leżą powyżej granicy słyszalności. Także najniższe możliwe do wytworzenia częstotliwości nie są słyszalne.

OBSŁUGA KLAWIATURY

Kiedy na klawiaturze 600XL/800XL zostanie naciśnięty klawisz, który nie jest specjalnym klawiszem bocznym ani klawiszem SHIFT lub CONTROL, przez POKEY wywoływane jest przerwanie, które informuje system operacyjny o naciśnięciu klawisza. W procedurze obsługi przerwania jest odczytywany kod klawiatury z rejestru KBCODE i przesyłany do dalszego wykorzystania do procedury przerwania wygaszania pionowego.

Wyjątek stanowi klawisz BREAK, bowiem jego naciśnięcie skutkuje wywołaniem procedury przerwania BREAK wskutek innego statusu przerwania POKEY-a.

Gdy jest naciśnięty jeden z normalnych klawiszy, to naciśnięcie jest wykrywane przez POKEY dzięki temu, że stale sprawdza on matrycę, w której punktach przecięcia leżą styki poszczególnych klawiszy.

Ponadto POKEY posiada dwie linie sygnałowe, za pomocą których wykrywa naciśnięcie klawisza SHIFT i CONTROL. Samo naciśnięcie tych klawiszy nie wywołuje przerwania, zmienia jednak przy jednoczesnym naciśnięciu innego klawisza jego kod.

W systemie mogą być obsłużone maksymalnie 64 klawisze. Dają one kody od 0 do 63 (00h-3Fh). Naciśnięcie klawisza SHIFT powiększa generowany kod o 64, a więc do zakresu od 64 do 127 (40h-7Fh).

W przypadku naciśnięcia klawisza CONTROL kody powiększane są o 128, do zakresu od 128 do 191 (80h-BFh). Gdy naciśnięte są jednocześnie klawisze SHIFT i CONTROL, klawisz CONTROL ma pierwszeństwo, tak więc razem są 192 kody klawiszy.

Aby teraz przypisać tym numerom matrycy odpowiednie kody znaków ATASCII (Atari ASCII), należy skorzystać z tabeli. Początek tabeli leży w (KEYDEFPTR), a więc w położeniu początkowym od adresu KEYDEF. Tabela o wielkości 192 bajtów posiada następujące wpisy, w których uwzględnione są tylko te funkcje, które są otrzymywane bez klawiszy SHIFT i CONTROL.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00h	L	J	;	F1	F2	K	+		O		P	U	Ret	I	-	=
10h	V		C	F3	F4	B	X	Z	4		3	6	Esc	5	2	1
20h	,	Spc	.	N		M	/	/|\	R		E	Y	Tab	T	W	Q
30h	9		0	7	Bsp	8			F	H	D		Caps	G	S	A

F1 - F4 są nie sprzętowo lecz programowo istniejącymi klawiszami funkcyjnymi w 1200XL.

O tym, że jest naciniety klawisz SHIFT, mówi 3 bit w SKSTAT. Gdy jest on zero, to klawisz SHIFT jest naciśnięty. Całkowity opis rejestru SKSTAT nastąpi w dalszej części.

OBSŁUGA POTENCJOMETRÓW

Potencjometry (paddle) są urządzeniami wejściowymi, które nie dostarczają sygnału cyfrowego jak np. dżojstik, lecz sygnał analogowy, który w tym przypadku jest wartością oporności.

Ponieważ Atari może przetwarzać tylko sygnały cyfrowe, powstaje problem, jak tę wartość oporności przekształcić na liczbę.

To przekształcanie nazywa się po prostu przetwarzaniem analogowo-cyfrowym, w angielskim skrócie ADC (Analog/Digital Converting).

Przetwarzanie analogowo-cyfrowego i cyfrowo-analogowe jest dotychczas dużym kłopotem w każdym systemie komputerowym. Nie dlatego, że technicy nie wiedzą jak to zrobić. Jest obecnie kilka metod. Przez to jest znacznie trudniej znaleźć kompromis między szybkością przetwarzania i dokładnością systemu.

Stosunkowo proste jest dla elektronicznych przetworników ustalenie, czy porównywane napięcie jest większe, mniejsze czy równe. Daje to już możliwość ustalenia, gdy wykorzystamy takie układy, czy napięcie wejściowe mieści się w jednym z określonych zakresów, tzw. okien.

Gdy teraz układ ma określić, w którym z n.p, 4096 okien znajduje się napięcie wejściowe, jest to trudne i długotrwałe.

POKEY jest układem, który dozoruje i mierzy jednocześnie 8 takich kanałów. Ponieważ może on to robić kilkadziesiąt razy na sekundę, obsługuje_ niezbyt dużo możliwych okien - tylko 228. Poszczególne wartości wejścia POT (potencjometrycznego) mogą być odczytane z rejestrów POT0 - POT7.

Trzeba zauważyć, że 600XL/800XL wykorzystuje tylko 4 z ośmiu możliwych kanałów, ponieważ posiada tylko 2 gniazda dżojstików.

Przy porównywaniu listy adresowej dla POKEY-a rzuca się w oczy, że adresy AUDI...n i POTn są jednakowe. Nie stwarza to żadnego problemu, gdyż wartości AUDI...n są tylko zapisywane, a wartości POTn tylko odczytywane.

Właściwa procedura pomiaru może rozpoczynać się od dwóch możliwych rejestrów. Jednym rejestrem jest SKCNTL, w którym bit 2 ustala, czy jest to pomiar normalny czy szybki. Pomiar szybki jest mniej dokładny.

Drugi rejestr nazywa się POTGO i jest prościej rozpocząć pomiar przez niego poprzez wpisanie jakiejkolwiek danej.

Każdy kanał posiada bit w rejestrze statusu ALLPOT, z którego informacja mówi, czy wartość z odpowiedniego kanału jest "gotowa" czy nie. Gdy odpowiedni bit jest "1", to pomiar nie jest jeszcze gotowy i wartość nie jest podawana.

WEJŚCIE/WYJŚCIE SZEREGOWE

Przez szeregowe wejście/wyjście rozumie się nadawanie i przyjmowanie danych po pojedynczej linii danych. Szeregowe wejście/wyjście jest przeciwieństwem wejścia/wyjścia równoległego, w którym kompletne bajty mogą być nadawane lub odbierane przez przebiegające obok siebie linie elektryczne lub inne drogi informacyjne. Taką równoległą drogą przesyłu jest np. transmisja na drukarkę znana jako standard Centronics.

Mając do dyspozycji tylko jedną linię danych trzeba się zastanowić, jak te dane przez nią przesyłać.

Problem jest łatwy do rozwiązania, gdy trzeba tylko nadawać lub tylko przyjmować dane(simplex). Gdy trzeba jednak zarówno nadawać jak i przyjmować, powstaje pytanie, czy przesyłanie danych będzie przebiegać tylko po jednej linii (półduplex), czy też po dwóch liniach (duplex).

Gdy zostanie wybrany półduplexowy system przesyłania danych, wynika problem informowania, czy i które z przyłączonych urządzeń ma nadawać.

Przy wykorzystaniu pełnego systemu duplexowego zamiast wyżej opisanych problemów pojawia się sprawa wyższych kosztów.

Ponieważ koszty linii przy tak krótkich połączeniach jak w systemie Atari są pomijalnie małe w przeciwieństwie do przełącznika półduplexowego, firma Atari zastosowała w swoich urządzeniach przesyłanie po dwóch liniach.

Gdy to pytanie jest rozstrzygnięte, natychmiast powstaje następne: czy przesyłanie ma być synchroniczne czy asynchroniczne?

Ponieważ w Atari rzadko występują w zasadzie procesy przesyłania, zastosowano tu asynchroniczny sposób przesyłania.

Przez przesyłanie asynchroniczne rozumiemy postępowanie, przy którym odbiornik stale czeka na przesyłane znaki. Początek i koniec takiego bajta jest oznaczany przez poprzedzający znak bit startowy i następujące po bajcie półtora lub dwa bity stopu.

Zgadza się, to nie jest błąd drukarski. W przeciwieństwie do poglądu, że jeden bit jest najmniejszą jednostką informacji, przy przesyłaniu danych używa się również połówek bitów. Łatwo to zrozumieć, jak to zrealizować, gdy zobaczy się wcześniej, jak bajt z bitami startu i stopu jest przesyłany po linii.

Przy przesyłaniu wychodzi się od tego, że zawsze tylko cały bajt jest nadawany, razem z bitami startu i stopu, a poza tym zarówno odbiornik jak i nadajnik pracują z tą samą szybkością przesyłania. Ta szybkość jest sercem każdego przesyłania szeregowego. Podczas przesyłania równoległego jedna linia mówi "Hallo odbiornik, teraz idzie bajt" i odbiornik po rozpoznaniu bajta daje meldunek "Hallo nadajnik, bajt został odczytany i przetworzony".

Przy przesyłaniu szeregowym nie ma takich możliwości, więc trzeba założyć, że odbiornik odczytuje każdy bit dokładnie tak, jak został wysłany przez nadajnik.

Ten efekt uzyskuje się przez krótkotrwałą równobieżność nadajnika i odbiornika, gdy oba te urządzenia wiedzą, że każdy bit zajmuje określony czas. Ponieważ ta równobieżność istnieje tylko na długości bajta (potem następuje ponowna synchronizacja przez kolejny bit startowy!), jest to łatwo osiągalne technicznie.

W ciągu ostatnich trzydziestu lat przyjęto kilka określonych szybkości przesyłania. Wychodząc od szybkości podstawowej, w miarę rozwoju techniki podwajano szybkość przesyłania. Istnieją więc obecnie następujące ogólnie przyjęte szybkości przesyłania:

45,45	bit/sek. (tylko w USA)
50	bit/sek.
100	"
150	"
300	"
600	"
1.200	"
2.400	"
4.800	"
9.600.	"
19.200	"
38.400	"

Większe szybkości przesyłania są w złączach szeregowych bardzo rzadkie, ponieważ występują wtedy problemy techniczne.

Jest więc teraz nadajnik lub odbiornik ustawiony na określoną szybkość przesyłania i można ustalić, jak długo jeden bit zajmuje linię przesyłową.

Np. szybkość przesyłania jest 19,200 bitów/sekundę, to każdy bit na 1/19.200 sekundy, a więc około 52 milionowe sekundy. Nie jest to dużo, ale komputerowi wystarcza.

Aby teraz nadać bit stopu, łatwo jest ustawić linię na czas trwania jednego bitu w stan spoczynku. Ten stan spoczynku trwa jednak trochę dłużej, np. 1,5-krotnie dłużej niż czas trwania jednego bitu, więc jest 1,5 bitu stopu.

Po tym nieco technicznym wprowadzeniu może być pomocne narysowanie dwóch przenoszonych bajtów:

Będą przesyłane bajty 53h I 8Ah z bitami startu i stopu:

53h = 0101 0011

8Ah = 1000 1010

f = linia w stanie spoczynku /SPACE/

S = bit startu, zawsze MARK

s = bit stopu, zawsze SPACE

fffff S 0 1 0 1 0 0 1 1 s fffff S 1 0 0 0 1 0 1 0 s fffffff

Przy synchronicznym przesyłaniu danych okresy ffff mogą być dowolnej długości.

Po rozstrzygnięciu tych wszystkich spraw jesteśmy w stanie przesłać pojedynczy bajt, a więc zarówno nadać jak i odebrać.

Aby jednak nastąpiło teraz prawidłowe przesyłanie danych, jest jeszcze konieczne ustalenie tzw. Protokołu transmisji, według którego poszczególne urządzenia powinny korzystać z linii szeregowej i przesyłać pełne bloki danych.

W Atari prawie wszystkie urządzenia dołączone do linii szeregowej są inteligentne, tzn., że posiadają one układy kontroli, które przejmują komunikację z systemem nadrzędnym, a więc z Atari 600XL/800XL.

Jedynym nieinteligentnym urządzeniem jest magnetofon kasetowy. Jest on włączany i wyłączany przez użytkownika, gdy komputer nakaże to sygnałem trąbki.

Inne urządzenia jak np. drukarka lub różnego rodzaju stacje dysków stale słuchają informacji przesyłanych przez szynę szeregową, jak również przez linię zwaną !COMMAND. Gdy linia ta znajduje się na poziomie zera logicznego, oznacza to, że teraz urządzenie musi nadawać. Kiedy system nadrzędny nadaje kod rodzaju urządzenia, numer urządzenia, jak również numer bloku dla stacji dysków, aby "obudzić" właściwe urządzenie. Gdy zaadresowane urządzenie jest włączone i gotowe do wykonania tych rozkazów, to odpowiada potwierdzeniem. Nadaje ono "A" i krótką informację o statusie z sumą kontrolną. Jeżeli urządzenie względnie rozkaz nie są w porządku, odpowiedzią jest "N" (potwierdzenie negatywne).

Jeżeli urządzenie jest gotowe, rozpoczyna się przesyłanie danych, przy którym bajty do zapisania lub odczytania są nadawane w blokach, przy czym linia !COMMAND przechodzi w stan wysoki ("1").

Jeżeli wszystkie bajty są przesłane lub przynajmniej nadajnik sądzi, że są przesłane, nadaje on jeszcze bajt kontrolny, Gdy urządzeniem zewnętrznym jest n.p, stacja dysków, a rozkaz wymaga zapisania określonego bloku, to rozpoczyna on własną pracę (zapis bloku) po otrzymaniu z systemu nadrzędnego danych do zapisania. Jeżeli teraz stwierdzi, że blok do zapisania jest nieprawidłowy, to nadaje "E" (Error) jako meldunek błędu do systemu nadrzędnego, Gdy cała operacja przebiegnie poprawnie, to urządzenie nadaje "C" (Completed).

System nadrzędny czeka przez cały czas przetwarzania na negatywny lub pozytywny meldunek urządzenia. Jeżeli po kilku (normalnie 7) sekundach jeszcze go nie otrzyma, to operację przerywa z powodu przekroczenia czasu (Timeout).

W takim przypadku system operacyjny powtarza tę operację tyle razy, ile podaje wartość rejestru CRETRY (normalnie jeszcze 1 raz).

Zadaniem programu głównego jest wtedy zapewnienie prawidłowej interpretacji ewentualnych raportów błędów rejestru statusu CIO lub SIO.

Wszystko to jest znacznie prostsze w przypadku magnetofonu kasetowego. Dane są wtedy nadawane po krótkim czasie oczekiwania i zakłada się przy tym, że są już prawidłowo przekazane. Przy czytaniu jest wszakże wykonywana bardzo pomysłowa, niestety tylko bardzo pobieżna kontrola: szybkość czytania ustawia się automatycznie na przyjętą wartość. Realizuje się to przez to, że każdy blok na kasecie musi się rozpoczynać od dwóch ściśle określonych bajtów. Mają one oba wartość AAh, co binarnie daje 1010 1010 i bardzo dobrze nadaje się do synchronizacji.

Kto chciałby ulepszyć zapis na kasecie, może spróbować, czy przesyłanie będzie pewniejsze, gdy zastosowane zostaną dwa inne bajty 1100 1100. Trzeba przy tym dokonać zmian w oprogramowaniu tworzących znacznie dokładniejszy wynik pomiaru.

Poza tym jest jeszcze jedna znacząca różnica w procesie transmisji magnetofonowej w porównaniu z innymi dotychczas znanymi urządzeniami. Magnetofon nie rozumie czystych sygnałów cyfrowych. Na kasecie zapisywany jest sygnał analogowy składających się z dwóch różnych tonów. Ton niski z kanału 2 tworzy logiczne 0, a ton wysoki z kanału 1 tworzy logiczne 1. Przy przebiegu cyfrowym (który teoretycznie też może być zapisywany) magnetofony kasetowe tej klasy cenowej i przy takiej szybkości przesyłania stwarzają znaczne trudności techniczne. Nie bez powodu prawdziwe pamięci taśmowe są jeszcze dużymi, drogimi szafami, lecz w szybkości przesyłu z łatwością mogą konkurować z obecnymi systemami dysków Atari.

POKEY ma przy tym wszystkim cały szereg zadań:

Po pierwsze tworzy on takt nadawania wzgl. odbioru. Tworzony przez POKEY sygnał taktujący jest podawany przez linię CLOCK OUTPUT.

Takt nadawania może być określony przez kanał 2, kanał 4 lub doprowadzony z zewnątrz linią CLOCK INPUT.

To samo dotyczy taktu odczytywania. Nadawane dane zmieniają się każdorazowo przy rosnącym zboczu sygnału taktującego, natomiast dane odbierane są czytane przy opadającym zboczu sygnału taktującego. Zabezpiecza to, że przy czytaniu czeka się pół taktu na ustalenie sygnału.

Po drugie dla nadawanych danych POKEY posiada rejestr SEROUT, a dla odbieranych danych rejestr SERIN.

Wartość zapisana w rejestrze SEROUT jest przepisywana do rejestru równoległo-szeregowego natychmiast, gdy tylko jest on wolny. Następnie POKEY wywołuje przerwanie, w którym ogłasza, że rejestr SEROUT jest pusty i gotowy do przyjęcia nowego bajta, chociaż układ jest jeszcze zajęty nadawaniem starych danych.

Gdy żadna nowa wartość nie zostanie zapisana do rejestru SEROUT, po opróżnieniu rejestru równoległo-szeregowego następuje kolejne przerwanie, które podaje, że zostało zakończone przekazywanie ostatniego bajta.

Przy czytaniu POKEY generuje przerwanie, gdy przeczyta pełny bajt. Przekazuje wtedy bajt do rejestru SERIN i ustawia niektóre bity statusu w rejestrze SKSTAT. Może się np. zdarzyć, że komputer nie odczytał w porę danej z SERIN, a POKEY odebrał nowy bajt, wpisał go do SERIN i skasował starą informację. W takim przypadku ustawiany jest bit 6 - "przepełnienia szeregowego".

Gdy nie zgadza się coś z bitami startu i stopu, to ustawiany jest bit 7 - "błąd ramki". Ten błąd może wystąpić, gdy np. nadawany jest przez wejście szeregowe znak BREAK. Nie jest to właściwy znak, lecz mechanizm synchronizacji i resetowania, przy którym linia w przybliżeniu przez ćwierć sekundy jest w stanie niskim (a więc aktywnym - MARK).

POKEY rozpoznaje to naturalnie jako bajt 00h, zauważa jednak przy tym brak bitów stopu.

Chcąc wiedzieć dokładnie, jaki poziom ma linia nadawczo-odbiorcza, trzeba sprawdzić bit 4 rejestru SKSTAT.

Bit ten jest sprzętową kopią sygnału wyjściowego i daje oprócz (nieprogramowanej) możliwości rozpoznania BREAK także ustalanie szybkości przesyłania, bez potrzeby bezpośredniego rozpoczynania zamiany szeregowo-równoległej.

Gdy zostanie napotkany błąd, to odpowiedni bit jest ustawiany przez to, że dowolna wartość jest zapisywana w rejestrze SKRESET.

Poszczególne bity SKSTAT mają następujące znaczenie:

Bit 0: Ten bit jest niewykorzystany i jest zawsze 1.

Bit 1: Rejestr równoległo-szeregowy jeszcze pracuje, tzn., że przesyłanie danych nie jest jeszcze zakończone. Ten bit nie mówi wprawdzie, czy następny bajt danych może być wpisany do rejestru SEROUT, ale gdy nie ma żadnych dokładnych informacji o statusie wyjścia szeregowego, trzeba koniecznie czekać na unifikację tego bitu,

Bit 2: Ostatni klawisz jest jeszcze naciśnięty.

Bit 3: Naciśnięty jest klawisz SHIFT.

Bit 4: Jest to sprzętowa kopia wejścia szeregowego. Ten bit jest sprawdzany tylko do specjalnych celów jak synchronizacja.

Bit 5: Błąd przepełnienia wejścia klawiatury. Co najmniej jedno naciśnięcie klawisza zostało opuszczone.

Bit 6: Błąd przepełnienia przy przesyłaniu szeregowym. Co najmniej jeden bajt został opuszczony.

Bit 7: Błąd ramki przy przesyłaniu szeregowym.

Bity w rejestrze SKSTAT są w stanie spoczynku zawsze na "1" i mają podane znaczenie, gdy przechodzą na "0".

Bity rejestru SKCNTL mają następujące znaczenie:

Bit 0: Gdy bity 1 i 0 są zgaszone, to następuje programowy reset POKEY-a.

Bit 1: Ten bit jest wykorzystywany przy wewnętrznej obsłudze klawiatury.

Bit 2: Przez ustawienie tego bitu jest przyspieszana częstość sprawdzania potencjometrów. Pomiar przebiega wtedy nie co 20 msek, lecz co dwie linie obrazu (128µsek.).

Bit 3: Gdy ten bit jest ustawiony, to następuje wysyłanie bitów w procesie dwutonowym,

Bit 4-6: Te trzy bity służą do ustalenia szybkości przesyłania przez nadajnik i odbiornik.

Bit 7: Przerwanie wyjścia szeregowego i nadanie SPACE (poziom "1") przy ustawieniu bitu na "1". Służy do rozpoczynania nadawania i nadawania więcej niż jednego bitu stopu.

Rejestr SKCNTL ma rejestr cien: SKCNTL$

Kombinacje bitów 4, 5 i 6 rejestru SKCNTL ustalają sześć technicznie możliwych stanów:

Bit 6	Bit 5	Bit 4	Znaczenie
0	0	0	Wszystkie cykle są określane z zewnątrz. Takty wewnętrzne są ustawione na zero.
0	0	1	Szybkość nadawania jest określone przez takt zewnętrzny, zaś szybkość odczytu przez kanał 4 (lub licznik 16-bitowy utworzony z kanałów 3 i 4). Następuje asynchroniczne czytanie danych.

0	1	0	Zarówno szybkość nadawania, jak i odbioru jest określana przez kanał 4.
0	1	1	Nie wykorzystywane, ponieważ takt wyjściowy różni się przy czytaniu bajtów (synchronizacja danych asynchronicznych).
1	0	0	Szybkość nadawania jest określana przez kanał 4, zaś szybkość odbioru przez takt zewnętrzny.
1	0	1	Nie wykorzystywane.
1	1	0	Szybkość nadawania określa kanał 2, a odbioru kanał 4. Takt z kanału 4 jest wyprowadzany na zewnętrzne wyjście taktujące.
1	1	1	Szybkość nadawania określa kanał 2, a odbioru kanał 4, lecz wejście i wyjście taktujące są niewykorzystane i zamknięte.

Kombinacje 110 1 111 są wprawdzie całościowe, ale zawierają jedno ograniczenie: Ponieważ kanał 2 musi być wykorzystany dla określenia taktu, nie można zastosować przesyłania dwutonowego!

Ostatnim pozostałym do omówienia rejestrem POKEY-a jest rejestr przerwań.

Rejestr ten jest złożony z dwóch części: Rejestr tylko do odczytu IRQSTAT i rejestr tylko do zapisu IRQEN.

Gdy POKEY wysyła do CPU jedno z niżej opisanych przerwań, sprawdza w IRQSTAT, które źródło przerwania jest właściwe i odsyła do odpowiedniej procedury. Jeżeli wszystkie bity w IRQSTAT są "1", to POKEY nie jest źródłem przerwania.

Znaczenie bitów w IRQSTAT (powód przerwania):

Bit 0: Wyzerowanie timera 1.
Bit 1: Wyzerowanie timera 2.
Bit 2: Wyzerowanie timera 4.
Bit 3: Transmisja szeregowa zakończona.
Bit 4: SEROUT gotów do przyjęcia danych.
Bit 5: Wejście szeregowe odczytało dane i umieściło je w SERIN.
Bit 6: Jakiś klawisz jest naciśnięty (oprócz BREAK).
Bit 7: Naciśnięcie klawisza BREAK.

Wszystkie procedury mogą być wywołane i zabronione pojedynczo. Przerwanie przez określone wydarzenie jest wtedy dozwolone, gdy należący do przerwania bit jest ustawiony (na "1"). Bit w IRQSTAT stojący na "0" jest ponownie ustawiany na "1", gdy jego odpowiednik w IRQEN również przynajmniej chwilowo jest "0". Przyporządkowanie bitów źródłom przerwań jest jednakowe w IRQSTAT i IRQEN:

Bit 0: Zezwala na przerwanie przez wyzerowanie timera 1.
Bit 1: Zezwala na przerwanie przez wyzerowanie timera 2.
Bit 2: Zezwala na przerwanie przez wyzerowanie timera 4.
Bit 3: Zezwala na przerwanie przez koniec przesyłania.
Bit 4: Zezwala na przerwanie przez SEROUT.
Bit 5: Zezwala na przerwanie przez wejście szeregowe.
Bit 6: Zezwala na przerwanie przez dowolny klawisz.
Bit 7: Zezwala na przerwanie przez klawisz BREAK.

IRQEN ma też swój rejestr-cień IRQEN$. Gdy przerwanie jest ponownie zezwolone lub zabronione, jest to czynione nie tylko w rejestrze-cieniu, lecz również w rejestrze sprzętowym. Gdy wyłączenie nie jest zbyt pilne, może więc zostać wykonane w następnym przerwaniu synchronizacji pionowej. Trzeba wtedy tylko skorygować rejestr-cień. Korekta następuje przez OR wzgl. AND odpowiednich bitów.

PIA

PIA jest wykorzystywany w Atari głównie jako układ do obsługi wejść dżojstików. Poza tym kilka wyjść PIA steruje MMU w Atari 6OOXL/800XL.

W starych modelach Atari te wyjścia są wykorzystywane do przyłączenia trzeciego i czwartego dżojstika. Dalsze wyjścia PIA sterują portem szeregowym.

W przeciwieństwie do takich układów jak ANTIC lub GTIA PIA nie jest układem specjalizowanym, lecz układem standardowym serii 65XX lub 68XX (6520 lub 6820/6821).

Dysponuje on dwoma niezależnymi 8-bitowymi portami (z liniami PA0-7 i PB0-7), w których każdy bit można programować jako wyjście lub wejście przez rejestr kontrolny w trybie porządkowania danych. Dla każdego portu są przewidziane dwa wejścia lub wyjścia kontrolne (CA1, CA2, CB1 i CB2). Obie "połówki" PIA zasadniczo nie wpływają na siebie i są identyczne pomijając małe różnice w elektrycznych właściwościach portów (które są ważne tylko dla ich producenta) i różnicę w traktowaniu CA2 i CB2.

Przez rejestry kontroli możliwe jest wywoływanie przerwań CPU. Rejestry kontroli były pierwotnie przewidziane dla synchronizacji przesyłania danych przez porty PIA. Ponieważ są one uniwersalne, można je w Atari wykorzystywać także do innych celów. Porty PIA są wykorzystywane tylko dla dżojstików i MMU, które nie wymagają synchronizacji.

Poniższa tabela przedstawia przeznaczenie linii portów PIA:

PORT A:

PA0	**dżojstik-port 1, naprzód**
PA1	**dżojstik-port 1, wstecz**
PA2	**dżojstik-port 1, w lewo**
PA3	**dżojstik-port 1, w prawo**
PA4	**dżojstik-port 2, naprzód**
PA5	**dżojstik-port 2, wstecz**
PA6	**dżojstik-port 2, w lewo**
PA7	**dżojstik-port 2, w prawo**
CA1	**SIO Proceed**
CA2	**SIO Motor On**

PORT B:

PB0	**MMU REN (RAM w obszarze systemu operacyjnego)**
PB1	**MMU !BE, (włączanie BASIC-u)**
PB2-6	**Niewykorzystane**
PB7	**MMU !MAP (SELF TEST)**
CB1	**SIO Interrupt**
CB2	**SIO Command**

W starych modelach Atari port B obsługuje dżojstiki 3 i 4 jak port A.

PIA zajmuje w systemie co najmniej 4 bajty. To, że w Atari zajmuje znacznie więcej miejsca, wynika z niepełnego adresowania. Każda połówka PIA ma więc dwa adresy. Nazywają się one PORTA, PORTACNTL, PORTB i PORTBCNTL. PORTACNTL i PORTBCNTL sterują funkcjami PIA. Poza tym rozstrzygają one, czy zapisać przenoszone dane w PORTA i PORTB, czy też poszczególne bity portów zaprogramować jako wejście lub wyjście (rejestr porządkowania danych).

Każdy bit w jednym z rejestrów porządkowania danych reprezentuje bit któregokolwiek portu. Jeżeli odpowiedni bit rejestru porządkowania danych jest "0", to odpowiadający mu bit portu pełni rolę wejścia. Odpowiednio ten bit jest wyjściem, gdy należący do niego bit rejestru porządkowania danych jest "1".

A oto opis funkcji poszczególnych bitów rejestru PORTACNTL:

Bit 0 i 1: Sterowanie CA1.

Bit 2: gdy "0", to PORTA jest rejestrem porządkowania danych A.

Bit 3-5: Sterowanie CA2.

Bit 6: IRQA1 jeżeli podczas odczytu rejestru PORTACNTL ten bit jest "1", to znaczy, że wcześniej na CA2 został spełniony warunek przerwania.

Bit 7: IRQA2 jak bit 6, lecz sterowany przez CA1.

Sterowanie CA1 następuje przez bity 0 i 1 PORTACNTL, jak uwidoczniono w tabeli:

Bit 1	Bit 0	Opis
0	0	Przy opadającym zboczu sygnału CA1 bit 7 PORTACNTL jest "1". Wyjście przerwania A PIA pozostaje na "1".
0	1	Przy opadającym zboczu sygnału CA1 bit 7 PORTACNTL jest "1". Wyjście przerwania A PIA przechodzi na "0" i żąda przez to przerwania z CPU.
1	0	Przy narastającym zboczu sygnału CA1 bit 7 PORTACNTL jest "1". Wyjście przerwania A PIA pozostaje na "1".
1	1	Przy narastającym zboczu sygnału CA1 bit 7 PORTACNTL jest "1". Wyjście przerwania A PIA przechodzi na "0" i żąda przez to przerwania z CPU.

Sterowanie CA2:

Gdy bit 5 PORTACNTL jest "0", to CA2 służy także jako wejście przerwań. Działa przy tym według tabeli sterowania CA1. Zamiast bitów 0 i 1 funkcjami sterują bity 3 i 4 rejestru PORTACNTL. W tabeli zamiast bitu 7 musi być bit 6.

Gdy bit 5 PORTACNTL jest "1" to CA2 służy jako wyjście. Obowiązuje przy tym poniższa tabela:

Bit 4	Bit 3	Opis
0	0	Gdy bit 7 PORTACNTL jest ustawiony przez CA1, wtedy przy pierwszym ujemnym zboczu sygnału zegarowego po operacji odczytu z PORTA CA2 przechodzi na "0" i ponownie na "1".
0	1	Przy pierwszym ujemnym zboczu sygnału zegarowego po operacji odczytu z PORTA CA2 przechodzi na "0" i przy pierwszym ujemnym zboczu sygnału zegarowego, przy którym PIA nie odpowiada, ponownie przechodzi na "1".
1	0	CA2 pozostaje na "0" jak długo jest ta kombinacja bitów.
1	1	CA2 pozostaje na "1" jak długo jest ta kombinacja bitów.

Sterowanie portu B jest w konstrukcji takie same jak portu A. Można zastosować wszystkie tabele z wyjątkiem ostatniej. Trzeba jednakże zastąpić PORTACNTL przez PORTBCNTTL. Zamiast IRQA musi być IRQB, zamiast CA1 i CA2 odpowiednio CB1 1 CB2 oraz zamiast rejestru porządkowania danych A bit 2 wpływa na rejestr porządkowania danych B. Przerwania, które pochodzą z CB1 wzgl. CB2, są przekazywane CPU przez wyjście przerwań B.

Gdy bit 5 PORTBCNTL jest "1", to CB2 służy jako wyjście. Obowiązuje przy tym następująca tabela:

Bit 4	Bit 3	Opis
0	0	Przy pierwszym narastającym zboczu sygnału zegarowego po operacji zapisu do PORTB CB2 przechodzi na "0". Gdy bit 7

		jest z zewnątrz przez CB1 ustawiony, to CB2 wraca na "1".
0	1	Przy pierwszym dodatnim zboczu sygnału zegarowego po operacji zapisu do PORTB CB2 przechodzi na "0" i przy pierwszym dodatnim zboczu sygnału zegarowego, przy którym PIA nie odpowiada, ponownie przechodzi na "1".
1	0	CB2 pozostaje na "0" jak długo jest ta kombinacja bitów.
1	1	CB2 pozostaje na "1" jak długo jest ta kombinacja bitów.

Na pierwszy rzut oka możliwości sterowania PIA wydają się zapewne nieprzejrzyste. Po rozważeniu można jednak zauważyć, że pierwotnie PORT A był pomyślany jako wejście, a port B jako wyjście. Po dokładniejszym rozważeniu możliwości sterowania znaczenie różnych możliwości kanałów sterujących CA1, CA2, CB1 i CB2 będzie wyraźniejsze. Umożliwiają one komunikację PIA z innymi układami peryferyjnymi, przy której CPU w dowolnej chwili otrzymuje sygnał przerwania, gdy dane podane z odbiornika są przetworzone lub są nowe dane do wczytania przez procesor. Przy tym przerwania pozwalają się wyłączyć. CPU może pomimo to sprawdzić, czy są spełnione warunki przerwania. Po wystąpieniu przerwania procesor łatwo stwierdzi powód przerwania przez sprawdzenie bitów 6 i 7 PORTACNTL wzgl. PORTBCNTL. Można także wpływać na kanały sterujące bezpośrednio z urządzeń zewnętrznych, co jest naturalnie znacznie szybsze niż za pośrednictwem CPU. Pozwala to znacznie przyspieszyć szybkość przesyłania danych. Dzięki tym możliwościom PIA pozwala, jak wcześniej wspomniano, dokonać automatycznej synchronizacji między urządzeniami zewnętrznymi i systemem głównym.

W systemie Atari PIA nie jest wykorzystywany do przesyłania danych. Przy programowaniu należy uważać, aby nie wpływać na inne zastosowania PIA. Przy błędnym programowaniu jest np. możliwe, że port szeregowy będzie zakłócany, ponieważ jego sterowanie następuje przez PIA. Dalsze niebezpieczeństwo stwarza MMU. Przy błędnym programowaniu PIA można wyłączyć część systemu operacyjnego przez złe sterowanie MMU.

Rejestry PIA znajdują się pod następującymi adresami:

```
PORTA       54016       D300h
POTRB       54017       D301h
PORTACNTL   54018       D302h
PORTBCNTL   54019       D303h
```

Kto potrafi przygotować PIA dla innych systemów mikrokomputerowych, zdziwi się kolejnością rejestrów PIA. Normalna kolejność tych rejestrów jest następująca:

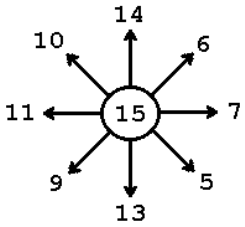
```
PCRTA
PORTACNTL
PORTB
PORTBCNTL
```

W urządzeniach Atari istnieje jednak wyżej opisana kolejność, ponieważ linie adresowe PIA są zamienione.

Aby ustalić położenie dżojstików, trzeba odczytać zawartość rejestrów PORTA i (w 400/800) PORTB. Wykorzystanie tych rejestrów jest jednak dla początkującego niełatwe. Dlatego podczas synchronizacji pionowej system operacyjny tworzy komórki pamięci, które reprezentują położenie dżojstików (rejestry te są więc rejestrami-cieniami PORTA i PORTB).

```
dżojstik 1        632   278h
dżojstik 2        633   279h
(dżojstik 3)      634   27Ah  (tylko w 400/800!)
(dżojstik 4)      635   27Bh  (tylko w 400/800!)
```

Te rejestry zawierają liczbę (dec) odzwierciedlającą położenie dżojstików przedstawione na poniższym rysunku:



S Y S T E M O P E R A C Y J N Y A T A R I

System operacyjny komputera można określić, niezależnie od wielkości, rodzaju, wieku i budowy maszyny, jako połączenie między hardwarem, a więc elektroniką i software, a więc oprogramowaniem. Jest on przede wszystkim odpowiedzialny za dostarczenie do dyspozycji użytkownika (wzgl. programisty) określonych pomocniczych środków programowych, dzięki którym można korzystać z możliwości komputera.

Poza tym system operacyjny jest takim oprogramowaniem, które po włączeniu ustawia komputer w stan wyjściowy. Zwane jest to powszechnie Power-Up. Przebiegający przy tym w systemie komputerowym proces określany jest jako Power-Up-Reset.

Podczas Power-Up-Reset są inicjowane wszystkie konieczne dla systemu operacyjnego rejestry, jak również układy wejścia/wyjścia.

Właśnie na układy wejścia/wyjścia jest w dzisiejszych systemach operacyjnych kierowana zwiększona uwaga. Jeszcze przed kilku laty normalne było wykonywanie każdego przesłania danych, np. z pamięci na dysk, bezpośrednio przez CPU, a więc przedłużało to czas pracy. Obecnie wykonują to tzw., kontrolery, które po krótkim impulsie z CPU przejmują całość przesyłania danych i sterowania urządzeń, tak że nie przeszkadza to więcej właściwemu systemowi.

Może więc kontroler być zajęty czytaniem na żądanie drugiego, innego kontrolera następnego elementu danych z pamięci i przesyłaniem tej danej do drugiego kontrolera, który może mieć wtedy zadanie zapisać ten bajt w określony sposób na dysku i sprawdzić prawidłowość tej operacji.

Takie sposoby są powszechnie zwane procesami DMA, co oznacza bezpośredni dostęp do pamięci baz korzystania z CPU.

DMA ma tę wcześniej już wspomnianą zaletę, że przy dobrze wykonanej idei DMA nie wpływa na właściwy czas obliczeń, ma jednak tę wadę, że kontroler posiadający inteligencję nie jest tani. Poza tym znacznie łatwiej dozorować system, w którym wszystkie zadania są wykonywane tylko przez jeden układ, niż taki, w którym są różne, równolegle pracujące układy inteligentne wzajemnie na siebie wpływające.

Jest jednak jeszcze trzecia możliwość zbudowania systemu. Będzie przy tym, podobnie do idei DMA, rozdzielona inteligencja. Istnieją więc tu też inteligentne urządzenia jak np. stacja dysków. Różnica w porównaniu z pracą DMA polega jednak na tym, że dysk nie "grzebie" bezpośrednio w pamięci systemu głównego, lecz tylko zawiadamia poprzez linię przerwań, że trzeba odebrać lub nadać dane. Jedynym zadaniem systemu nadrzędnego jest wtedy znaleźć, skąd pochodzi i w jaki sposób obsłużyć to przerwanie.

W normalnym przypadku takie przerwanie zawiadamia najpierw CPU. Jest tak dlatego, że CPU przerywa wykonywany program, aby wywołać procedurę przerwania. W tej procedurze przerwania, która zwykle jest częścią systemu operacyjnego, znajdują się rozkazy. Jest wtedy albo przez nie programowany układ, który wczytuje i przerabia dane leżące w przerywającym urządzeniu (wzgl. przekazuje dane żądającemu urządzeniu), albo CPU przejmuje przez nie przetwarzanie danych, aby wrócić potem do przerwanego programu użytkownika.

Prawie we wszystkich urządzeniach zasługujących obecnie na nazwę komputera spotykamy mieszane formy tych trzech wyżej podanych idei. Jest to prawdziwe zarówno dla komputerów domowych, jak i znacznie droższych komputerów biurowych.

Każdy producent ma swoje całkiem osobiste nastawienie do określonych koncepcji systemu, sympatię i antypatię do poszczególnych układów (lub ich wytwórców!), tak że nie można mówić o idei komputera.

Jak należy oczekiwać po tych wprowadzających objaśnieniach, także system Atari 600XL/800XL jest mieszaniną wszystkich trzech koncepcji.

Jako przykład czystej idei CPU może służyć sprawdzanie klawiatury. Kody klawiszy są wprawdzie czytane przez układ POKEY, lecz służy on jedynie jako układ buforowy. Właściwe dekodowanie, który kod klawisza należy do którego znaku ATASCII, jest wykonywane przez samą CPU.

Koncepcja DMA, jest, jak podano w odpowiednim rozdziale, stosowana dla ANTIC-u i GTIA w dobrze technicznie rozwinięty sposób. ANTIC służy przede wszystkim jako inteligentny kontroler DMA dla dostępu GTIA do danych.

Jest również stosowany trzeci rodzaj systemu: Atari 400/800 i 600XL/ 800XL wykorzystują w szerokim zakresie przerwania, których znaczna część służy do przekazywania danych od lub do urządzeń zewnętrznych przez złącze szeregowe i układ POKEY.

Można przy tym zasadniczo powiedzieć, że system dla komputera tej klasy cenowej posiada zdumiewający poziom, niekoniecznie dzięki podstawowej koncepcji; znajduje się podobne rozwiązania także w konkurencyjnych modelach. Ma on jednak dwie inne, niedostrzegalne bezpośrednio dla zwykłego użytkownika, zalety, które tylko wtedy są zauważane, gdy ich brakuje.

Jedną jest wygładzony, całościowy system i poszczególne składniki czysto do siebie dostrojone. Te i inne cechy zabezpieczają, że system ma możliwość "wyłapania" i skorygowania każdego dającego się wymyśleć błędu przy wykonywaniu różnorodnych czynności. Jest dość komputerów, które zawierają sprytne elementy podstawowe. Nie mają one jednak struktury kontrolującej, która wykrywa błędy.

Konkretny przykład na to znajdziemy w procedurach portu szeregowego. Może już zauważyłeś, że Atari przy zapisie lub odczycie na/z dysku czasem "zasypia", a więc przez kilka sekund nic nie robi, a potem zupełnie prawidłowo dalej pracuje. Efekt ten występuje szczególnie często wtedy, wykorzystujemy stację dysków innego producenta.

Przyczyny błędu trzeba szukać w sposobie komunikacji między komputerem i stacją dysków. Atari czeka po nadaniu rozkazów na odpowiedź nieznanego urządzenia. Odpowiedź przychodzi jednak nieco za szybko dla Atari, ponieważ jest on np. zajęty obsługą przerwania, odbiera tą odpowiedź jako błędną i czeka na prawidłowy meldunek. Ten oczywiście nie może nadejść, ponieważ stacja dysków już odpowiedziała i teraz może sama czeka na dane.

Ten konflikt, przy którym dwa jednocześnie czekające urządzenia blokują się nawzajem, jest zwany deadlock (martwy punkt) i występuje przy technikach systemu operacyjnego obawiających się zbyt kompleksowego programu i w niektórych komputerach domowych prowadzi do "załamania" systemu, ponieważ konflikt nie jest rozpoznawany. Nie dochodzi do tego jednak w systemie operacyjnym Atari. Tu tzw. timeout stara się o to, żeby przesyłanie następowało dalej. CPU dowiaduje się przez to, że widocznie przesyłanie uległo zawieszeniu i próbuje ponownie.

Drugą zaletą systemu operacyjnego Atari w przeciwieństwie do innych producentów jest to, że panuje tam porządek.

To oznacza, że cały system jest podzielony na małe procedury, w których bardzo łatwo można określić, co się otrzyma po określonym pobudzeniu.

Z małymi wyjątkami można powiedzieć, że ten system operacyjny był pisany przez inżynierów, którzy doskonale rozumieją swoją pracę.

Tak np. procedury wejścia/wyjścia nie są inne dla każdego układu, lecz są tzw. bloki kontroli wejścia/wyjścia (IOCB - Input-Output-Control-Block), które definiują rodzaj urządzenia, rozkaz i wszystkie związane z tym wartości. Wyjście odbywa się teraz przez to, że procedura wyjścia podaje tylko numer IOCB i komputer już "wie", jak ma obsłużyć dane urządzenie i którą operację wejścia wzgl. wyjścia powinien przeprowadzić.

Ten porządek jest też dlatego podstawą, że już od wejścia starych modeli 400/800 Atari nie wydawało różnych systemów, które wtedy nie pasowałyby do siebie.

Atari ma wprawdzie nową wersję, w której niektóre błędy zostały poprawione, trzeba jednak jasno powiedzieć, że oprogramowanie, które było napisane dla 400/800, w pełni pasuje do nowych systemów, o ile wykorzystuje tylko opublikowane i znane autoryzowanym producentom adresy skoków.

Jest jasne, że program zawierający wyrafinowane tricki, jest przeznaczony dla określonej wersji systemu operacyjnego. Rozważny programista nie będzie miał z tym żadnych problemów przy zmianie systemu operacyjnego Atari.

Przyczynia się do tego jeszcze, że system operacyjny w Atari jest rzeczywiście tylko systemem operacyjnym, a nie np. kombinacją z wbudowanym językiem programowania, jak BASIC. W innych systemach może się np. zdarzyć, że w interpreterze BASIC-u są zdefiniowane procedury złączy zewnętrznych. Jeżeli więc BASIC jest wyłączony, to trzeba te złącza programować na nowo.

Wszystkie te wady są w Atari 600XL/800XL maksymalnie zmniejszone. Jest wprawdzie trwale wbudowano ROM procedur matematycznych, nie jest jednak wykorzystywany przez system operacyjny. W normalnej konfiguracji systemu przy określonych operacjach wejścia/wyjścia są niespodziewanie określone miejsca w tym ROM-ie używane do odczytania wzgl. bezpośredniego skoku. Nie znaczy to jednak, że muszą tam być robione obliczenia, lecz że ten ROM może być wyłączony przez jakiekolwiek urządzenie zewnętrzne i wtedy na zwolnionym miejscu dostępna jest pamięć operacyjna. W tym obszarze adresowym mogą wtedy leżeć dodatkowe programy.

Jakie to mogą być urządzenia można tylko wyczuć, ponieważ oprócz dodatkowej pamięci zewnętrznej 64KB w 600XL nie są znane żadne urządzenia, które są dołączaną do PBI. Tylko bowiem w takim przypadku urządzenie zewnętrzne może wypełnić obszar adresowy leżący w ROM-ie matematycznym.

Obecnie już używa się stacji dyskietek Karin Maxi czy interfejsu dysku twardego KMK/JŻ IDE wersji 1 i 2, urządzeń konstruowanych przez fanów Atari, które korzystają z PBI/ECI.

Całkiem ogólnie można powiedzieć o połączeniu tego systemu operacyjnego z wiadomościami o sygnałach szyny i dekodowaniu klawiatury, że istotnie nie daje to wcale systemu operacyjnego Atari 600XL/800XL. W każdym przypadku chodzi o zmodyfikowany system operacyjny 1200XL.

Można to poznać po tym, że system operacyjny 600XL/800XL zawiera sprawdzanie klawiszy funkcyjnych 1-4, które jednak nie są przewidziane w tych urządzeniach. 1200XL posiada jednakże cztery klawisze funkcyjne F1-F4. Po tym i kilku innych szczegółach fachowiec może poznać, że ten system operacyjny nie jest własnym pakietem programowym 600XL/800XL.

W porównaniu ze starym systemem operacyjnym 400/800 niektóre rzeczy są zdumiewające i godne zapamiętania. I tak są w systemie zawarte procedury, które mają taką samą praktyczną wartość jak odpowiednie procedury w starym systemie 400/800, ale są zupełnie inaczej zbudowane i przy tym niekoniecznie lepsze, szybsze i prostsze. Są też pojedyncze, kompletne bloki programowe przesunięte tylko o kilka bajtów, chociaż można było je tam zostawić. To wszystko jest jednak nieprzyjemne tylko dla takich programistów, którzy nie trzymają się podanych przez firmę Atari wzorów.

Powstaje pytanie, czy ten otwarcie żądany przez Atari "efekt wychowawczy" jest w ogóle przez opisane kręgi osób zauważany; sądzimy, że niestety nie, ponieważ znane są już sposoby i środki skutecznego udawania przez 600XL/800XL, że jest on 400/800.

RESET I PRZERWANIA

Przy "zimnym starcie", zwanym teżPower-Up Reset, są inicjowane wszystkie konieczne do pracy rejestry i układy Atari. Są przy tym również nastawiane przerwania, które utrzymują pracę systemu po "zimnym starcie".

W Atari 600XL/800XL jest cały szereg różnorodnych źródeł przerwań. Jako pierwsze i najważniejsze trzeba wymienić przerwanie wygaszania pionowego (50Hz Vertical Blank Interrupt). Służy ono w systemie jako sygnał czasu lub, inaczej mówiąc, według niego jest ustawiany wewnętrzny zegar systemu.

Przy napotkaniu takiego przerwania, które może tylko sygnalizować, że w tym momencie nie są nadawane żadne informacje o obrazie, gdyż strumień elektronów w kineskopie wraca z dolnego prawego rogu ekranu do górnego lewego, są wykonywane niemal wszystkie przebiegające w głębi procesy, które same nie mogą wywołać przerwania. Należą do nich przede wszystkim procesy obsługi rejestrów-cieni i zegara.

Rejestry-cienie mają zadanie, przy okresowo zapisywanych przez system rejestrach, przechować zapisane informacje. Przy wielu rejestrach nie ma sensu zapisywanie przez użytkownika wartości bezpośrednio do rejestrów sprzętowych, ponieważ odpowiednie układy po krótkim czasie proszą o odświeżenie informacji. Bez rejestrów-cieni system nie wie wtedy, którą wartość powinien podać, mając jednak zapisaną żądaną informację w rejestrze-cieniu, a więc w całkiem normalnej komórce pamięci, system operacyjny może podczas przerwania wygaszania pionowego każdorazowo odczytać zawartość tej komórki RAM i przenieść ją do rejestru sprzętowego.

Wykorzystanie odczytywanych rejestrów cieni zwykle jest inne: wiele rejestrów statusu i pomiarowych uznaje odczyt pomierzonej wartości równocześnie jako rozkaz ponownego ustawienia, rozpoczyna więc nowy pomiar lub po prostu kasuje tylko określone informacje.

Mierząc wartość i czytając pomierzoną dotąd wartość nie wiedząc, czy pomiar jest zakończony, w większości wypadków otrzymujemy właśnie błędny wynik. W przeciwieństwie do tego mamy zwykłą komórkę pamięci, przy której ten problem nie występuje, bo system operacyjny już prawidłowo ją zapisuje.

Następnym efektem stosowania rejestrów-cieni jest to, że można celowo zmieniać wyniki pomiarów. Tak można po kolei sprawdzać określone bity rejestru statusu. Gdy są one ustawione, to wywołują odpowiednie działanie. Działanie to zmienia nie tylko ten jeden bit statusu, ale i kilka innych.

Po działaniu można znów ustawić "gotowe" bity w komórce RAM, co nie jest możliwe w rejestrze sprzętowym.

Zegar ma zadanie koordynować zależne od czasu działania wewnątrz systemu.

Istnieją dwa dokładnie od siebie oddzielone rodzaje zegarów. Jednym rodzajem są liczniki sprzętowe umieszczone w POKEY-u. Gdy jeden z liczników 1, 2 lub 4 jest przez takt ponownie zerowany, to może być przez POKEY wywołane przerwanie. Tym rodzajem zegarów nie będziemy się dłużej zajmować, ponieważ ich działanie jest opisane w rozdziale o POKEY-u.

Obecnie zajmiemy się tylko innymi licznikami programowymi TIMCOUNT1 - TIMCOUNT5. Są to 16-bitowe liczniki, które użytkownik ustawia na dowolną wartość w podanym zakresie, i wartość ta jest zmniejszana przy każdym przerwaniu wygaszania pionowego.

Gdy po zmniejszeniu TIMCOUNT1 lub TIMCOUNT2 osiągną wartość zero, to pośrednio przez odpowiedni wektor skoku TIMER1VKT lub TIMER2VKT jest wykonywany skok do zaprogramowanej przez użytkownik procedury przerwania zegara. Ta procedura kończy się zawsze normalnym RTS przy "uporządkowanym" stosie. To znaczy, że kiedy musi być wykonany skok powrotny, na stosie nie mogą leżeć jakiekolwiek lokalne wartości.

Gdy wartość zero osiągnie jeden z zegarów TIMCOUNT3, TIMCOUNT4 lub TIMCOUNT5, nie ma skoku do żadnej procedury, lecz tylko każdorazowo odpowiednie flagi TIMER3SIG, TIIMER4SIG lub TIMER5SIG są ustawiane z wartości 00h na FFh. Program użytkownika może wtedy sam zauważyć zmianę tej wartości. Nie zawsze jest konieczne pisanie automatycznie wywoływanej procedury dla każdego zegara, a więc dla każdego momentu, w którym powinna być wykonywana określona operacja, ponieważ jest możliwe, że odpowiedni zegar powinien kilkakrotnie zliczyć do otrzymania pożądanego wyniku.

Kolejną operacją w przerwaniu wygaszania pionowego jest wywołanie tzw., "opóźnionej" procedury przerwania wygaszania pionowego. Ponieważ ta procedura nie występuje normalnie po zimnym starcie, więc jej wskaźnik VBLKDVKT jest w procedurze przerwania wyjścia EXITVBL. Skok do procedury EXITVBL następuje też wtedy, kiedy istnieje taka opóźniona procedura przerwania i jest zakończona.

EXITVBL osiąga to, że pierwotnie przerwany program otrzymuje ponownie swoje stare wartości rejestrów.

Trzeba koniecznie wspomnieć, że TIMCOUNT1 jest obsługiwany przy każdym przerwaniu wygaszania pionowego, pozostałe zegary natomiast tylko wtedy, gdy flaga CRITICIO jest skasowana. Flaga zabezpiecza, żeby nie było zatrzymywania w pełnej pracy przy bardzo szybko następujących przerwaniach, np. przy czytaniu z dysku, z powodu takich "drobnostek" jak obsługa zegarów. Zegar 1 jest jednak wykorzystywany zawsze, gdyż jego zadaniem jest rozpoznawanie możliwej sytuacji Timeout. Naturalnie może on to robić tylko wtedy, gdy jest regularnie obsługiwany.

Obok tych regularnych przerwań są też różne przerwania asynchroniczne, a więc takie, których przypadkowe wystąpienia nie mogą być dokładnie opisane. Należy do nich np. przerwanie szeregowej linii przesyłu, które jest przez POKEY zbierane i zmieniane. Te i inne przerwania posiadają swoje wektory od adresu VPRECEDE.

Każda tu umieszczona procedura przerwania musi być zakończona instrukcjami:

```
PLA
RTI
```

aby gładko powrócić do przerwanego programu.

BLOKI KONTROLI WEJŚCIA/WYJŚCIA

W komputerze Atari wszystkie wejścia wzgl. wyjścia powinny przebiegać przez tzw. bloki kontroli (IOCB Input Output Control Block). Bloków tych jest 9, jeden na stronie zerowej i osiem pod adresami od 0340h do 03BFh. Każdy blok zajmuje 16 bajtów.

To znaczy, że nie ma dla każdego urządzenia własnego adresu, który użytkownik musi zapamiętać, lecz są adresy, które obejmują większość wejść i wyjść. Do tego opis urządzeń wejścia i wyjścia znajduje się w jednym bloku kontroli. Są tam nazwy urządzeń, numery szyn, adresy buforów i długości buforów, jak również informacje o statusie:

<table>
<tr><th>Wpis IOCB</th><th colspan="2">Opis</th></tr>
<tr><td>IOCBCHID</td><td colspan="2">IOCB Channel Identification Number
Numer identyfikacyjny kanału IOCB. Jest to część wpisu w tabeli HATABS. Istnieją już w niej następujące wpisy:
00h = "P" drukarka (printer)
03h = "C" magnetofon (cassette)
06h = "E" edytor (editor)
09h = "S" ekran (screen)
0Ch = "K" klawiatura (keyboard)
"D" stacja dysków (disk drive)
Ostatni wpis nie stoi początkowo w HATABS, ponieważ nie przebiega bezpośrednio przez CIO, lecz przez własny system operacyjny.</td></tr>
<tr><td>IOCBDSKN</td><td colspan="2">Disk Number – Numer stacji dysków. Stacje dysków są w Atari jedynym urządzeniem zewnętrznym, które może być dołączane do szyny w większych ilościach. Nie wystarcza więc informacja "D" przy przesyłaniu danych i trzeba jeszcze podać numer stacji. Numer stacji może być teoretycznie od 1 do 9, jednak stacje Atari rozpoznają tylko numery 1–4.</td></tr>
<tr><td rowspan="11">IOCBCMD</td><td colspan="2">Command. Pod tym adresem podany jest rozkaz centralnego pakietu programowego wejścia/wyjścia. Są tu następujące ważne dla wszystkich urządzeń rozkazy:</td></tr>
<tr><td>kod rozkazu</td><td>Opis</td></tr>
<tr><td>03h</td><td>OPEN. Ustala, czy urządzenie jest włączone i podłączone do szyny.</td></tr>
<tr><td>05h</td><td>GET RECORD. Czyta następny blok z urządzenia, jeśli jest ono czytelne. W przeciwnym razie zgłasza błąd.</td></tr>
<tr><td>07h</td><td>GET CHARACTER(s). Czyta następny bajt lub bajty do NEWLINE ze wskazanego urządzenia, gdy jest ono czytelne, inaczej melduje błąd.</td></tr>
<tr><td>09h</td><td>PUT RECORD. Zapisz podany blok na otwartym kanale danych.</td></tr>
<tr><td>0Bh</td><td>PUT CHARACTER/s/. Zapisz następny bajt wzgl. bajty do NEWLINE na otwartym kanale.</td></tr>
<tr><td>0Ch</td><td>CLOSE. Zamknij odpowiedni, otwarty kanał i zapisz ewentualnie znajdujące się jeszcze w buforze dane przed zamknięciem kanału.</td></tr>
<tr><td>0Dh</td><td>STATUS. Odczytaj meldunek statusu odpowiedniego urządzenia,</td></tr>
<tr><td>0Eh</td><td>SPECIAL. Wywołaj specjalną procedurę specyficzną dla urządzenia.</td></tr>
<tr><td colspan="2">Poza tym istnieją rozkazy specjalne, które obowiązują tylko niektóre urządzenia:</td></tr>
</table>

	11h	DRAW LINE. Narysuj linię od jednego punktu do drugiego w zaprogramowanym trybie graficznym.
	12h	DRAW LINE WITH RIGHT FILL. Jak 11h, tylko że w prawo od linii do ew. znajdującej się z prawej drugiej linii wypełnij zawartość ekranu.
	Obie ostatnie komendy obowiązują tylko dla ekranu, natomiast następne są tylko dla stacji dysków:	
	20h	RENAME DISK FILE.
	21h	DELETE DISK FILE.
	22h	FORMAT DISKETTE.
	23h	LOCK DISK FILE. Po tym otwarty zbiór jest tylko do odczytu.
	24h	UNLOCK FILE.
	25h	POINT. Ten rozkaz pozycjonuje głowicę stacji na wskazanym sektorze I wskazanym bajcie.
	26h	NOTE. Analogicznie do POINT. Podaje, gdzie znajduje się głowica.
IOCBSTAT	Do tego bajta jest odsyłany z CIOMAIN meldunek, po którym można rozpoznać, jak procedura CIO wykonała rozkaz. Są możliwe, zależnie od urządzenia i rozkazu, następujące meldunki statusu:	
	01h	SUCCESS. Operacja przebiegła skutecznie.
	80h	BREAK_ABORT. Podczas wykonywania rozkazu został wciśnięty klawisz BREAK.
	81h	PREVIOUS_OPEN. Próba otwarcia już otwartego kanału.
	82h	NOT_EXISTENT_DEVICE. To urządzenie jest nieznane w HATABS.
	83h	WRITE_ONLY. Próba odczytu z kanału tylko do zapisu.
	84h	INVALID_CMD. Podany rozkaz nie istnieje.
	85h	NOT_OPEN. Próba zapisu/odczytu z zamkniętego kanału.
	86h	BAD_IOCBNR. Niewłaściwy numer kanału IOCB.
	87h	READ_ONLY. Próba zapisu w kanale tylko do odczytu.
	88h	EOF_ERROR. Napotkano koniec zbioru.
	89h	TRUNCATED. Przekroczona długość wiersza.
	8Ah	TIMEOUT. Przekroczony czas przeznaczony na odpowiedź urządzenia.
	8Bh	DEVICE_NACK. Urządzenie nie gotowe lub wyłączone.
	8Ch	FRAMING_ERROR. Ten kod występuje, gdy POKEY ustali przy czytaniu, że nie wystąpił bit stopu, lecz rozpoczął się następny bajt. Ten błąd występuje przede wszystkim wtedy, gdy nadajnik i odbiornik mają inną szybkość przesyłania danych. Właściwą interpretacją tego błędu może być też warunek BREAK.
	8Dh	CURSOR_OVERRANGE. Kursor ustawiony poza ustalonym zakresem obrazu, np. przy błędnym rozkazie DRAWTO.
	8Eh	SIO-OVERRUN. Ten błąd wskazuje, że był wczytany nowy bajt przed odczytanie starego przez system. Stary bajt został przez to utracony.
	8F	SIO_CHECKSUM_ERROR. Błąd sumy kontrolnej SIO.
	90h	DEVICE_ERROR. Urządzenie nie może prawidłowo przeprowadzić operacji.

	91h	BAD_SCREEN_MODE. Ten tryb jest nieznany procedurze obsługi obrazu.
	92h	FUNCTION_NOT_IMPLEMENTED. Podana operacja nie jest wprawdzie zabroniona, lecz jednakże nie jest przewidziana.
	93h	INSUFFICIENT_SCREENMEMORY. Będąca do dyspozycji pamięć nie wystarcza, aby rozpocząć żądany tryb graficzny. Może to wystąpić, gdy do maszyny 16KB załadujemy DOS, uruchomimy Basic i wybierzemy dużą rozdzielczość grafiki.
	Ogólnie można powiedzieć o meldunkach błędów, że ich kody ustawia CPU przy odpowiednim sprawdzeniu negatywnych flag.	
IOCBBUFA	Tu jest umieszczony adres początkowy bufora danych (zwykle ustawiony do wykorzystania przez użytkownika). Wartość ta nie jest zmieniana po wykonaniu rozkazu, niezależnie od wyniku operacji.	
IOCBPUTB	Tu jest adres początkowy procedury, która przesyła bajty do zdefiniowanego urządzenia. Przy urządzeniu tylko do odczytu wektor wskazuje odpowiednią procedurę obsługi błędu.	
IOCBBUFL	Jest to długość bufora rozpoczynającego się od IOCBBUFA.	
IOCBAUXn	Te cztery bajty są rejestrami pomocniczymi, z których pierwszy – IOCBAUX1 chwilowo jest wykorzystywany do programowania CIO. Są przy tym dozwolone następujące wartości:	
IOCBAUXn	01h	APPEND. Ten kod zezwala na dodatkowy zapis do umieszczonych danych lub na odczyt z ekranu.
	02h	DIREĆTORY. Następny rozkaz OPEN powoduje dostęp do zarządzania stacją dysków.
	04h	OPEN_FOR_INPUT. Ten rozkaz może teoretycznie wystąpić przy każdym urządzeniu, mogą być jednak fizyczne przeszkody w wykonaniu (np. drukarka).
	08h	OPEN_FOR_OUTPUT. Tak samo jak OPEN_FOR_INPUT.
	10h	OPEN_FOR_OUTPUT_AND_INPUT. Tak samo jak OPEN_FOR_INPUT.
	12h	OPEN_FOR_MIXED_MODE. Dozwolone tylko dla edytora i ekranu.
	20h	OPEN_WITHOUT_CLEAR_SCREEN. Dozwolone tylko dla edytora i ekranu.

Aby teraz przekazać rozkaz do procedury CIO trzeba przesłać do rejestru X numer IOCB pomnożony przez 16 i do akumulatora bajt do nadania na JUMPTAB+06h.

Obsługa dysków nie przebiega przez procedury CIO, ponieważ występują tu inne warunki niż przy pozostałych urządzeniach. Przebiega ona przez blok kontroli dysku DCB (Disk Control Block) od adresu DSKDEVICE.

Pod tym i następnymi adresami są umieszczone następujące wartości:

DSKDEVICE	Numer rozpoznawczy szyny stacji dysków numer 1. Ta wartość jest powołaniem na następne, pozostałe urządzenia.	
DSKUNIT	Tu znajduje się numer własny odpowiedniej stacji dysków.	
DSKCMD	Można wykorzystać następujące rozkazy:	
	! 21h	FORMAT_DISKETTE
	P 50h	PUT_SECTOR_WITHOUT_VERIFY
	R 52h	GET_SECTOR

	S 53h	STATUS_REQUEST
	W 57h	PUT_SECTOR_WITH_VERIFY
DSKSTATUS	Po wykonaniu rozkazu znajduje się tu status operacji.	
DSKBUFFER	Adres początkowy bufora danych.	
DSKTIMOUT	Liczba sekund do meldunku Timeout.	
DSKBYTCNT	Długość bufora rozpoczynającego się od DSKBUFFER.	
DSKAUX1 DSKAUX2	Bajty pomocnicze. Przy większości rozkazów znajdują się tu numery zapisywanych/odczytywanych bloków.	

Skok do obsługi dysków następuje nie przez wektor CIO, lecz przez JUMPTAB+09h, a więc interfejs SIO.

P R O C E D U R Y S Y S T E M U
O P E R A C Y J N E G O

Teraz zostaną omówione poszczególne podprogramy system operacyjnego. Są one uporządkowane według adresów w ROM-ie. Wartości użyte w opisach procedur przedstawione są w kodzie hex. Cztery wartości liczbowe przed nazwą procedury oznaczają w kolejności od lewej do prawej:

Adres w 600/800XL		Adres w 400/800		Nazwa procedury
Hex	Dec	hex	dec	
C000	49152	----	-----	CHECKSROM
C001	49153	----	-----	

Te adresy zawierają sumę kontrolną wszystkich bajtów z następujących obszarów pamięci:

C002....CFFF

5000....57FF

D800....DFFF

Z tą wartością wchodzi się do CHECKROM.

C00Ch 49164 E6D5 59093 NMIENABLE

Zezwala na obsługę NMI i ustawia zawartość TRIG3 w GINTLK. W 400/800 ta procedura ma dodatkowo zadanie inicjowania układów portów.

C018h 49176 E7B4 59316 NMIFIRST

Każde NMI skacze pośrednio przez NMIVEC w to miejsce. Tu jest testowane, czy chodzi o przerwanie programu ANTIC-u. Jeżeli tak, to wtedy przerwanie programu ANTIC-u skacze pośrednio przez DLIVEC. Jeżeli nie, to po umieszczeniu na stosie zawartości akumulatora sprawdzany jest status klawisza RESET. Gdy ten klawisz jest naciśnięty, to wykonywany jest skok do wektora gorącego startu (JUMPTAB+24h). Jeżeli żadne z tych sprawdzeń nie jest skuteczne, to po odzyskaniu rejestrów X i Y ze stosu wywołuje pośrednio VBLKIVEC.

C02C 49196 E6F3 59123 JMPIRQVEC

Każde INT i każda operacja BREAK przechodzi pośrednio przez INTVEC do tego adresu. Jest tu (w przeciwieństwie do 400/800) kasowana flaga dziesiętna rejestru statusu procesora. To właśnie często jest zapominane przy tworzeniu procedur przerwania i jeszcze w 400/800 prowadzi do błędów. Ponieważ przy cyklu INT CPU 6502 automatycznie umieszczana jest na stosie zawartość rejestru statusu procesora, to po RTI flaga D ma ponownie początkową wartość. Po skasowaniu flagi wszelkie procedury przerwania skaczą pośrednio przez VMIMEDIRQ.

C030h 49200 E70B 59147 SINRDYIRQ

Ta procedura przejmuje, w połączeniu z przerwaniem, kontrolę złącza szeregowego i innych jeszcze nie istniejących. Jest przy tym sprawdzane, czy przez port szeregowy jest przesłany bajt. W takim przypadku musi być ustawiony bit 5 w IRQST wzgl. IRQST$ i następuje skok do procedury odczytu pośrednio przez VSERIN. Następnie jest sprawdzane, czy połączenie NEWIOMASK and NEWIOPORT ma wartość różną od zera, a więc czy zgłoszone jest w tym miejscu nowe urządzenie i czy generuje ono przerwanie. Jeżeli to jest ten przypadek, to wykonywany jest skok pośrednio przez NEWIOINTV.

Gdy to sprawdzenie jest bezskuteczne, sprawdzany jest IRQSTATUS w tym sensie, czy jest tam przerwanie POKEY-a, które zawiadamia system, że szeregowy rejestr jest gotowy do przyjęcia nowego bajta danych lub nawet, że przesyłanie ostatniego znaku zostało zakończone. Pierwszy warunek jest sygnalizowany przez bit 4 bajta statusu, a drugi przez bit 3. Gdy bit 4 jest ustawiony, to wykonywany jest skok pośredni do adresu zawartego w VSERREADY, przy bicie 3 skok jest do adresu zawartego w VSERCLOSE. Ponieważ te i następne sprawdzenia wykonywane są w jednym cyklu, każdorazowo znaleziony wektor (np. VSERREADY) jest ładowany do NEWIOPTR i z tego, teraz stałego adresu skacze się pośrednio do własnego odgałęzienia.

Gdy dotąd nie zostało znalezione żadne źródło przerwania, to może to być też m.in. TIMER1, TIMER2 lub TIMER4 POKEY-a, które również wywołują przerwania. Jeżeli jedno takie urządzenie znaleziono aktywne, to odpowiednie odnogi (VTIMER1, VTIMER2, VTIMER4) są wywoływane wyżej opisanym sposobem.

Jeżeli nie jest to również przerwanie zegara, to może być jeszcze żądanie klawiatury. Już w tym miejscu można wyróżnić dwie możliwości: Każdy normalny klawisz wskazuje na wektor VKEYBOARD, wspomnianym wyjątkiem jest klawisz BREAK. Gdy jest on naciśnięty i KBDISABLE jest 0 (więc zezwala), następuje pośredni skok przez VBREAKKEY. Gdy nadal nie znaleziono żadnego

źródła przerwania, są sprawdzane bity IRQ w PIA. Gdy bit 7 PORTACNTL jest 1, to wywoływany jest VPRECEDE, przy bicie 7 PORTBCNTL równym 1 wywołuje VINTERRUPT.

W końcu może to być jeszcze programowe BREAK, które jest sygnalizowane w rejestrze statusu CPU. Gdy bit BRK jest ustawiony, to jest wywoływana procedura VBREAK.

Gdy procesor nie znalazł żadnego aktywnego źródła przerwania, wtedy następuje powrót do przerwanego programu.

C092 49298 E785 59269 BRKEVENT

Ta procedura jest wywoływana, gdy został naciśnięty klawisz BREAK i klawiatura jest odblokowana (KEYDISABLE=0). Są tu kasowane flagi STARTSTOP, CURSORINH i IRQST wraz z IRQST$. System przechodzi dzięki temu ponownie do normalnego stanu pracy.

Wektor VBREAKKEY wskazuje na BRKEVENT.

C0CF 49359 ---- ----- MASKTAB

Tablica ta jest wykorzystywana do zasłaniania poszczególnych bitów, Zawiera ona w rosnącej kolejności następujące wartości (hex):

01, 02, 04, 08, 10, 20, 40, 80

C0D7 49375 ---- ----- VECTAB

Ta tablica pozostaje w bezpośrednim związku z MASKTTAB, przy obsłudze poszczególnych źródeł przerwania w SINRDYIRQ. Daje ona każdorazowo do maski odgałęzienie wektora skoku do 200h. Zawiera ona w rosnącej kolejności następujące wartości (hex):

08, CA, CC, CE, 1C, 12, 14, 3E

Przykład: Trzeba przetestować bit 1 (a więc drugi bit!).

Ładowany jest rejestr wartością 3 (X) i odpowiedni bajt statusu jest maskowany przez MASKTAB,X. Gdy wynik jest równy zero, to przerwanie jest znalezione i trzeba podać adres odgałęzienia. Ponieważ X ma jeszcze wartość 3, łatwo do NEWIOPTR załadować wartość z 200h (VECTAB,X). (VECTAB,X) daje wartość 12h, więc wektor wskazuje na adres bezwzględny 212h=200h+12h. Dokładnie to samo zachodzi dla NEUIOPTR+1 oraz 201h,(VECTAB,X). Po tym NEWIOPTR zawiera wskaźnik przerywającego urządzenia.

C0DF 49375 ---- ----- WAITFRRES

Ta część programu blokuje wszystkie przerwania i inicjuje pętlę czekania (65536 x nic), a następnie skacze do RESET.

C0F0 49392 E7D1 59345 SYSTEMVBL

Ta procedura przerwania jest wywoływana przy każdym przerwaniu wygaszania pionowego i przeprowadza większość kontroli systemu i sprawdzeń.

Jako pierwszy jest zwiększany zegar Atari TIMER, Nie jest to zegar w zwykłym sensie, lecz jedynie 3 bajty (TIMER, TIMER+1, TIMER+2) zliczające wszystkie przerwaniu wygaszania pionowego. Ponieważ w naszym systemie telewizji co 20ms jest tworzony półobraz, to strumień elektronów w kineskopie musi być ponownie przesunięty do lewego, górnego rogu ekranu, a więc Atari musi naturalnie także postarać się, by do tego czasu wygenerować nowe dane obrazu. Tak więc co każde 20ms Atari wywołuje przerwanie, po którym przeprowadzana jest procedura SYSTEMVBL.

Ten stały interwał czasowy może być także wykorzystany w zegarze z tą małą różnicą, że podaje nie sekundy, ale 1/50 sek. Przy tym TIMER+2 jest najmniej znaczącym (a więc najszybszym licznikiem, TIMER+1 jest średni a

TIMER zlicza przepełnienia TIMER1. Normalny czas można obliczyć z następującego wzoru:

CZAS = INT(((TIMER+2)+256*(TIMER+1)+256*(TIMER)))/50)

CZAS jest wtedy liczbą sekund, które upłynęły od uruchomienia komputera. Przy każdym zwiększeniu TIMER+1 jest podwyższany również ATRACT. Ten rejestr ma zadanie zmieniać kolor i jasność obrazu, kiedy przez określony czas nie został użyty żaden klawisz. Gdy ATRACT osiągnie wartość 0, (a więc po upływie 128*256/50/60 = 10,9 min. zaczyna być cyklicznie zmieniany kolor i jasność obrazu. Przebiega to w ten sposób, że zawartość ATRACTMSK jest zmieniana z FE na F6 i COLREGSH jest ładowany wartością z TIMER+1. Zmiany te mają wpływ na kolor i jasność. Przy przepisywaniu rejestrów koloru i jasności obowiązuje następująca reguła stosowania ATRACTMSK i COLREGSH:

KOL+JASN =(KOL+JASN EX-OR COLREGSH) AND ATRACTMSK

Dzięki użyciu TIMER+1 i COLREGSH automatycznie tworzy się na ekranie zmienne kolory.

Następnie są odświeżane rejestry-cienie, względnie nowe wartości do rejestrów-cieni są przesyłane z rejestrów sprzętowych. W szczególności w tym miejscu procedury są przepisywane rejestry:

LPENV	**→**	**LPENV$**	**Pionowe położenie pióra świetlnego**
LPENH	**→**	**LPENH$**	**Poziome położenie pióra świetlnego**
DLPTR$	**→**	**DLPTR**	**Display List Pointer**
DMACNTL$	**→**	**DMACNTL**	**Ustawienie parametrów DMA**
GTIACNTL$	**→**	**GTIACNTL**	**Ustawienie parametrów GTIA**

Jeżeli FINESCROL jest różny od zera, to jest zmniejszany i wartość 8-FINESCROL (modulo 8) jest wpisywana do VSCROL, aby umożliwić dalszy przesuw.

Teraz do CONSOL jest ładowane 8, aby umożliwić odczyt rejestru i przez to wykrycie naciśnięcia klawisza specjalnego.

Następnie jest przeprowadzane odświeżanie i ewentualnie wyżej opisana zmiana rejestrów koloru i jasności. Rejestry COLPMO$ - COLBAK$ są kopiowane do rejestrów COLPMO - COLBAK, przy czym każda wartość jest modyfikowana według wyżej podanego równania.

Dalej przepisywane są rejestry CHARBASE$ i CHARCNTL$ do rejestrów CHARBASE i CHARCNTL. Pierwsza wartość dostarcza ANTIC-owi wyższy bajt adresu aktualnego generatora znaków, których dwa są wbudowane w Atari 600XL/800XL, a które przez ten rejestr można rozszerzyć. CHARCNTL określa, czy znaki będą tworzone normalnie, czy odwrotnie.

Następny jest blok testowania i modyfikacji zegarów TIMCOUNT1 - TIMCOUNT5.

Każdy z tych 5 zegarów jest 16-bitowym licznikiem, który jest zmniejszany co 20ms przez procedurę DECTIMER. Zegarom TIMCOUNT1 i TIMCOUNT2 są przyporządkowane wektory skoku TIMERIVEC i TIMER2VEC, które są wykorzystywane, gdy po zmniejszeniu licznik osiągnie stan zero. Dla trzech pozostałych liczników TIMCOUNT3, TIMCOUNT4 i TIMCOUNT5 są po osiągnięciu zera ustawiane flagi TIMER3SIG - TIMER5SIG.

Przebieg czasowy jest taki, że najpierw jest zmniejszany TMCOUNT1 i, w razie potrzeby, wywoływany TIMER1VEC. Potem następuje niżej opisane sprawdzenie, ale tylko wtedy, gdy w tym miejscu programu CRITICIO ma wartość zero, a więc jest skasowany.

Jeżeli jest to ten przypadek, to TIMCOUNT2 jest zmniejszany. Przy wystąpieniu wartości zero jest wzywana przez JMPTIMER2 i pośrednio przez TIMER2VEC odpowiednia procedura. Po tym są obsługiwane zegary 3 - 5.

Następnym blokiem jest obsługa klawiatury. Daje ona użytkownikowi przez kilka rejestrów różnorodne możliwości blokowania, zmiany kodu i maskowania poszczególnych klawiszy.

Właściwa procedura czytania najpierw testuje, czy jest naciśnięty klawisz. Gdy żaden nie jest naciśnięty, to sprawdza, czy licznik KEYDELAY jest zero. Jeśli nie, to jest on zmniejszany, w przeciwnym wypadku kontynuowana jest obsługa dżojstików.

Zmienna KEYDELAY zabezpiecza przed zbyt szybkim powtórzeniem naciśniętego klawisza, ponieważ system akceptuje nowe podanie klawisza, gdy ostatnio naciśnięty klawisz "wydźwięczy". Jest ona inicjowana wartością 3.

Kiedy jest naciśnięty nowy klawisz, to licznik SRTIMER, inicjowany przez KEYDELAY (wartość standardowa 40), rozpoczyna liczenie wstecz, gdy ten klawisz nadal pozostaje wciśnięty. Jeżeli przy przerwaniu osiągnie on wartość zero, a klawisz jest nadal wciśnięty i KBDISABLE=0, tzn. zezwala, to rozpoczyna się automatyczne powtarzanie klawiatury. Aby ustalić szybkość powtarzania, wykorzystuje się SRTIMER. Tak długo, jak ten klawisz pozostaje wciśnięty, SRTIMER jest ładowany wartością z KEYREP i przy każdym przerwaniu wygaszania pionowego zliczany wstecz do zera. Gdy osiągnie zero, to jest z rejestru KBCODE wzgl. KBCODE$ odczytywany kod klawisza.

KEYREP określa więc szybkość powtarzania i jest inicjowany wartością 5, podczas gdy wartość z SRTIMER leżąca w KEYDELAY określa czas do rozpoczęcia potarzania.

Ponieważ ten system operacyjny, jak już wspomniano, jest systemem operacyjnym również dla 1200XL, teraz wykonuje się kilka działań bezsensownych dla 600XL/800XL:

Odczytany kod klawisza jest ponownie sprawdzany na kilka określonych znaków specjalnych. 1200XL dysponuje czterema programowalnymi klawiszami funkcyjnymi F1 – F4. Dziwnym zbiegiem okoliczności Atari sprawdza tu kod klawisza na CNTL i F1, CNTL i F2, CNTL i F4, CNTL i 1 jak też widoczne w matrycy klawiatury nie istniejące kody w trybie normalnym, z SHIFT i CONTROL. Gdy odczytany kod klawisza nie odpowiada żadnej z dwunastu wartości, to kod ten umieszczany jest w KBCODE$, inaczej nie. Interesujące jest, że w tym miejscu nie jest sprawdzany.

Po sprawdzeniu klawiatury Atari sprawdza ponownie porty dżojstików.

Procedura ta rozpoczyna się od adresu C201h i może być wywołana, gdy wszystkie poprzednie sprawdzenia nie były potrzebne. Należy jednak zauważyć, że normalne przerwanie wygaszania pionowego korzysta z tej procedury. Obowiązuje to również przy sprawdzaniu triggera i paddle'ów.

Ponieważ 600XL/800XL dysponuje tylko dwoma portami dżojstików, a nieco programów używa portów 3 i 4, Atari symuluje brakujące porty dla zachowania kompatybilności.

Najpierw są czytane 4 górne bity portu A PIA i równocześnie użytkowane jako wartości dla JOYSTICK1 I JOYSTICK3; w 400/800 wartość dla JCYSTICK3 jest pobierana z PORTB. Następnie cztery dolne bity z PORTA są przesyłane do JOYSTICK0 i JOYSTICK2.

Teraz (od C219h) są sprawdzane wejścia przycisków. Tu także dane z portu 1 i 2 są wykorzystywane dla portów 3 i 4 w ten sposób, że są czytane z TRIGGER0 i TRIGGER1 i zapisywane do TRIGGER0$ i TRIGGER2$ wzgl. do TRIGGER1$ i TRIGGER3$.

Podobnie postępuje rozpoczynające się od C22Bh czytanie wejść paddle'ów. Rejestry POKEY-a PADDLE0 – PADDLE3 są kopiowane na PADDLE0$ – PADDLE3$ wzgl. PADDLE4$ – PADDLE7$ i czytany jest przycisk paddle.

Obowiązuje przy tym następujący porządek:

z JOYSTICK0 bit 2 ==: PTRIG0, PTRIG4

JOYSTICK0 bit 3 ==: PTRIG1, PTRIG5

JOYSTICK1 bit 2 ==: PTRIG2, PTRIG6

JOYSTICK1 bit 3 ==: PTRIG3, PTRIG7

W tym miejscu pierwsza procedura przerwania wygaszania pionowego jest zamykana i następuje pośredni skok przez VBLKDVEC do tzw. "opóźnionej"

procedury przerwania wygaszania pionowego. Jest ona wolna do wykorzystania przez użytkownika. Należy uważać przy wykorzystywaniu tego wektora, aby procedura przerwania nie była zbyt długa, gdyż w innym razie właściwy proces (np. BASIC) nie zostanie uruchomiony.

C25D 49757 E8EF 59631 JMPTIMER1

Przeprowadza skok pośredni z VTIMER1.

C26O 49760 E8F2 59634 JMPTIMER2

Przeprowadza skok pośredni z VTIMER2.

C263 49763 E8F5 59637 DECTIMER

Procedura DECTIMER zmniejsza podany zegar, gdy jest on różny od zera. Gdy odnośny zegar jest albo zero przed zmniejszeniem, albo różny od zera po zmniejszeniu, to DECTIMER umieszcza w akumulatorze wartość FF, w przeciwnym wypadku 00. Przez ładowanie wywołany program może wykorzystać wartość flag Zero. Gdy zegar jest ustawiony na zero, to Z=1, inaczej Z=0.

DECTIMER otrzymuje numer obsługiwanego zegara, pomnożony przez 2, przy wywołaniu z rejestru X; a więc np. wartość 06h dla TIMCOUNT3.

C280 49792 E912 59666 SETVBLVEC

Ta procedura jest powszechnie wykorzystywana do inicjowania wektorów przerwań lub wartości zegarów. Do tego muszą być podane do SETVBLVEC następujące wartości:

Wartość	Rejestr
Wyższy bajt zapamiętanego adresu lub zapamiętanej wartości	X
Niższy bajt zapamiętanego adresu lub zapamiętanej wartości	Y
Numer rejestru	A

Numer wektora może być:

0 ==: VIMMEDIRQ
1 ==: TIMCOUNT1
2 ==: TIMCOUNT2
3 ==: TIMCOUNT3
4 ==: TIMCOUNT4
5 ==: TIMCOUNT5
6 ==: VBLKIVEC
7 ==: VBLKDVEC

Numery wektorów mogą być też większe niż 7, należy jednak wtedy pamiętać, że procedura może umieszczać tylko 16-bitowe wartości i tylko pod prostymi adresami.

Przed modyfikacją wektorów czeka się na przerwanie wygaszania pionowego, ponieważ wektory mogłyby być połowicznie zmienione, gdy zostaną "przyłapane" przez przerwanie.

C298 49816 E93E 59710 EXITVBL

Ta procedura umieszcza tylko wartości ze stosu w rejestrach Y i X oraz w akumulatorze (w tej kolejności) i kończy przez RTI to (dowolne) przerwanie.

C29E 49822 F11B 61723 RESETWARM

RESETWARM może być wykonana bezpośrednio lub przez JUMPTAB+24. Na początku procedury gorącego startu jest sprawdzane, czy nie oddziaływają na system różnorodne przecież obciążające wpływy, które koniecznie wymagają zimnego startu z zainicjowaniem wszystkich rejestrów. Do tych wpływów należą np. wyjęcie lub ponowne włożenie kartridża w szczelinę. Ponieważ w 600XL/800XL, w przeciwieństwie do 400/800 nie jest komputer wyłączany automatycznie przy zmianie kartridża, a więc ten test jest konieczny.

Do tego jest czytana zawartość TRIGGER3 GTIA i porównywana z zawartością urządzonego do tego celu ROMFLAG. Jaki cel ma czytanie TRIGGER3, zdradza nam spojrzenie do otwartego urządzenia: linia RD5, która przy włożeniu kartridża wyłącza ewentualnie znajdującą się w tym samym miejscu RAM, jest połączona właśnie z tą linią przycisku (triggera), która jest wolna dzięki zwolnieniu portów dżojstików 3 i 4. Kiedy teraz od ostatniego resetu kartridż jest albo włożony, albo wyjęty, TRIGGER3 i ROMFLAG nie są zgodne i Atari forsuje wywołanie RESETCOLD.

Gdy TRIGGER3 i ROMFLAG są identyczne, to jest możliwe, że wprawdzie kartridż jest w gnieździe, jednak jest on inny niż przy ostatnim resecie. Dla tego przypadku są sprawdzane ostatnie adresy ROM-u, dokładniej przez wywołanie NEWCART jest tworzona suma kontrolna od BFF0 do C0EF i porównywana z sumą kontrolną ostatniego wywołania. Jeżeli suma kontrolna nie jest są porządku, to jest przeprowadzany zimny start. Jest jeszcze trzecia możliwość, przy której można przejść od gorącego startu do zimnego.

Przez ustawienie COLDSTART na wartość różną od zera. Jeżeli nie ma żadnego znanego źródła forsującego zimny start, to jest wywoływany RESET+02h z wartością akumulatora FF, tzn. też jest przeprowadzana właściwa procedura gorącego startu.

C2B8 49848 F125 E1733 RESETCOLD

Skok do tej procedury może być wykonany bezpośrednio lub przez JUMPTAB+27. Droga przez JUMPTAB jest zastosowana dla zapewnienia kompatybilności programów z modelami 400/800.

Na początku zimnego startu, analogicznie do RESETWARM, jest sprawdzane, czy rzeczywiście o to chodzi lub czy nie oddziaływały od ostatniego resetu żadne wpływy które mogłyby skierować system do gorącego startu. Do tego najpierw jest sprawdzane, czy trzy adresy, które przy zimnym starcie są ustawione na określone wartości, jeszcze takie zawierają.

Te adresy i ich zawartości są następujące:

POWUPBYT1 musi przy gorącym starcie zawierać 5C

POWUPBYT2 musi przy gorącym starcie zawierać 93

POWUPBYT3 musi przy gorącym starcie zawierać 25h

Właściwie jest niejasne, co oznaczają te wartości, wynika to jednak z tego, że nowo dołączony układ pamięci RAM określa nie te trzy wartości.

Normalnie pamięci dynamiczne po włączeniu zasilania mają zawartość 00 lub FF zależnie od ich fizycznej budowy.

Kiedy jeden z tych trzech adresów nie zawiera wartości gorącego startu, to WARMFLAG jest ładowany przez 00, co informuje po tym procedurę RESET, że chodzi o zimny start.

Gdy te trzy adresy są w porządku, to jest wywoływana RESETWARM.

C2D6 49878 F126 61734 RESET
C2D8

W tej części programu jest inicjowane mniej lub więcej rejestrów i układów, zależnie od tego, czy chodzi o start zimny czy gorący. Właściwą decyzję o tym, czy chodzi o start zimny czy gorący, podejmuje się nie tu, lecz w RESETWARM wzgl. RESETCOLD. Gdy jest wywoływany RESET, to WARMFLAG jest ustawiany na 0C, co sygnalizuje dalszej procedurze zimny start. Gdy jest wywoływany RESET+02, to w WARMFLAG jest umieszczana wartość przekazana z akumulatora. Jeśli ta wartość = FF, to WARMSTART sygnalizuje gorący start.

Po tym rozpoczynają się operacje resetowania. Najpierw DOSVEC jest ustawiany na wartość TESTROMEN. Jeżeli przy włączeniu komputera lub innym starcie zimnym jest naciśnięty klawisz OPTION, to znajdujący się w 600XL/800XL BASIC jest wyłączany. Gdy stacja dysków nie jest dołączona lub

tylko nie odpowiada prawidłowo, Atari próbuje po pewnym czasie, uczynić coś innego niż odejście. Ponieważ jednak nie ma do dyspozycji BASIC-u i nie posiada monitora, to aby nie runąć, skacze przez DOSVEC. Dzięki temu wektorowi jest wtedy włączany wbudowany w ROM Self Test.

Zanim jednak można to wszystko uczynić, trzeba jeszcze zrobić cały szereg innych rzeczy.

Najpierw jest sprawdzane przez CARTGO, czy kartridż jest w gnieździe. Jeśli tak i jeśli ten ROM jest działający (zob. CARTGO), to procedura nie wraca, lecz skacze do tego ROM-u pośrednio przez BFFE.

Jako następna jest tworzona przez NEWCART suma kontrolna ostatnich bajtów ewentualnego ROM-u i umieszczana w ROMFLAG.

Dalej jest tu sprawdzane, czy BASIC jest włączony i porty zainicjowane. Potem Atari rozpoczyna test RAM, jeśli chodzi o zimny start. Przy tym są ustawiane odpowiednie zmienne z SYSINIT.

Gdy to się zdarzy lub gdy chodzi o start gorący, wtedy są inicjowane zmienne z zakresu C200-03ED i 0000-007F. Teraz X64KBFLAG jest ustawiany na wartość bitu 1 PORTB i zmienne POWUPBYT1, POWUPBYT2 i POWUPBYT3 są ładowane wartościami 5C, 93 i 25. Dalej lewa i prawa krawędź okna tekstowego są ustalane w RIGMARGIN i LFTMARGIN na wartości 39 wzgl. 2.

Następny fragment procedury dotyczy 600XL/800XL, bowiem w 400/800 nie jest stosowany. Jak wiadomo, w USA stosowany jest system telewizyjny NTSC, w Europie PAL. Aby ustalić w jakim systemie pracuje Atari, jest czytany Rejestr GTIA NTSCPAL. Gdy bity 1 - 3 tego rejestru są równe zero, wtedy chodzi o PAL, inaczej o NTSC. Każdy system ma teraz różnie ładowane następujące rejestry:

Rejestr	NTSC	PAL	Opis
NTSCPAL$	00	01	Flagi do dalszego wykorzystania
KEYRPDELY	30	28	Wartość inicjująca dla SRTIMER, kiedy klawisz jest nowo naciśnięty.
KEYREP	06	05	Wartość inicjująca dla SRTIMER, kiedy klawisz jest naciśnięty, a SRTIMER przepełnił się przynajmniej raz.

Aby uruchomić całą maszynerię przerwań, zegarów i wektorów trzeba zainicjować adresy 0200 - 0225 (DLIVEC - VBLRDVEC) Wartościami, które znajdują się od adresu INIT200. Bezpośrednio po tym inicjowane są wektory obsługi od HATABS adresami podanymi od INIT31A.

Z procedury testowania pamięci jest do wykorzystania jeszcze jedna flaga: adres 0001 z zawartością 00 lub inny dla wyniku testowania pamięci są wykorzystywane jako rejestry pomocnicze.

Jeżeli ten wynik nie jest zero, to przez GOMEMTEST jest włączany test ROM-u, który inicjuje zmienne CHARCNTL wartością 02 i CHARBASE$ wartością E0 (dla generatora znaków od E000) i bezpośrednio wywołuje test pamięci.

Jeżeli pod adresem 0001 jest zero, to są uzbrajane IOCB0. Programuje to rozkaz OPEN, adres buforu OPENSCST, jak również IOCBAUX1 na READ+WRITE (zob. objaśnienie IOCB). Słowami znaczy to, że IOCB0 jest przygotowany do operacji zapisu/odczytu edytora obrazu. Następnie jest próbowane wykonanie tego OPEN. Ponieważ nie występują przy tym właściwie żadne problemy, to przy ewentualnym błędzie wykonywany jest RESETCOLD bezpośrednio bez drugiej próby.

Jeżeli wszystko dotąd przebiegło gładko, jest sprawdzane, czy jest transmisja kasetowa lub czy istnieje działająca stacja dysków. Jeśli tak, to jest ładowane przez BOOT, a ponadto gdy TMPRAMSIT = 01 i bit 2 z adresu BFFD jest ustawiony, to skacze pośrednio przez COLDCART. Jeżeli nie jest to

ten przypadek, to ewentualnie zmieniony w międzyczasie wektor DOSVEC jest wykorzystywany pośrednio jako wektor skoku.

C3BD 50109 ---- ----- GOMEMTEST

Tu bit 0 PORTB jest ustawiony na 0. Przez to znajdująca się w obszarze 5000-57FF RAM jest wyłączana i leżący w obszarze D000-D7FF za układami I/O test ROM jest kopiowany przez MMU na obszar 50OO-57FF. To pozornie trudne kopiowanie jest prosto wykonywane przez to, że MMU w adresie binarnym y101 xxxx xxxx xxxx (x=dowolne) ustawia bit y na 1. Innymi słowami MMU dodaje do adresu 5xxx wartość 8000 i otrzymuje Dxxx.

Tu jest więc teraz test ROM, który zawiera dwa początki: adresy TESTROM i TESTROM+03. Przez TESTROM dochodzi się do "menu", które można również otrzymać przy zimnym starcie z naciśniętym klawiszem OPTION. TESTROM+03 prowadzi bezpośrednio do testu pamięci.

Po przełączeniu następuje więc skok do TESTROM+03h.

C431 50225 ---- ----- COLDCARTC

Przeprowadza pośredni skok do (COLDCART).

C434 50228 ---- ----- DOSVECC

Przeprowadza pośredni skok do (DOSVEC).

C437 50231 ---- ----- INITCARTC

Przeprowadza pośredni skok do /INITCART/,

C43A 50234 ---- ----- CLCANDRTS

Jak wskazuje nazwa tej procedury, jest tu tylko kasowana flaga Carry i następuje powrót (Return). Ta z pozoru bezsensowna procedura może być łatwo wykorzystana w programach użytkowych do sygnalizowania meldunków OK przy wyjściu procedury. '

C43C 50236 F0E3 61667 INIT31A

Ta tablica zawiera wartości, którymi jest inicjowana tabela skoków od HATAB$.

C44B 50251 F10D 61709 DERRMSG

Tu znajduje się meldunek "BOOT ERROR (CR)".

C459 50265 E480 58496 INIT200

W tym miejscu znajdują się dane, które są ładowane przez RESET do wektorów przerwań i zegarów, wzgl. do samych zegarów.

C47F 50303 ---- ----- CARTGO

Ta procedura sprawdza, czy jest włożony kartridż. Kiedy jest spełniony ten warunek i adres znajdujący się w ROM na BFFC i BFFD jest większy lub równy 8000, przy czym musi to być adres x000, to następuje pośredni skok przez BFFE.

Kiedy to nie jest ten przypadek, to są inicjowane różnorodne porty przez IOPORTINI i sprawdza się, co dzieje się z bitem 1 PORTB. Ten bit jest wyjściem i podaje, czy może być aktywny wbudowany BASIC. BASIC jest włączony, gdy bit 1 PORTB ma wartość zero. Ma on tę wartość, gdy X64KBFLAG jest zero i nie jest naciśnięty klawisz OPTION. To ostatnie wskazuje bit 2 CONSOL. Gdy jest 1, to klawisz nie jest naciśnięty i odwrotnie. Po ustawieniu tych bitów procedura kończy się. Ponieważ jednak nie jest to

samodzielna procedura, przechodzi do następnej procedury GETRAMHI i dopiero stamtąd następuje powrót.

C4B7 50359 F258 62040 GETRAMHI

Ta procedura umieszcza w rejestrze TMPRAMSIZ liczbę, będącą do dyspozycji użytkownika, pełnych 256-bajtowych bloków RAM (pamięć operacyjna). Są przy tym czytane pierwsze bajty takich bloków i dokładnie na te same miejsca są zapisywane uzupełnienia do 1 tych wartości. Jeżeli po tym zawartość komórek pamięci nie zgadza się z zapamiętanymi uzupełnieniami, to widocznie nie chodzi tu o komórki RAM i program kończy się. Gdy obie wartości zgadzają się, to z powrotem są zapisywane oryginalne wartości.

Jeżeli komórki pamięci pozwalają się przeprogramowywać, to chodzi tu o komórki RAM, inaczej program jest przerywany. Ponieważ przy testowaniu adresy TMPRAMSIZ+01 i TMPRAMSIZ służą jako wskaźnik testowanej komórki pamięci, więc TMPRAMSIZ jako wyższą wartość wskaźnika zawiera starszy bajt pierwszego adresu nie-RAM i przez to liczbę będących do dyspozycji, niżej leżących bloków.

C4D7 50391 ---- ----- NEWCART

Jest tu tworzona suma kontrolna wszystkich bajtów od BFF0 do C0EF i porównywana z CARTCKSUM. Gdy obie wartości są równe, CPU wraca z powrotem z ustawioną flagą Zero, w przeciwnym wypadku umieszcza nową wartość w CARTCKSUM i przed powrotem kasuje flagę Zero.

C4E8 50406 F281 62081 IOPORTINI

W tej części programu są inicjowane wszystkie znane porty i układy I/O, a dokładniej obszary:

D000-D0FF (GTIA)
D200-D2FF (POKEY)
D300 (PORT)
D302-D3FF (PORT)
D400-D4FFh (ANTIC)

są w całości ustawiane na 00. Następnie PORTB jest przełączany na wyjście i wszystkie jego bity są ustawiane na 1. Po tym są czytane słowa statusu z PORTA i PORTB, aby wykryć ewentualne przerwanie.

Gdy porty są zainicjowane, to kanał 4 POKEY-a jest ustawiany jako takt nadawania lub odbioru. Następnie rejestry AUDCNTL3 i AUDCNTL4 są ustawiane na A0, a AUDIOCNTL na 28. Na zakończenie bajt FF jest posyłany do szeregowego rejestru nadawania.

C543 50499 F294 E2100 SYSINIT

Ta wywoływana przez RESET procedura ma głównie zadanie inicjowania wszystkich układów I/O. Do tego najpierw kasuje ona bity zezwolenia BREAK w IRQST$ oraz w BRKEVENT. Po tym RAMSIZE jest ładowany wartością z TMPRAMSIZ i MEMTOP jest również odpowiednio ustawiany, aby potem umieścić wartość 700 w MEMLO jako dolną granicę RAM użytkownika.

Następnie są wywoływane w tej kolejności procedury inicjowania edytora, obrazu, klawiatury, drukarki i magnetofonu przez odpowiednie wektory od EDITORVEC.

Brakuje tu jeszcze pierwszych wywołań CIOINIT, SIOINIT i NMIENABLE, aby uzupełnić strukturę przerwań systemu, jak również wywołania DISKINIT.

Gdy CPU wraca po tej procedurze, to ustawia NEWIOINIT na NEWIOREQ i na końcu wywołuje NEWINIT przez NEWINITC.

Po tych obsługach sprzętowych przed powrotem z procedury jest jeszcze STARTTST ustawiany na 01, gdy naciśnięty jest klawisz START, w przeciwnym razie na OO. Ta wartość jest później wykorzystywana w BOOT.

C599 50585 F2CF E2159 BOOT

Na początku tej procedury jest sprawdzane, czy chodzi o skok Boot przy gorącym starcie. Jeżeli jest to ten przypadek (WARMLAG różne od 00) i poza tym jest załadowany rozsądny DOS, a więc DOSACTIV=1, to wykonywany jest pośredni skok do DOSINITV.

Jeżeli jest to wprawdzie gorący start, lecz nie jest załadowany DOS, to jest po prostu przerywane wykonywanie procedury.

Ostatni przypadek jest w tej procedurze interesujący:

Najpierw próbuje otrzymać meldunek statusu ze stacji dysków numer 1. Dzieje się to po ustawieniu STATUSCMD w DSKCMD, wstawieniu numeru 1 w DSKUNIT i wywołaniu DISKINTERF przez JUMPTAB+03. Gdy procedura wraca ze skasowanym N w rejestrze statusu CPU, to status jest w porządku i można ładować z dysku. Jeżeli nie, to BOOT wraca z ustawionym bite N.

Gdy sprawdzenie statusu przebiegło pozytywnie, to jest ładowany sektor numer 1 od adresu 0400. W tym celu numer bloku 0001 jest ładowany do DSKAUX1 (młodszy bajt) i DSAUX2 (starszy bajt) oraz wartość 0400 do DSKBUFFER, aby potem odczytać blok przez GETSECTOR. GETSECTOR może czytać zarówno z kasety, jak i z dysku i w wypadku błędnego lub nieodczytanego bloku przesyła wartość negatywną.

W takim przypadku jest podawany meldunek "BOOT ERROR" (błąd ładowania) i przy ładowaniu z kasety jest procedura przerywana, w innych wypadkach następuje ponowna próba odczytu.

Przed opisem dalszych działań wskazane jest wyjaśnienie dwóch pokrewnych pojęć:

SEKTOR jest to zapisany na dysku zestaw danych. W Atari jest na dysku 40d (decimal) koncentrycznych ścieżek, z których każda zawiera 18d sektorów po 128d bajtów danych. Ścieżki te są ponumerowane od 00h do 39h, a sektory od 01h do 12h. Dlaczego zastosowano różne sposoby liczenia, jest niejasne, ale są one ogólnie przyjęte.

W przeciwieństwie do fizycznego sektora BLOK jest logicznym zbiorem danych. Teoretycznie może on zawierać dowolną ilość bajtów danych. Dla uproszczenia blok w Atari zawiera tyle bajtów danych, co sektor. Na dyskietce o pojedynczej gęstości bloki są ponumerowane od 01 do 02D0.

Pomimo to zbiór powinien być trochę objaśniony, ponieważ przy korzystaniu z rozszerzeń Atari (np. systemy CP/M) to oddzielenie może być bardzo interesujące. Przy wykorzystaniu stacji dysków z podwójną gęstością zapisu blok zawiera tylko połowę danych mieszczących się w sektorze, tak że system operacyjny wykonuje tu ładowanie poszczególnych danych. Ale o tym później.

Gdy pierwszy blok jest prawidłowo odczytany, cztery pierwsze bajty są kopiowane na DSKFLAG, DSKSECCNT i DSKLDADR. Określają one długość sektora, liczbę bloków do odczytania i adres początkowy, od którego odczytane dane powinny zostać umieszczone w pamięci. Następne dwa bajty określają adres początkowy zbioru i są umieszczane w DOSINITV.

C5C9 50633 F301 62209 BLOCK1

(Ta etykieta stoi tu, w środku procedur, dlatego, że wskakuje się tu z CASBOOT!)

Po tym cały odczytany blok jest umieszczany od podanego adresu, zmniejsza się DSKSECCNT i, jeśli jeszcze jest blok do odczytu, to jest on umieszczany na następnych adresach. Po wystąpieniu błędu odczytu przy

czytaniu z kasety praca jest przerywana, zaś przy czytaniu z dysku podejmowana jest próba kolejnego odczytu całego zbioru.

Jeżeli wszystkie bloki są prawidłowo odczytane i były ładowane z kasety, to jest czytany jeszcze blok END OF FILE, który jednak nie jest dalej wykorzystywany.

W przeciwnym razie jest wywoływany przez BLOAD początek procedury odczytu. Ten początek leży w szóstym bajcie pierwszego odczytanego bloku. Przy programie dyskowym można tu sprawdzić, czy rzeczywiście wszystko jest prawidłowo załadowane. Kiedy wszystko jest w porządku, musi ten test powrócić ze skasowaną flagą Carry, dzięki czemu nie próbuje się jeszcze raz ładować.

Gdy ładowanie przebiegło skutecznie, to jest przez DOSINITC inicjowany system operacyjny dysku i potem DOSAKTIV jest ustawiany na 1.

C637 50743 F36C 62316 BLOAD

Wartość z DSKLDADR+6 jest ładowana do RAMTSTPTR i potem pośrednio, przez RAMTSTPTR skacze wczytana procedura.

C649 50761 F37E 62334 DOSINITC

Przeprowadza pośredni skok przez DOSINITV.

C64C 50764 F381 62337 DSKRDERR

Rejestry CPU X i Y są ładowane niższym wzgl. wyższym adresem meldunku błędu DERR. Ta część programu sama nie wraca, lecz przechodzi do następnej procedury.

C650 50768 F385 62341 PUTLINE

Ta procedura pobiera z rejestru X CPU młodszy bajt, a z rejestru Y starszy bajt adresu, od którego znajduje się w pamięci tekst do wydrukowania. Wiersz tekstu musi kończyć się specyficznym dla Atari NEWLINE (9B) i nie może być dłuższy niż 255 znaków. Procedura niszczy ewentualne dane w IOCB, ponieważ go wykorzystuje (normalna obsługa ekranu). Poza tym przy zmianach systemu może być ona przesunięta tylko razem z DSKERR (zobacz tam!).

C667 50791 F39D 62365 GETBLOCK

Przy czytaniu z kasety lub dysku jest to procedura wywoływana raz dla każdego bloku. Przy odczycie z kasety to CASSTART ma wartość różną od zera i przez JUMPTAB+2A jest wywoływana procedura CASREADBLK.

Gdy chodzi o czytanie dysku, to DSKCMD jest ładowany wartością READCMD (52), DSKUNIT ustawiany na 1 i przez JUMPTAB+03 wywoływana procedura SIO DISKINTERF.

C67C 50812 F3B2 62386 CASBOOT

Kiedy WARMFLAG wskazuje gorący start i bit 1 DOSACTIV jest 1, to jest przez CASINITC C wywoływana procedura CASIi1IT; w przeciwnym razie w przypadku gorącego startu wraca do wywołanego programu.

Gdy natomiast chodzi o zimny start, to można ładować, kiedy STARTTST przez wartość różną od zera sygnalizuje, że przy wystąpieniu zimnego startu był naciśnięty klawisz START. Gdy tak nie jest lub nie było, to CASBOOT wraca do miejsca wywołania.

W innym razie jest przeprowadzane OPEN na kasetę, przy czym Timeout jest ustawiany na "długi". W tym celu istnieje rejestr GAPTYPE. Jeżeli GAPTYPE jest ustawiony na wartość między zero i 127, to czytanie z kasety jest przerywane po krótkim czasie, gdy nie ma nowego bloku odczytania. Przy większej wartości (jako wartość ze znakiem: ujemna!)oczekiwanie jest znacznie dłuższe. Ma to ten sens aby czekać na początek zbioru na kasecie.

Właściwe OPEN jest przeprowadzane z ustawiony na 1 CASSTART przez JUMPTAB+2D i CASOPENIN, przy czym potem jest zaraz wywoływana BLOCK1. Jeżeli blok jest odczytany prawidłowo, to DOSACTIV wraca z BLOCK1 z wartością 2. Jeśli to ten przypadek, to jako ostatnia zawartość DOSINITV zostaje przepisana do CASINITV.

C6AE 50862 F3E1 62433 CASINITC

Przeprowadza pośredni skok przez CASINITV.

C6B1 50865 EDEA 60906 DISKINIT

DSKTIMOUT jest inicjowany wartością A0 i długość sektora dysku w DSKSECLEN jest ustawiana na 0080 (a więc 128 bajtów/sektor). Tu zaczyna być interesująca opisana w BOOT różnica między blokiem i sektorem!.

C6C1 50881 EDF0 60912 DISKINTERF

Ta procedura wykonuje wszystkie zadania inicjacji do odczytu lub zapisu sektora dysku, formatowania bądź odczytu statusu stacji dysków. W tym celu odpowiednie wartości są zapisywane w zmiennych DSKDEVICE, DSKSTATUS, przy rozkazie statusu DSKBUFFER, DISKTIMOUT i DSKBYTCNT. Te wartości stosuje się zależnie od rozkazu. Użytkownik ma do dyspozycji następujące:

Nazwa	Kod(hex) w DSKCMD	Opis
GET SECTOR	52 (R)	Czyta sektor i zapisuje go w pamięci od DISKBUFFER.
PUT SECTOR	50 (P)	Zapisuje sektor nr DSKAUX1 o długości DSKBYTCNT na dysku nr DSKDEVICE pełną zawartością pamięci od DSKBUFFER. W przeciwieństwie do 400/800 może 600XL/800XL wykonać ten rozkaz także przez opisany tu interfejs dysków.
WRITE SECTOR	57 (W)	Ten rozkaz przeprowadza takie same operacje jak PUT SECTOR, ale potem przeprowadza jeszcze weryfikację zapisanego sektora.
STATUS REQUEST	53 (S)	Żądanie podania statusu. Ten rozkaz sam ustawia sobie wartość z DSKBUFFER na DEVICSTAT, nie trzeba więc, obszaru bufora przekazywać na meldunek statusu. Rozkaz ten oddaje (poza DISK_READ_OK) cztery bajty w DEVICSTAT - DEVICSTAT+3: DEVICSTAT status rozkazu: bit0 jest ustawiany po nieważnym rozkazie. bit1 jest ustawiany po odczycie badsektora bit2 jest ustawiany po niepowodzeniu PUT SECTOR bit3 jest ustawiany, gdy dysk jest chroniony przed zapisem bit4 jest ustawiany, gdy po podaniu statusu stacja zamelduje z powrotem

		DEVICSTAT+1 status sprzętowy: **Ten bajt jest nadaną z dysku kopią bajta statusu kontrolera dysków FD1771, który przeprowadza całą obsługę zapisu i odczytu wewnątrz stacji dysków. Chociaż 1771 jako przestarzały kontroler posiada zupełnie inne właściwości sprzętowe niż nowe kontrolery, to słowa statusu są w nich wszystkich jednakowe, tak, że przy zastosowaniu innych kontrolerów nie powstają żadne trudności przy ich interpretacji. Tylko przy własnych projektach wykorzystujących kontrolery, które potrzebują do pracy znacznie więcej parametrów niż 1771/1797, mogą tu wystąpić trudności.**
		DEVICSTAT+2 wartość Timeout: **Zawartość tego rejestru określa, po ilu sekundach wystąpi Timeout.**
		DEVICSTAT+3 niewykorzystany
FORMAT DISK	**25 (!)**	**Ten rozkaz oczekuje jako parametru tylko kodu rozkazu. Przy tym rozkazie Timeout jest ustawiany znacznie wyżej, ponieważ formatowanie dysku trwa dłużej niż 7 sekund.** **Gdy formatowanie jest zakończone, użytkownikowi są przekazywane informacje o stanie dysku. W DSKBYTCNT znajduje się liczba defektów, a więc sektorów nie nadających się do zapisu po formatowaniu, a od DSKBUFFER są umieszczone numery uszkodzonych sektorów. Jako ostatni wpis w tym buforze znajduje się FFFF.**

Wartości do inicjowania DISKINTERF dla DSKTIMOUT i DSKBYTCNT są przekazywane tej procedurze każdorazowo w DSKTIMCON względnie DSKSECLEN. Adres bufora DSKBUFFER, rozkaz DSKCMD, jak również numer sektora (dla zapisu i odczytu) w DSKAUX1 (młodszy bajt) i DSKAUX2 (starszy bajt) muszą być przed wywołaniem DISKINTERF umieszczone w rejestrach.

Jako ogólna wartość powrotna jest zawarty w rejestrze Y bajt statusu, który w przypadku błędu jest ujemny. Flaga Negative CPU jest przy powrocie z procedury ustawiana odpowiednio do wyniku.

C748 51016 EE6D 61037 PUTADRESS

Kopiuje DSKBUFFER do BUFFERADR.

C758 51027 ---- ----- LOADER

Ta procedura nie jest wykorzystywana dla aktualnie istniejącego sprzętu firmy Atari, lecz daje użytkownikowi możliwości samodzielnego sporządzenia rozszerzeń tabeli HATABS i obsłużenia również tych nowych urządzeń przez IOCB. Jak wiadomo najłatwiej przez HATBS obsłużyć urządzenia zorientowane znakowo: edytor, ekran, klawiatura, drukarka, jak również magnetofon, nie przewidziano jednak w tej tabeli miejsca dla ewentualnych dalszych urządzeń. Ta wada modeli 400/800 jest tu usunięta za pomocą procedur takich jak LINK, UNLIK i innych.

Na początku tej procedury są kasowane RUNADR, HIUSED oraz ZHIUSED. Potem są czytane przez GETBYTEA dwa bajty, przy czym pierwszy bajt jest umieszczany w HIBYTELD, a drugi w SECLEN. Ogólnie obowiązuje dla tej

procedury, że wraca ona z ustawionym Carry i zawartością Y 90, gdy GETBYTEA melduje błąd przez flagę Carry.

Gdy HIBYTELD jest różne od 0B, jest wywoływana procedura, której adres jest NEWVECTOR + 2*HIBYTELD. Jeżeli po tej procedurze RECLEN jest równe LCOUNT, to znowu dwa bajty są ładowane do HIBYTELD i SECLEN, a gra idzie dalej. W innym przypadku tylko jeden znak jest czytany przez GSTBYTEA i pośrednio przez RUNADR jest wywoływana podana tam procedura, po czym LCOUNT zwiększa się i, gdy różny od zera, wracamy do miejsca porównywania RECLEN i LCOUNT. Jeżeli jednak LCOUNT jest tu zero, to procedura jest przerywana.

Jeżeli HIBYTELD jest równe 0B, to GETBYTEA jest wzywany dwukrotnie i oba odczytane znaki są umieszczane jako zawartość RUNADR i RUNADR+1, a więc definiują kompletny adres. Jeśli w tym miejscu RECLEN jest równe zero, to RUNADR jest znowu kasowany, a jeśli RECLEN = 01, to RUNADR pozostaje niezmieniony, zaś w pozostałych przypadkach do wartości RUNADR jest dodawane LODADRESS. W każdym przypadku jednak procedura powraca ze skasowaną flagą Carry i wartością Y = 01h.

C7DD 51165 ---- ----- GETBYTEAC

Przeprowadza skok pośredni przez (LODGBYTEA).

C7E0 51168 ---- ----- RUNADRC

Przeprowadza skok pośredni przez (RUNADR).

C7E3 51171 ---- ----- PUTCHAR

Ten podprogram potrzebuje dwóch parametrów: bajta informacyjnego w akumulatorze oraz bajta w LCOUNT.

Jeżeli bajt w akumulatorze ma wartość 0, to bajt z akumulatora jest umieszczany w NEWLDTMP1 i NEWADRLOD, a jeśli 1, to w NEWLDTMP1+1 i NEWADRLOD+1.

W ostatnim przypadku są jeszcze przeprowadzane obliczenia, które w szczególności muszą zgadzać się z zawartością z HIBYTELD:

HIBYTELD	wykonane obliczenie 16-bitowe
00 lub 0A	NEWADRLOD := NEWLDTMP1 + LODADRESS
inaczej	NEWADRLOD := NEWLDTMP1 + LODZLOADA

Potem jest umieszczany na stosie 16-bitowy rejestr pomocniczy, który zawiera wartość:

HIUSEDLOD+RECLEN-2

Przeprowadzane po tym porównanie znowu zależy od wartości HIBYTELD:

HIBYTELD	Porównanie 16-bitowe	Operacja gdy skuteczne
00 lub 0A	POMOCN = HIUSEDLOD	HIUSEDLOD := POMOCN
inaczej	POMOCN = LODZHIUSE	LODZHIUSE := POMOCN

Jeżeli rejestr pomocniczy jest większy niż MEMTOP, to procedura wraca Zawartością Y=9D i ustawioną flagą N w CPU do przedostatniej procedury w hierarchii, a więc nie do właściwego miejsca wywołania.

Jeżeli LCOUNT nie ma wartości ani 0, ani 1, to znajdujące się w akumulatorze bajty są przesyłane pod adresy określone przez wskaźnik LOADERTMP, który jest obliczany przez LOADERTMP := NEWADRLOD + LCOUNT - 2.

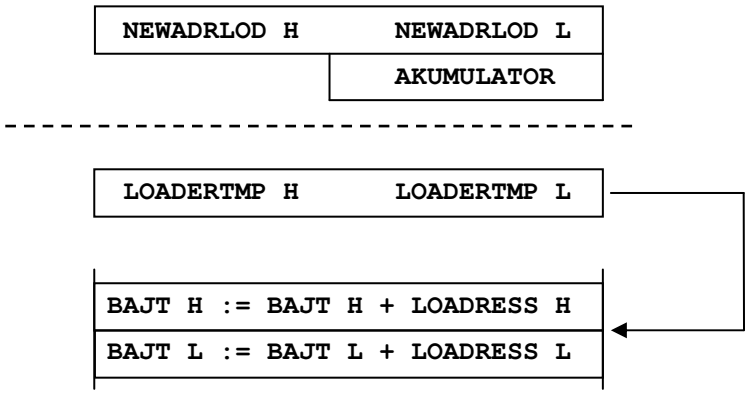
C87B 51323 ---- ----- ADD28E

Przeprowadza następujące obliczenia z wartością podaną przez akumulator:

LOADERTMP := NEWADRLOD + Aku

(LOADERTMP) := (LOADERTMP) + LOADRESS operacja 16-bitowa

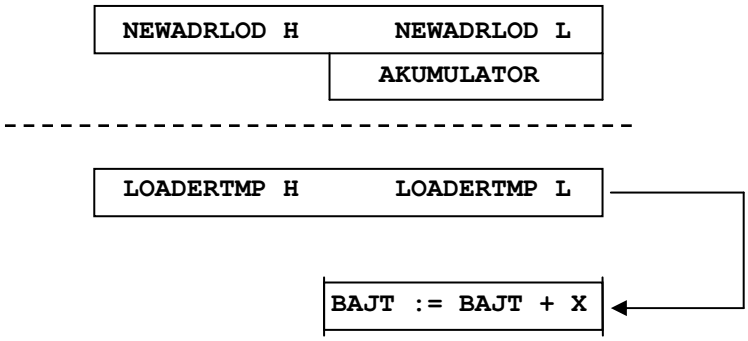
Poniższy rysunek ułatwia nieco wyjaśnienie tego:



C8A0 51395 ---- ----- ADD28EWRD

Jeżeli HIBYTELD ma wartość większą lub równą 1, to wartość X jest równa niższemu bajtowi LOADADRESS, inaczej zaś równą niższemu bajtowi LODZLOADA. Do akumulatora jest przesyłany bajt, który jest wykorzystywany w następujący sposób:

LOADERTMP := NEWADRLOD + Aku
(LOADERTMP) := (LOADERTMP) + X operacja 8-bitowa

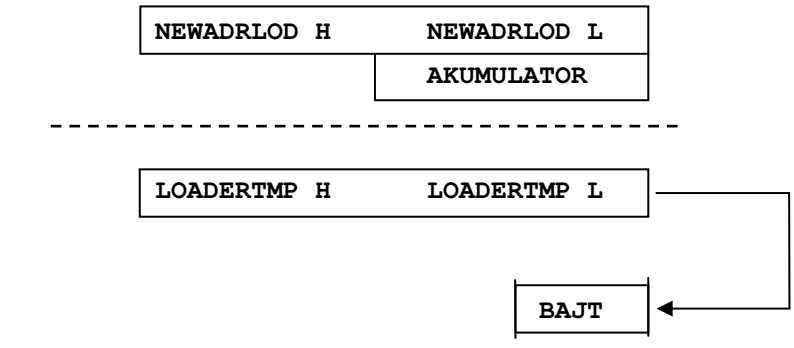


C8C3 51395 ---- ----- ADD28EGET

Znów jest przesyłany do akumulatora bajt, który tworzy odgałęzienie do adresu. Jeżeli LCOUNT ma przy wywołaniu tej procedury wartość parzystą, to jest przeprowadzane pierwsze obliczenie, inaczej drugie.

Przy parzystym LCOUNT:

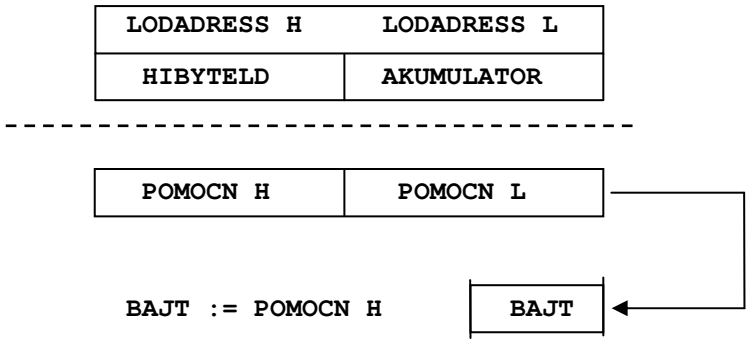
LOADERTMP := NEWADRLOD + Aku
HIBYTELD := (LOADERTMP) operacja 8-bitowa



a przy nieparzystym LCOUNT:

POMOCN := LODADRESS + Aku + 256 * HIBYTELD

(LOADERTMP) := POMOCN H operacja 8-bitowa



C8F2 51442 ---- ----- NEWVECTOR

Ta tablica zawiera punkty skoków dla pośrednich wywołań z LOADER. W szczególności są wektory skoków:

PUTCHAR
PUTCHAR
ADD28EWRD
ADD28EWRD
ADD28EWRD
ADD28EWRD
ADD28E
ADD28E

C90A 51466 ---- ----- SWITCHROM

Ta procedura włącza testowani ROM i startuje przez JUMPTAB+33 bezpośrednio do 5000. Włączenie ROM zachodzi przez ponowne ustawienie bitu 7 PORTB. Dla dokładnego powrotu do systemu operacyjnego jest w tym miejscu COLDSTART ustawiane na wartość FF (a więc start zimny).

C9A1 51482 ---- ----- NEWINIT

!!! UWAGA !!! Ta procedura pod żadnym pozorem nie może być wywoływana podczas przebiegu programu; prowadzi ona w każdym przypadku do runięcia systemu!

Jest ona przewidziana dla testowania później rozwiniętego sprzętu i wywołuje pod określonymi, łatwo osiągalnymi w normalnej pracy warunkami procedurę, która powinna być umieszczona w obszarze arytmetyki zmiennoprzecinkowej ROM. Ponieważ obecnie w tym miejscu właśnie znajduje się jeszcze ROM, to procedura skacze tu "na pustynię".

Mimo to omówimy tu krótko jej właściwe funkcje:

Ustawia ona, rozpoczynając od bitu 0, kolejno wszystkie bity w NEWPORT i sprawdza, czy w D803 jest wartość 80, a w D80B wartość 91. Jeżeli tak, to jest wywoływana procedura od D819, gdzie powinny być prawdopodobnie później skierowane IOCB i wskaźniki obsługi urządzenia, które ma się właśnie zameldować. Jeżeli procedura od D819 jest zakończona lub oba bajty nie są prawidłowo ustawione, to przeprowadzane jest całe sprawdzanie z następnym bitem NEWPORT.

W celu zakończenia ustawia się NEWPORT na zero.

C941 51521 E959 59737 SIOINTERF

Ta procedura tworzy punkt skoku przez wektor JUMPTAB+09 dla obsługi SIO.

Dla nas jest teraz interesujące tylko to, że najpierw jest włączany CRITTCIO, aby stąd wziąć procedurę SIO i po zakończeniu obsługi załadować rejestr Y wartością z DSKSTATUS, po czym CRITICIO ponownie jest kasowany.

Dla przypadku, gdy po wejściu do tej procedury rejestr NEWVORHDN nie jest zero, jest przeprowadzanych jeszcze kilka operacji. Najpierw rejestr X jest ładowany wartością 08 jako parametrem do przesłania do GETLOWEST i wywoływany jest GETLOWEST. Jeżeli procedura wraca ze skasowaną flagą Zero, to X jest zachowywany dla dalszego wykorzystania i wywołuje się D805 (ROMI: matematyczny!). Gdy flaga Carry jest jako status z D805 kasowana, jest powtarzana właśnie opisana procedura ze zmniejszonym X, w przeciwnym razie lub gdy GETLOWEST wraca z Z=1, to jest wywoływane SIO i przebiega dalej jak bez tej dygresji.

C97C 51580 ---- ----- NEWIOREQ

Ta część programu jest wzywana, jeżeli z przyszłego urządzenia I/O przyjdzie przerwanie. W takim przypadku wartość z NEWIOREQ jest zapisywana na stosie, aby teraz zapisać w NEWIOREQ i NEWPORT numer pozycji najniższego ustawionego bitu bajta przesłanego do akumulatora. Potem jest wprost wywoływana procedura leżąca (jeszcze) w matematycznym ROM, aby po jej zakończeniu ponownie umieścić w NEWPORT i NEWIOREQ stare wartości. Następnie, jak zwykle przy przerwaniu I/O, przywoływane zachowane wartości rejestrów X i A i następuje powrót przez RTI do przerwanego programu.

C99F 51615 ---- ----- NEWDEVC1

Ładuje Y wartością 01 i wywołuje CHECKNEWP.

C9A4 51620 ---- ----- NEWDEVC3

Ładuje Y wartością C3 i wywołuje CHECKNEWP.

C9A9 51E25 ---- ----- NEWDEVC5

Ładuje Y wartością C5 i wywołuje CHECKNEWP.

C9AE 51630 ---- ----- NEWDEVC7

Ładuje Y wartością 07 1 wywołuje CHECKNEWP.

C9B3 51635 ---- ----- NEUDEVC9

Ładuje Y wartością C9 i wywołuje CHECKNEWP.

C9B8 51640 ---- ----- NEVDEVCB

Ładuje Y wartością 0Bh i wywołuje CHECKNEWP.

C9BD 51645 ---- ----- GETLOWEST

Ta procedura oczekuje w X wartości między 1 i 8. Zależnie od wielkości tej liczby rozpoczynane jest od $2^{(8-X)}$ bitu wartości z NEWVORHDN sprawdzanie, czy jest on 1. Innymi słowami rozpoczyna się od najniższego bitu, gdy X ma wartość 8 i przerywa się procedurę, gdy X jest 0. Jeżeli zostanie w NEWVORHDN znaleziony dowolny bit, który ma wartość 1, to w NEWIOREQ i NEWPORT jest ustawiany dosłownie tylko ten bit, Po dołączeniu

odpowiedniego sprzętu ta procedura ma rozsądną wartość, nadając dla każdego ustawionego w NEWVORHDN bitu jeden raz życzenie (request) do urządzenia zewnętrznego. Procedura ta jest wywoływana z SIOINTEIRF, kiedy NEWVORHDN jest różne od zera, w przeciwnym razie jest wykorzystywana z CHECKNEWP.

C9D8 51672 ---- ----- NEWPORTERR

Również ta procedura nie ma zastosowania przy obecnym asortymencie sprzętu. Zawiera ona w Y wartość, która jest wykorzystywana jako odgałęzienie. Potem wartość z D80D+Y i z D80E+Y są umieszczane na stosie, a następnie akumulator jest ładowany wartością z 024C,a rejestr X z 024D, aby potem ponownie podać do Y wartość 92.

C9EA 51690 ---- ----- CHECKNEWP

Do tej procedury są przekazywane dwie wartości: jedna w akumulatorze i druga w rejestrze X. Są one umieszczone w 024C i 024D i zabezpieczają chwilowy stan CRITICIO, aby potem ustawić te flagi na wartość 1.

Na końcu wywoływane jest GETLOWEST w pętli z wartością X od 08. Jeżeli GETLOWEST wraca ze skasowaną flagą Zero, to jest wywoływane NEUPORTERR. Gdy ta procedura wraca z ustawioną flagą Carry, to wartość z akumulatora jest ładowana do 024C. Jeżeli GETLOWEST wraca z ustawioną flagą Zero, co oznacza, że nie istnieje dalszy port I/O, to Y jest ładowany wartością 82.

Niezależnie od tego, jaką wartość dostarczył dotąd GETLOWEST, są ustawiane na zero NEWIODREQ i NEWPORT. Potem CRITICIO jest ponownie ładowany wartością początkową, akumulator wartością z 024C, Y wartością z 024D, odpowiednio te wartości ustawiają flagi CPU i następuje powrót do miejsca wywołania.

Gdy flaga Carry po obsłudze tych ruchów jest zero, pętla jest przebiegana kolejny raz, jednak teraz ze zmniejszoną wartością X.

CA2F 51759 ---- ----- BITMASK

Jest to konieczne do zasłaniania bitów i zawiera wartości 80, 40, 20, 10, 08, 04, 02, 01.

CA37 51767 ---- ----- PREPLINK

Ta procedura jest właściwa dla otwarcia kanału IOCB, z którego będzie korzystał nowy, późniejszy sprzęt. Wywołuje ona INITLOAD, LINK+06 oraz DEVS2PL3+17 i skacze wtedy w środek procedury CIOMAIN, aby zakończyć instrukcję OPEN. Jeżeli podczas tej całej obsługi wystąpi błąd, to również skacze w środek CIOMAIN, lecz tym razem CIOMAIN sygnalizuje wartością Y = 82, że żądane urządzenie nie istnieje.

CA65 51813 ---- ----- Tablice

CB64h 52068 ---- ----- CHECKFF

Ta procedura tworzy sumę kontrolną (z Carry!) bajta, na który wskazuje ZCHAIN i 17 następnych bajtów. Jeżeli suma kontrolna ma wartość FF, to CHECKFF wraca z ustawioną flagą Zero, w przeciwnym razie flaga jest kasowana. Ta procedura sprawdza więc zawartość listy wskaźników, która jest zarządzana przez LINK lub UNLINK.

CB73 52083 ---- ----- FREE0

Odtąd do CBFF jest 141 wolnych bajtów, które mogą być użyte do rozszerzeń programowych. Uważaj przy starcie systemu na tworzenie sumy kontrolnej! Najlepiej po prostu przeprowadzić raz procedurę sumy kontrolnej i zanotować obliczone wartości dla każdego bloku (zob. CHECKROM1 i CHECKROM2).

CC00 52224 ---- ----- INTERCHAR

W tym miejscu rozpoczyna się drugi wewnętrzny zestaw znaków Atari. Jest on włączany przez ustawienie CC w CHARRBASE$ i umożliwia wtedy pisanie znaków specjalnych jak "â ä è ç" i wiele innych zamiast znaków graficznych.

D000 53248 D000 53248 Grupa układów I/O

E000 57344 E000 57344 STANDCHAR

Tu jest standardowy zestaw znaków, który jest załączany przez ustawienie E0 w CHARBASE$. Zawiera on znaki graficzne.

E400 58368 E400 58368 EDITORVEC

E410 58384 E410 58384 SCREENVEC

E420 58400 E420 58400 KBVEC

E430 58416 E430 58416 LPTVEC

E440 58432 E440 58442 HANDLERTB

Te adresy tworzą tabelę operatorów, która dla każdego urządzenia zawiera następujące wpisy:

OPEN adres procedury
CLOSE adres procedury
GET adres procedury
PUT adres procedury
STATUS adres procedury
SPECIAL adres procedury

złączony rozkaz skoku do każdorazowej inicjalizacji POWERON

jeden bajt pomocniczy

Urządzenie	OPEN	CLOSE	GET	PUT	STAT	SPEC	JMP	POWER	POMOC
EDITOR	EF93	F22D	F249	F2AF	F21D	F22C	4C	EF6E	00
SCREEN	EF8D	F22D	F17F	F1A3	F21D	F9AE	4C	EF6E	00
KEYBOARD	F21D	F21D	F2FC	F22C	F21D	F22C	4C	EF6E	00
LPT	FEC1	FF01	FEC0	FECA	FEA2	FEC0	4C	FE99	00
CASSETTE	FCE5	FDCE	FD79	FDB3	FDCB	FCE4	4C	FCDB	00

Poniższa tabela skoków jest jedyną możliwością ułożenia programów, które mogą być uruchamiane na 400/800 i na 600XL/800XL. Zawiera ona wszystkie istotne punkty bezpośrednich skoków do procedur bez konieczności tworzenia pośrednich wektorów.

Każda pozycja tabeli zawiera dokładnie jeden rozkaz skoku.

E450 58368 E450 58368 JUMPTAB+00

Skok do DISKINIT.

```
E453        58371       E453        58371       JUMPTAB+03
      Skok do DISKINTERF.

E456        58374       E456h       58374       JUMPTAB+06
      Skok do CIOMAIN.

E459        58377       E459        58377       JUMPTAB+09
      Skok do SIOINTERF.

E45C        58380       E45C        58380       JUMPTAB+0C
      Skok do SETVBLVEC.

E45F        58383       E45F        58383       JUMPTAB+0F
      Skok do SYSTEMVBL.

E462        58386       E4E2        58386       JUMPTAB+12
      Skok do EXITVBL.

E465        58389       E465        58389       JUMPTAB+15
      Skok do SIOINIT.

E468        58392       E468        58392       JUMPTAB+18
      Skok do SENDENABL.

E46B        58395       E46B        58395       JUMPTAB+1B
      Skok do NMIENABLE.

E46E        58398       E46E        58398       JUMPTAB+1E
      Skok do CIOINIT.

E471        58401       ----        -----       JUMPTAB+21
      Skok do TESTROMEN.

E474        53404       E474        53404       JUMPTAB+24
      Skok do RESETWARM.

E477        58407       E477        53407       JUMPTAB+27
      Skok do RESETCOLD.

E47A        58410       E47A        58410       JUMPTAB+2A
      Skok do CASREADBL.

E47D        58413       E47D        58413       JUMPTAB+2D
      Skok do CASOPENIN.

E480        58416       E480        58416       JUMPTAB+30
      Skok do TESTROMEN.
```

To nie jest błąd drukarski, ten skok jest rzeczywiście wzięty dwukrotnie. Pierwszy skok jest właściwie przejęciem skoku błędu, ponieważ tam w 400/800 znajduje się skok do etykiety SIGNON, przy której drukowany jest raport "ATARI COMPUTER - MEMO PAD" i wtedy następuje skok do funkcji echa. Ta funkcja jest w 600XL/800XL zastąpiona przez test pamięć ROM, do którego następuje teraz skok stąd.

Ponieważ ta "uzupełniająca" funkcja teoretycznie jest oddzielona od właściwej funkcji włączającej test ROM-u, więc są tu umieszczone oba, może tylko chwilowo różne wektory skoków.

E483 58419 ---- ----- JUMPTAB+33

Skok do TESTART.

Ten skok jest sensowny tylko wtedy, kiedy również test ROM-u jest włączony lub jego zawartość jest skopiowana w RAM od adresu 5000.

E486 58422 ---- ----- JUMPTAB+36

Skok do NEWDEVICE.

E489 58425 E489 58425 JUMPTAB+39

Skok do UNLINK.

E48Ch 53428 ---- ----- JUMPTAB+3C

Skok do LINK.

E48F 58431 ---- ----- CALLTAB

Tu leży 6 adresów w tabeli, na którą wskazują NEWDEVC1 - NEWDEVCB. Ma to ten cel, aby przy pomocy tych tabelarycznych wartości można było wywołać odpowiednie procedury urządzeń z RTS.

Powinno być ogólnie znane, że CPU 6502 przy skoku powrotnym z procedury żąda ze stosu wartości 16-bitowej jako adresu, na który wskazują ostatnie wykonywane bajty procedury, aby wykonać wtedy znajdujące się dalej rozkazy. Ten efekt można teraz wykorzystać, aby wywoływać procedury pośrednio przez tabelę. Trzeba do tego jedynie wywoływany adres-1 umieścić na stosie i przeprowadzić RTS (Return from Subroutine).

E49B 58523 ---- ----- NEWINITC

Następuje bezpośredni skok do NEUINIT.

E49E 58526 ---- ----- FREE1

Odtąd jest 35 bajtów pamięci programowej jeszcze wolnych. Przy zajmowaniu tych miejsc użytkownik musi pamiętać o skorygowaniu sum kontrolnych ROM-u (zob. CHECKROM1 i CHECKROM2)!

E4C0 58560 ---- ----- RTS

Tu stoi zupełnie osamotniony RTS. Jest on wykorzystywany przez różnorodne procedury.

E4C1 58561 E4A6 58543 CIOINIT

Ta procedura kasuje wszystkie IOCB przez ustawiając nazwy urządzeń CHANNELID (0340, 0350...) na FF i wskaźniki PUTBYTE (0346+7, 0356+7...) na CIONOTOPN jako raport błędu.

E4DC 58588 E4C4 58561 CIONOTOPN

Jeżeli próbowano skomunikować się z nieistniejącym urządzeniem, to jest wywoływana ta procedura. Umieszcza ona ponownie wartość 85 w Y, sygnalizując błąd DEVICE_NOT_OPEN przez ustawienie ujemnej flagi CPU.

E4EF 58591 E4C4 58565 CIOMAIN

Ta główna procedura wyjściowa oczekuje dwóch parametrów wejściowych: pierwszego dla nadawanego bajta w akumulatorze i drugiego dla numeru IOCB*16 kanału, po którym będzie wysyłany bajt. Mnożnik 16 występuje dlatego, że każdy IOCB zajmuje 16 bajtów.

Najpierw przesyłany bajt i numer IOCB są przesyłane do IOCBCHARZ wzgl. IOCBNUMZ, aby potem sprawdzić numer IOCB, czy po pierwsze nie jest za duży (ponieważ jest tylko 8 IOCB, to największym numerem jest 70), a po drugie, czy jest on podzielny przez 16. Jeżeli jeden z tych warunków nie jest spełniony, to w Y jest sygnalizowany błąd BAD_IOCB przez wartość 86.

Ogólnie można o tej procedurze CIO powiedzieć, że ona wraca z ustawioną flagą ujemności, kiedy zostanie napotkany błąd. Kod błędu może wtedy być rozpoznany przez wartość Y.

Potem blok leżący od IOCB jest kopiowany na IOCBCHIDZ, a więc możliwym miejscu strony zerowej.

Tu znów są przeprowadzane sprawdzenia odnośnie jeszcze nie wykorzystanych konstrukcji. Jeżeli jako numer kanału IOCBCHIDZ jest podana wartość 7F i IOCBCMDZ nie jest 0C (POWER_ON_INIT) oraz HNDLRLOAD jest zero, to również jest sygnalizowany błąd INVALID_CODE, jak gdyby IOCBCMDZ był mniejszy niż 3 (OPEN). Jeśli HNDLRLOAD nie jest zero, to wywoływany jest PREPLINK i ewentualnie meldowany błąd. Gdy żaden błąd nie wystąpił, kontynuowana jest obsługa rozkazu. Jeżeli już na początku IOCBCMDZ jest równe 0C, to natychmiast przeskakuje do rozkazu CLOSE.

W przeciwnym razie zależnie od rozkazu wywoływane są stąd CLOSE, CIOSTATSP, CIORREAD albo CIOWRITE.

Gdy HNDLRLOAD ma wartość różną od zera, to jest wywoływana procedura SPECHANDL, w przeciwnym razie DEVS2PL3. Jeśli ta procedura wraca z Carry=1, pomimo to skacze jeszcze raz do HNDLRLOAD, w innym wypadku kasuje DEVICSTAT i DEVICSTAT+1 i wywołuje COMENT.

E57C 58748 E54E 58675 CIOCLOSE

Tu jest, zależnie od przekazanego do CIO parametru, ustawiany odpowiedni IOCB na FREE (wolny) i należące do niego działanie CLOSE jest obliczane z tabeli i wywoływane.

E597 58775 E54E 58702 CIOSTATSP

Tu jest wykonywane, tak samo jak w CIOCLOSE, odpowiednie działanie dla rozkazu STATUS względnie SPECIAL, gdy istnieje żądane urządzenie. Jeśli nie, to jest ustawiane według możliwości i wtedy wykonywany jest skok do należącej do niego części obsługi.

E5B2 58802 E569 58729 CIOREAD

Na początku jest sprawdzane, czy ten IOCB nie jest chroniony przed odczytem. Jest tak, jeśli w IOCBAUX1Z bit 2 nie jest ustawiony. W takim przypadku CIOREAD wraca z błędem WRITE_ONLY. Urządzeniem chronionym przed odczytem jest na przykład drukarka.

Jeżeli chodzi o odczytywane urządzenie, to jeszcze porównywana jest do zera będąca do dyspozycji długość bufora w IOCBBUFLZ (16 bitów). Gdy jest zero, to tylko jeden znak jest odczytywany przez odpowiedni punkt i następuje powrót.

Gdy długość bufora jest różna od zera, daje to trzy możliwości przerwania odczytu:

Pierwszy warunek przerwania widzi procedura odczytu, gdy oznaczy błąd odczytu, a więc gdy wywołany ruch ustawi swoją flagę Carry, zanim wróci do CIOREAD.

Drugą może być urządzenie zorientowane blokowo (np. stacja dysków). Wtedy czytanie jest kończone, gdy długość bufora po odczycie ostatniego znaku będzie zero.

Trzecią możliwość daje się stworzyć przy urządzeniach zorientowanych wierszowo, jak drukarki. Przy tym czyta się tak długo, dopóki nie wystąpi NEWLINE (9B). Gdy bufor dotąd wystarczył, wszystko jest w porządku, w przeciwnym razie wszystkie znaki, które przyszły po wypełnieniu bufora, są ignorowane i ostatni bajt bufora jest zamieniany na NEWLINE. Mając np. bufor o długości 5 znaków i czytając wiersz o długości 8 znaków, znajdzie się w buforze:

1.ZNAK 2.ZNAK 3.ZNAK 4.ZNAK 5.NEWLINE

Poza tym w takim przypadku do rejestru Y przesyłane jest TRUNCATED_CODE.

Ogólnie można powiedzieć, że każdy blok, niezależnie od formatu, po odczycie kończy się znakiem NEWLINE. To przy 400/800 nie zawsze było wykonywane.

Rozróżnienie, czy chodzi o urządzenie zorientowane blokowo czy wierszowo, jest przeprowadzane przez bit 1 IOCBCMDZ. Gdy jest on zero, to chodzi o urządzenie zorientowane wierszowo.

Ostatnią operacją jest podanie w IOCBBUFLZ ostatecznej, prawidłowej długości bufora.

E61E 58910 E5C9 58825 CIOWRITE

Również przy zapisie najpierw sprawdza się przez IOCB, czy w ogóle jest dostęp do urządzenia. W tym celu musi być ustawiony bit 4 IOCBAUX1.

Jeżeli zapis jest dozwolony, to jest sprawdzane, czy długość bufora jest zero. Jeśli tak, to są przesyłane jedynie pojedyncze znaki. Ponieważ licznik długości bufora jest zmieniany, jest to sensowne przy założeniu, że przy wywołaniu tej procedury conajmniej jeden znak powinien być przesłany. Właściwe przesyłanie przebiega przez PUT urządzenia.

Jeżeli długość bufora jest większa od zera, to następne znaki są czytane z bufora i przesyłane, aby potem przez INCBFP i DECBUFL zmniejszyć wskaźnik bufora. Gdy przy zapisie znaku zostanie napotkany błąd, jest to odpowiednio wskazywane przez rejestr Y.

Jeżeli jest to urządzenie zorientowane wierszowo, to pozostaje tak długo w CIOWRITE, aż zostanie przesłany jako ostatni znak NEWLINE (9B), w przeciwnym razie tak długo, aż bufor zostanie opróżniony. Jeżeli przy urządzeniu zorientowanym wierszowo ostatnim znajdującym się w buforze znakiem nie jest NEWLINE, to jest on automatycznie ustawiany na końcu.

E670 58992 E4C4 58564 CIORETURN

Ta procedura zamykająca dla wszystkich rozkazów CIO ma dwa punkty wejścia: CIORETURN i CIORETURN+02.

Jedynym dodatkowym zadaniem CIORETURN w porównaniu z CIORETURN+02 jest umieszczenie w IOCBSTATZ przekazanej procedurze wartości rejestru Y (status operacji).

Potem znów zaprogramowany przez użytkownika adres buforu jest przenoszony do IOCBBUFAZ, a następnie cały IOCB strony zerowej jest przesyłany ponownie do obszaru użytkownika, aby następna część programu mogła wykorzystać IOCB. Nie można zapominać, że procedura CIO pracuje w programowanym przez użytkownika obszarze 0340–03BF, lecz kopiuje się w niższym obszarze, ponieważ znacznie szybciej i efektywniej można pracować

na stronie zerowej (wyjaśnia to odpowiednia literatura o metodach programowania CPU 6502). Po kopiowaniu HNDLRLOAD jest ustawiany na zero, a wtedy z powrotem podawane są ostatnio odczytany lub zapisany znak do akumulatora, indeks IOCB do X, a do Y status ostatniej operacji IOCB.

Przez ostatnie ładowanie rejestru Y jest znacząca flaga N statusu procesora.

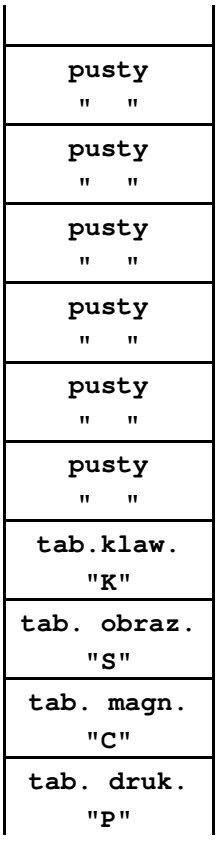
E695 59029 E63D 58941 COMPENTRY

Sprawdzane jest odgałęzienie kanału IOCBCHIDZ. Jeżeli jest większe niż 33, to numer kanału jest traktowany jako błędny i COMPENTRY wraca z błędem DEVICE_NOT_OPEN.

W innym wypadku przy HATABS+Y leży początek 3-bajtowego wejścia. Pierwszy bajt zawiera nazwę urządzenia, drugi i trzeci wskaźnik na początek struktury obsługi. Za strukturą obsługi zawiera adresy do pracy procedur CIO.

Te adresy początkowe są teraz umieszczane w IOCBAUX2 i IOCBAUX3. Następnie jest tworzone wewnętrznie odgałęzienie obsługi, które podaje się z rozkazu i CNDTAB. Potem wskaźnik IOCBAUX2 i IOCBAUX3 jest przestawiany tak, że wskazuje bezpośrednio początek procedury obsługi.

Całe to przejście pozwala się najprościej wyjaśnić na rysunku:



Rejestr Y ma np. wartość 06h

HATABS

Najpierw jest umieszczany w IOCBAUX2-3 wskaźnik na wejście "tabeli edytora", a potem sama tabela edytora.

E6BB 59067 E633 58931 DECBUFL

Tu następuje zmniejszenie 16-bitowej wartości długości bufora w IOCBBUFLZ. Flaga Zero jest ustawiona, gdy po zmniejszeniu licznik jest zero.

E6C8 59080 ---- ----- DECBFP

Zmniejszany jest wskaźnik bufora w IOCBBUFAZ. Nie może to spowodować żadnej relacji przez flagi, to jednak jest bezsensowne, dokładnie jak przy INCBFP, ponieważ wszystkie rozstrzygnięcia programowe zachodzą nie przez wskaźnik bufora, lecz przez licznik długości bufora.

E6D1 59089 E670 58992 INCBFP

IOCBUFAZ jest zwiększane o jeden (działanie 16-bitowe).

E6D8 59096 E677 53999 SUBBFL

Tu jest obliczana efektywna długość bufora. Przy urządzeniach zorientowanych wierszowo nieznana jest ilość odsyłanych danych. Można ją określić przez to, że poszukuje się wewnątrz bufora znaku NEWLINE, z drugiej strony ta procedura udostępnia liczbę przesłanych znaków w IOCBBUFLZ jako wartość powrotną. Oblicza ona w tym celu po prostu

IOCBBUFLZ := IOCBBUFLx - IOCBBUFLZ

przy czym x jest numerem urządzenia określone poprzez IOCBBUFLZ.

E6EA 59114 E689 59017 GOHANDLER

Ta procedura wywołuje przez CIOJUMP wcześniej obliczony manipulator urządzenia. W tym celu najpierw ładuje rejestr Y wartością FUNCTION_NOT_IMPLEMENTED (92h), aby zabezpieczyć wystąpienie raportu błędu, gdy wywołany potem manipulator prowadzi tylko do RTS. Po powrocie z procedury manipulatora jest ustawiana flaga Negative zależnie od wartości Y. Flaga Carry jest zawsze ustawiana na jeden.

E6F4 59124 E693h 59027 CIOJUMP

Właśnie tu jest zastosowany proces, który jest opisany w CALLTAB. Adres początkowy podprogramu manipulatora jest umieszczany na stosie i potem manipulator jest wywoływany pośrednio przez RTS. Fakt, że chodzi o adres początkowy przy koniecznej do tego wartości tabelarycznej, jest bardzo ważny przy tworzeniu nowej tabeli wektorów manipulatorów. Łatwo można manipulator uczynić niezdolnym do działania przez to, że został podany błędny adres. Jest także do pomyślenia, że otrzymywany przy każdym wejściu urządzenia adres skoku do inicjowania Power-Up musi wskazywać na odpowiednią procedurę, ponieważ Power-Up nie pośrednio przez tą funkcję CIOJUMP, lecz przez bezpośrednio przed tym adresem stojący kod JMP-Up jest wywoływany. Ta idea nie jest łatwo zrozumiała, ale po pewnym przyzwyczajeniu także tam można łatwo rozpoznać błąd.

E6FF 59135 E6B8 59064 DEVS2PL3

Ten podprogram określa wartość z IOCBDSNZ, a więc aktualnie wykorzystany numer stacji dysków. W tym celu jest czytany numer, który znajduje się w kolejnym bajcie pod (IOCBBUFAZ) Gdy leży on między 31 i 39 (kody hex cyfr 1-9), to do IOCBDSKNZ przesyłana jest odpowiednia wartość 0C - 09, w przeciwnym razie wartość jeden.

E712 59154 E69E 59038 DEVICSRCH

Tu jest znak leżący przy (IOCBBUFAZ) interpretowany jako nazwa urządzenia i odszukiwany w tabeli od HATABS. Jeżeli zostanie znaleziony zgodny wpis, to jest zapisywany w IOCBHIDZ, z prawidłową nazwą, a flaga Carry jest kasowana. Jeżeli w tabeli HATABS nie ma takiego wpisu, to sygnalizowany jest błąd przez ustawienie flagi Carry.

E72D 59181 E6C9 59081 COMTAB

Ta tabela tworzy polecenie dla każdego rozkazu IOCB na wpis wektora w tabeli procedur manipulatorów od HATABS.

Prowadzi	Rozkaz	Na wektor od bajta
3	OPEN	0
4, 5	GET_RECORD	4
6, 7	GET_CHARACTER	4
8, 9	PUT_RECORD	6
10, 11	PUT_CHARACTER	6
12	CLOSE	2
13	STATUS_REQUEST	8
14	SPECIAL	10

E739 59103 ---- ----- LINKSOMETH

Również ta część programu jest przewidziana dla późniejszego użycia przez dalsze urządzenia peryferyjne, przy których obszar HATABS będzie zbyt mały. Trzeba pamiętać, że w HATABS jest jeszcze miejsce dla pracy sześciu dalszych urządzeń.

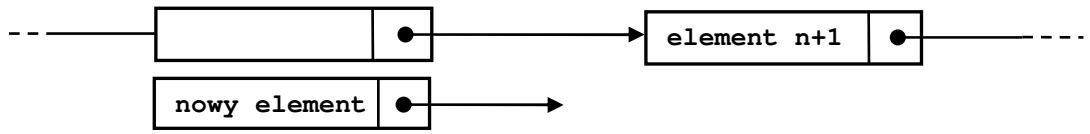
Wszystkie te procedury LINK i UNLINK wychodzą z tego, że od adresu STARTTST znajduje się struktura o długości 19 bajtów. Jest przy tym interesujące, że korzystanie z tej instrukcji jest podpatrzone z wykorzystania kaset. Jest to całkiem zrozumiałe, ponieważ oba stojące w tym miejscu adresy są wykorzystywane tylko do ładowania z kasety. Wychodząc teraz z tego, że ten ogromny nakład programowania prowadzi do dołączenia do Atari ekskluzywnego sprzętu, musi być jasne, że przewidziane do tego oprogramowanie nie będzie ładowane z kaset.

Na początku procedury jest sprawdzane, czy WARMFLAG jest ustawiony. Jeżeli tak, to ZCHAIN jest ustawiany na bezpośrednią wartość STARTTST i rozpoczyna się pętla kontrolna.

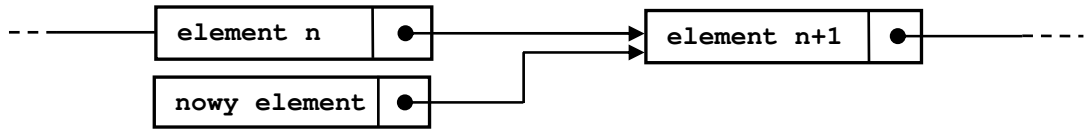
Przedtem, zanim tę pętlę i jej punkt wyjściowy będziemy mogli opisać, trzeba wprowadzić nieco teorii odnośnie użycia list liniowych w Atari 600XL/800XL.

Pod pojęciem liniowej listy rozumie się obraz, przy którym każdy element ma jedno odesłanie do następnego elementu. Jest to np. częsty przypadek w tekstach programów BASIC-u. W wierszu programu są umieszczone: numer wiersza, adres logicznie następującego po nim wiersza. i potem właściwe dane. Jest teraz wskaźnik na pierwszy wiersz, znajdzie się przez niego drugi, potem trzeci itd. Ten proces może niekoniecznie jest najszybszy, ma jednak kilka ważnych zalet. Trzeba np. przy włączaniu nowego wiersza jedynie tak przestawić wskaźnik, aby wskazywał nowy element i wskaźnik nowego elementu ustawić na element wskazywany przedtem przez poprzedni. Dzięki temu nowy wiersz leży logicznie w środku tekstu, a gdzie on jest fizycznie (a więc w pamięci), to jest zupełnie obojętne.

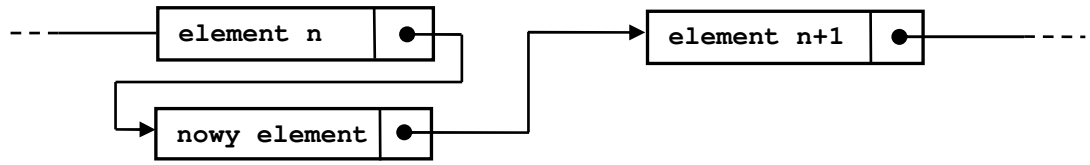
1) Istnieje liniowa lista, na której szczegółowo wskazane są dwa elementy. Dokładnie między te dwa elementy trzeba wstawić nowy element:



2) W tym celu najpierw wskaźnik elementu n jest kopiowany we wskaźniku nowego elementu:



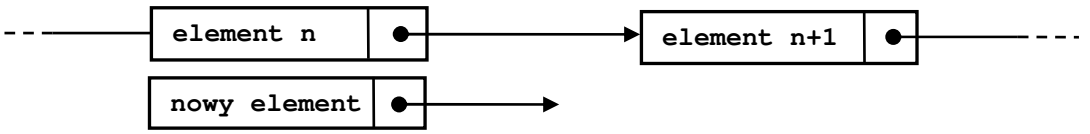
3) Teraz zbyteczny wskaźnik elementu n jest tak przestawiany, aby wskazywał nowy element. Odtąd nowy element jest zawarty w liście liniowej



Ten sposób włączenia nazywany jest "włączeniem prostym", ponieważ zawsze istnieje tylko jeden wskaźnik na następnika, lecz niejeden na poprzednika. Takie połączenie było znane w informatyce jako "podwójnie włączona lista".

Z pewnością musi być jasne, że nie można łatwo usunąć z listy takiego elementu względnie, że musi być podany dokładny protokół dla włączenia nowego elementu lub usunięcia umieszczonego w liście. W powyższym przykładzie np. wykonując najpierw krok 3 i potem krok 2 tracimy wszystkie elementy od elementu n+1

Pozycja wyjściowa wygląda jak pokazano wcześniej:

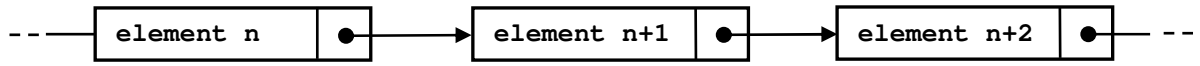


2) Przenosimy wskaźnik elementu n w nowy element:

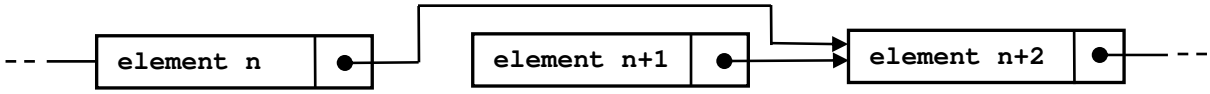
Już mamy taki efekt, że teraz wskaźnik nowego elementu nie wskazuje na żadne sensowne miejsce i nie można teraz ustawić go na element n+1, bo gdzie wcześniej mielibyśmy zanotować położenie elementu n+1? Stanęliśmy na nowym elemencie i włączanie jest "sknocone".

Przy wyłączaniu elementu problemy są nieco innej natury:

Przyjmijmy, że mamy liniową listę, z której przedstawiamy trzy gdziekolwiek w środku leżące elementy:



Mamy teraz wyłączyć element n+1, więc trzeba jedynie wskaźnik z elementu n tak przestawić, by miał taką samą wartość jak element n+1. W ten sposób element n+1 jest omijany:



Problemem przy tym jest zaznaczenie, które obszary pamięci są teraz jeszcze wypełnione elementami listy, a które są wypełnione zbędnymi elementami. Wykonanie tych obliczeń zużywa względnie dużo czasu. W innych systemach (np. w Microsoft Basic możliwym do otrzymania na Atari) jest wbudowana cała obsługa ciągów w dynamicznym zarządzie pamięci. Wyraz dynamiczny mówi, że nie jest zarezerwowana stała przestrzeń pamięci, lecz zawsze tylko dokładnie taka, jaka jest konieczna. Przy użyciu bardzo wielu, bardzo krótkich ciągów wydaje się, że komputer "stoi", chociaż po 2-3 minutach liczy dalej. Jest on wtedy w tzw. procedurze Garbage-Collect, która gromadzi ciągi-śmiecie.

Dodatkowo trzeba jeszcze wyjaśnić, co dzieje się ze wskaźnikiem ostatniego elementu. Nie wskazuje on "w powietrze", lecz otrzymuje zdefiniowaną wartość, która jest zwana ogólnie NIL. Jest ona przeważnie równa zero.

Po tym krótkim wyjaśnieniu dynamicznego zarządzania danymi wracamy do naszej, właśnie rozpoczętej pętli. Przy użytych tu elementach ustawia włączenie każdego elementu w bajcie numer 18 i 19, odnosząc do adresu początkowego każdego elementu.

ZCHAIN wskazuje np. na dowolny element, więc pod nazwą ZCHAIN.LINK rozumiemy włączanie (LINK) elementu, na który wskazuje ZCHAIN.

```
ZCHAIN      bajt 0
            bajt 1
               .
               .
               .
            bajt 18     oba te bajty są w tym
            bajt 19     przypadku ZCHAIN.LINK
```

Pętla zostaje przerwana np. gdy ZCHAIN.LINK dostarcza wartość NIL, a więc chodzi przy tym elemencie o ostatni na liście.

W przeciwnym przypadku jest sprawdzany następny element. Zachodzi to przez podstawienie:

```
ZCHAIN := ZCHAIN.LINK
```

Przez to ZCHAIN wskazuje teraz na następny element. Teraz jest sprawdzane, czy suma wszystkich wpisów od zerowego do siedemnastego bajta jest równa FF (CHECKFF). Jeżeli nie, to również pętla jest przerywana.

W przeciwnym razie jest ustawiana flaga Carry (przez CALLVECC1) i, również przez to, skok do CALLVEC, co z drugiej strony powoduje, że jest wykonywany skok do bajta rozkazu, który stoi w bajcie 12 struktury oznaczonej przez ZCHAIN. Proszę przypomnieć sobie, że tabela manipulatorów od EDITORVEC ma również od 12 bajta rozkaz skoku do procedury inicjowania

odpowiedniego manipulatora. Dzięki temu może być wyjaśniona funkcja i budowa nowowprowadzonej struktury. Jest ona funkcjonalnie identyczna z tabelą manipulatorów od EDITORVEC, a w budowie różni się jedynie parametrami koniecznymi do zarządzania listą.

Jeżeli ten manipulator inicjowania wraca teraz z ustawioną flagą Carry, to również przerywa pętlę, w przeciwnym razie stosuje wyżej opisany test NIL odnośnie następnika na nowy element.

Jeżeli na początku procedury WARMSTART był skasowany, to CHAINLINK jest ustawiany na NIL i rozpoczyna dalsze inicjowanie IOCB w programie.

E76E 59246 ---- ----- *???????*

Po wywołaniu z DCBINIT, przy którym jest inicjowany IOCB0 z danymi dla stacji dysków nr 1, odbywa się wywołanie SIOINTERF przez JUMPTAB+_09.

Jeżeli SIOINTERF wraca z negatywnym statusem, to jest to odbierane jako błąd podobnie do procedury CIO i ten błąd jest natychmiast podawany "do góry".

Jeżeli przesyłanie było w porządku, to CHAINTP1 jest ładowany przez sumę z MEMLO i status urządzenia DEVICSTAT.

Jeśli po tych operacjach MEMTOP jest mniejszy niż CHAINTP1, to DSKAUX1 i DSKAUX2 są ładowane przez 4E i wywoływany jest DCBINIT. Potem następuje skok do początku procedury.

Jeżeli MEMTOP jest większy lub równy CHAINTP1, to wartość z CHAINTMP jest zapamiętywana, aby potem załadować CHAINTMP z MEMLO. Następnie jest wywoływany INITLOAD z zapamiętanym młodszym bajtem z CHAINTMP w akumulatorze.

Gdy ta procedura wraca z ustawioną flagą N, to jest wywoływany LINK. W przeciwnym razie, zależnie od flagi Carry, przy skasowanej Carry jest ta procedura rozpoczynana zupełnie od nowa, a przy ustawionej dalej przechodzi tam, gdzie DCBINIT jest wywoływany z wartością 4E w DSKAUX1 i DSKAUX2.

Właściwym punktem wyjściowym z tej pętli jest więc negatywna wartość powrotna z SIOINTERF.

E7BE 59326 ---- ----- DCBINIT

Tu są dane z TABSICINI ładowane do bloku kontroli dysku od DSKUNIT, wartość z akumulatora do DSKAUX1 i wartość z rejestru Y do DSKAUX2, aby potem wykonać skok do SIOINTERF przez tabelę skoków przy JUMPTAB+09.

E7D4 59348 ---- ----- TABSIOINI

Tu znajdują się wartości początkowe dla DCB (zob. DCBINIT). Są to w szczególności:

Nazwa	Wartość	Opis
DSKDEVICE	4F	ignorowane
DSKUNIT	01	stacja dysków nr 1
DSKCMD	40	SIO-CMD GET_DATA
DSKSTAT	40	ignorowany
DSKBUFFER	02EA	tu jest DEVICSTAT, obszar przechowania statusu
DSKTIMOUT	001E	odpowiada 30 sekundom oczekiwania do Timeout
DSKBYTECNT	0004	informacja o statusie składa się z 4 bajtów

E7DE 59358 ---- ----- INITLOAD

Do tej procedury jest przekazywana w akumulatorze wartość dla CHAINTP1H, która jako pierwsza jest tam umieszczana. Niższa część CHAINTP1L

jest ustawiana na zero, a TEMP3 na FF. Gdy bit 0 w CHAINTMP jest "1", więc wektor CHAINTMP jest nieparzysty, to CHAINTMP jest zwiększany. Zabezpiecza to, że CHAINTMP wskazuje parzysty adres po tej procedurze.

Potem ładowane są LODADRESS wartością z CHAINTMP, LODGBYTEA adresem INCLOAD i LODZLOADA wartością 80, a następnie jest wywoływany LOADER.

E816 59414 ---- ----- INCLOAD

Po zwiększeniu TEMP3, jeśli wynik nie jest zero, akumulator jest ładowany bajtem (C37D + wartość TEMP3) oraz kasowana jest flaga Carry.

Jeżeli po zmniejszeniu TEMP3 jest zero, to jest ustawiany na 80 i wywołana jest DCBXINIT, co ładuje DCB wartością dla rozkazu NOTE_SECTOR i wywołuje SIOINTERF. Gdy przejście NOTE przebiegło pozytywnie, również akumulator jest ładowany bajtem (037D + wartość TEMP3) i flaga Carry jest kasowana.

E833 59443 ---- ----- DCBXINIT

Tu jest, analogicznie do DCBINIT, ładowany blok kontroli dysku wartością z SIOTAB i przez JUMPTAB+09 wywoływany jest SIOINTERF.

E851 59473 ---- ----- SIOTAB

Tu znajdują się wartości początkowe DCB dla rozkazu NOTE (zobacz DCBXINIT). Są to w szczególności:

Nazwa	Wartość	Opis
DSKDEVICE	00	ignorowane
DSKUNIT	01	stacja dysków nr 1
DSKCMD	26	NOTE_SECTOR
DSKSTAT	40	ignorowany
DSKBUFFER	03FD	tu jest początek obszaru RAM niezajętego przez OS
DSKTIMOUT	001E	odpowiada 30 sekundom oczekiwania do Timeout
DSKBYTECNT	0080	będzie czytany blok 128-bajtowy

E85D 59485 ---- ----- SEARCHLIS

W akumulatorze i rejestrze Y są podawane starszy i młodszy bajt adresu elementu listy, według którego trzeba szukać od adresu STARTTST.

Danymi wyjściowymi są zawartości akumulatora i rejestru X, oraz flaga Carry. Flaga Carry daje następującą informację:

1) Carry=0: poszukiwany wektor został znaleziony i w A jest starszy bajt, a w X młodszy bajt tego wektora.
2) Carry=1: albo LINK ma wartość NIL, albo wśród przeszukiwanych struktur znalazła się taka, której suma kontrolna nie ma wartości FF (wywołanie CHECKFF!)

W pierwszym przypadku np. gdy szukany jest LINK(n+1)dostajemy następujący obraz:

```
03F9  element 1   element 2   element n   element n+1
      LINK(2)     LINK(3)     (LINK(n+1)
ZCHAIN
```

E894 59540 ---- ----- CALLVECC1

Skacze z ustawioną flagą Carry w środek procedury LINK, aby najpierw przez CALLVEC skoczyć do manipulatora Power-Up.

E898 59544 ---- ----- LINK

Ta procedura ustawia nowy element struktury, którego adres początkowy jest podany w akumulatorze (starszy bajt) i rejestrze Y (młodszy bajt), na końcu istniejącej listy. W tym celu najpierw trzeba odszukać ostatni element istniejącej listy (wywołanie SEARCHLIS z A=X=0) i jego wartość NIL tak zmienić, że ten Link wskazuje teraz na nowy element (zob. LINKSOMETH!). Potem Link nowego elementu jest ustawiany na NIL i bezpośrednio skacze na 12 bajt nowej struktury, co powoduje, że stojący tam rozkaz skoku przekazuje kontrolę programu do procedury Power-Up nowo zainstalowanego urządzenia. Jeżeli ta procedura wraca z ustawioną flarą Carry, to ten element jest podobnie znowu usuwany (wywołanie UNLINK ze starymi wartościami A i Y) i następuje powrót do miejsca wywołania.

Jeżeli to nie jest ten przypadek, jest sprawdzane, czy wywołana procedura ma przekazaną skasowaną czy ustawioną flagę Carry. Jeżeli ona jest skasowana, to kasowane są bajty 16 i 17 nowej struktury (MINTLK i GINTLK). Potem, niezależnie od flagi Carry, ustawiany jest na nową wartość MEMLO.

MEMLO := MEMLO + ZCHAIN.MINTLK (operacja 16-bitowa)

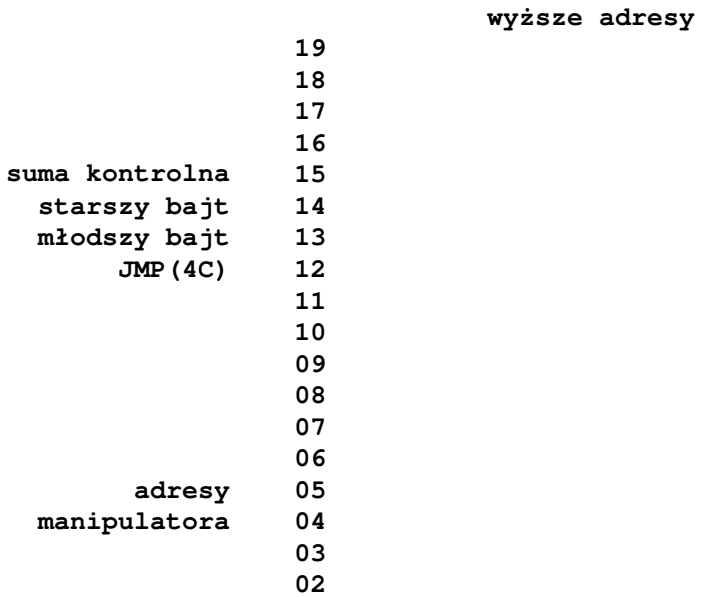
Zatem z wywołanej procedury może być określone, czy po wstawieniu nowego elementu MEMLO powinno być zmienione, czy nie. Gdy Carry jest ustawione, MEMLO jest zmieniane, w przeciwnym razie nie. Ma to sens szczególnie wtedy, gdy ta procedura wywołuje początek systemu i kieruje się wtedy wszystkie programy jak DOS, BASIC i podobne do nowego, zmienionego MEMLO.

Następnie jest obliczana suma kontrolna nowej struktury (CHECKFF) i wynik jest umieszczany w 15 bajcie struktury. Bajt 15 jest tez np. przy EDITORVEC bajtem pomocniczym - teraz przyjmuje jedną funkcję.

Warto w tym miejscu wspomnieć, że CHECKFF oprócz flagi Zero podaje jeszcze w akumulatorze różnicę między sumą kontrolną i wartością FF. Wartość ta może być teraz wstawiona, przez co przy dalszym sprawdzeniu suma kontrolna jest prawidłowa.

E900 59648 ---- ----- CALLVEC

To skacze do 12 bajta struktury na którą wskazuje ZCHAIN:



01
ZCHAIN 00
niższe adresy

E912 59666 ---- ----- SETBVBLVECC

Ten adres zawiera jedynie skok do SETBVBLVEC.

E915 59669 ---- ----- UNLINK

Przez SEARCHLIS jest szukany określony w akumulatorze i Y element w liniowej liście od STARTTST. Jeśli nie zostanie znaleziony, to procedura wraca z ustawioną flagą Carry.

Znaleziony element jest również kasowany, gdy COLDSTART jest ustawiony, a więc różny od zera, lub gdy bajty 16 i 17 są zero a suma kontrolna struktury ma wartość FF (CHECKFF). W tych przypadkach element jest usuwany w sposób szczegółowo opisany w LINKSOMETH.

Jeżeli element zostanie skasowany, to UNLINK wraca ze skasowaną, a w innym wypadku ustawioną flagą Carry.

E9569 59737 ---- ----- JMPSIOINT

Przeprowadza bezpośredni skok do SIOINTERF.

E95C 59740 E944 59716 SIOINIT

Ta procedura służy do inicjowania POKEY-a. Wyłączany jest tu silnik magnetofonu i dźwięk. Następnie linia !COMMAND wyjścia szeregowego jest ustawiana w stan wysoki (nieaktywny).

E971 59761 E959 59737 SIO

Ta procedura przejmuje wszystkie zadania koordynacji sterowania szyny szeregowej.

Najpierw, aby uniknąć błędnej interpretacji przerwania, CRITICIO jest ustawiany na jeden i w DSKDEVICE sprawdzane jest, czy przy podanym urządzeniu chodzi o magnetofon kasetowy. Jeżeli tak, to jest wykonywany skok do CASENTER - program obsługi magnetofonu. Różnica ta jest w tym miejscu istotna, ponieważ wszystkie inne urządzenia poza magnetofonem posiadają wewnętrzny kontroler, który daje im "inteligencję".

Jeżeli mamy do czynienia z takim inteligentnym urządzeniem, to CASFLAG jest ustawiany na zero, aby po powrocie z procedury odczytu wzgl. zapisu można prawidłowo reagować. Potem DRETRY i CRETRY są inicjowane wartościami C1 lub CD i prędkość przesyłania ustawiana jest na około 19200 bitów/sek. Ostatnie ustawienie zachodzi przez programowanie kanałów częstotliwości 3 i 4, które są ładowane wartościami 20 wzgl. 0C.

Następnie identyfikator urządzenia, który składa się z rodzaju i numeru urządzenia, rozkaz, jak również pierwszy i drugi bajt pomocniczy DSKAUX1 i DSKAUX2 są ładowane w odpowiednich zmiennych od CMDDEVIC, aby potem DISKBUFFER mógł wskazać, na którym miejscu znajduje się obsługiwany rozkaz. BUFENDPTR wskazuje na miejsce bezpośrednio za CMDAUX2.

Jeżeli te obsługi są zakończone, to linia !COMMAND jest przez PORT B ustawiana w stan niski, aby powiedzieć wszystkim urządzeniom "UWAGA, idzie nowy adres urządzenia!". Również to nadanie jest wykonywane przez procedurę SENDINIT. Jeżeli wzywane urządzenie odpowie błędnym statusem lub zupełnie inne urządzenie zgłosi gotowość, to jest z powrotem podawany w ERRORFLAG kod błędu, który jest różny od zera. W takim przypadku lub gdy było zanotowane w Y, że urządzenie sygnalizowało NEGATIV_ACKNOWLEDGE przez

ustawienie Y na zero, to CRETRY jest zmniejszany i, gdy nie jest zero, ponownie próbuje wezwać urządzenie. Jeśli to się nie powiedzie, następuje meldunek błędu z wywoływanego modułu.

Gdy jednak urządzenie zamelduje się właściwie, DSKSTATUS jest odczytywany i sprawdzany, czy ma dodatnią wartość. Jeżeli tak, to przed pierwszym nadaniem lub odczytem zestawu danych oczekuje się jeszcze, aż urządzenie wykona zupełnie podany przedtem rozkaz lub wystąpi błąd Timeout. Potem jest nadawany do urządzenia zestaw danych rozkazu. Blok kontroli dysku musi zawierać odpowiednie parametry dla nadawania.

Po właściwym wykonaniu tego rozkazu DSKSTATUS wskazuje, czy są ewentualnie jeszcze dane do czytania z urządzenia. Jeśli tak, to są one czytane przez RECEIVE.

Jeżeli podczas całej obsługi nie wystąpi żaden błąd, to do DSKSTATUS jest dodawane z powrotem C1, w przeciwnym razie kod błędu. Kody błędów są zawsze ujemne i flaga N jest przy powrocie z SIO ustawiona.

Na SIO wskazuje JUMPVEC+09.

EA37 59959 EA1A 59930 WAIT

Ten moduł jest wykorzystywany do wysłania rozkazu do inteligentnego urządzenia do oczekiwania na odpowiedź.

Tą odpowiedzią może być np., że urządzenie nie może wykonać tego rozkazu (na przykład zapis na dysku chronionym przed zapisem) lub też pozytywny raport, który tu jest tworzony, przez meldunek COMPLETE.

Jeżeli nadejdzie odpowiedź negatywna lub do Timeout nie nadejdzie żadna odpowiedź, to rejestr Y jest ładowany przez 00 i w ERRORFLAG jest umieszczony odpowiedni błąd dla wywołanego modułu.

EA88 60040 EA6B 60011 SEND

Po wywołaniu SENDENABL sterowane przerwaniami wyjście jest pobudzane przez to, że pierwszy bajt z określonych przez DSKBUFPTR jest zapisywany do rejestru wyjściowego SEROUT POKEY-a. Gdy ten znak zostanie przesłany, przerwanie jest kasowane.

Dalej w tej procedurze jest jedynie sprawdzane, czy klawisz BREAK został naciśnięty. Jeśli tak, to przesyłanie jest przerywane.

W przeciwnym wypadku czeka się, aż XNITEND zostanie ustawiony na wartość różną od zera, ponieważ oznacza to koniec przesyłania.

Na koniec jeszcze jest wyłączany nadajnik przez SENDDIS.

EAAD 60077 EA90 60048 ISRODN

Ta procedura obsługi przerwań (Interrupt Service Routine if Output Data Needed) jest wywoływana, kiedy POKEY sygnalizuje, że przesłał ostatni znak z rejestru wyjściowego SEROUT do swojego wewnętrznego rejestru szeregowo-równoległego. Wtedy DSKBUFPTR jest zwiększany i sprawdzany, czy osiągnął wartość BUFENDPTR. Jeśli tak się stanie, co oznacza koniec, i przesyłany jest bajt sumy kontrolnej i ustawiana flaga CHKSUMSND.

Gdy są jeszcze znaki do przesłania - kolejny z nich jest pobierany z (DSKBUFPTR) i przekazywany do SEROUT.

EAEC 60140 EAD1 60113 ISRXD

Jeżeli POKEY sygnalizuje przy przerwaniu, że zakończył przesyłanie bajta zawartego w rejestrze równoległo-szeregowy i jeszcze żaden nowy bajt nie został zapisany w SEROUT, to jest wykonywana ta procedura.

Gdy była już nadana suma kontrolna, jest sygnalizowane w XNITEND przez wartość różną od zera, że przesyłanie szeregowe zostało zupełnie

zakończone. Po zakończeniu procedury przerwania XMITEND jest sprawdzany przez SEND.

EAFD 60157 EAE2 60130 RECEIVE

Jeżeli przy tym rozkazie czytania chodzi o kasetę, to jest kasowany rejestr sum kontrolnych DSKCHECKSUM, w innym wypadku nie. Potem zarówno przy magnetofonie, jak i przy każdym inteligentnym urządzeniu są kasowane flagi BUFFULL i RECEIVEND. Po wywołaniu RECEIVEN, w którym najpierw jest umożliwiany właściwy odbiór, jest w pętli sprawdzane, czy:

1) naciśnięty jest klawisz BREAK. Wtedy odbiór danych zostaje przerwany.
2) był sygnalizowany przez TIMEFLAG warunek Timeout (jest tak gdy TIMEFLAG jest wyzerowany.
3) RECEIVED ma wartość różną od zera.

W każdym z tych trzech przypadków odczyt jest kończony, przy czym w dwóch pierwszych wskazywany jest wywołującemu modułowi błąd przez DSKSTATUS.

EB2C 60204 EB11 60177 ISRSIR

Ta procedura obsługi przerwania przy gotowości wejścia szeregowego (Interrupt Service Routine at Serial Input Ready), jest wywoływana, gdy przy przerwaniu POKEY przez swoje flagi wyjawia mniemanie, że odebrał znak i przesłał go do SERIN.

W takim przypadku z SKSTAT jest odczytywany status POKEY-a i ewentualne błędne bity są kasowane przez zapis w SKSTASTRES. Potem jest sprawdzane, czy rzekomo będący w pogotowiu znak był prawidłowo wczytany, lub czy został napotkany błąd FRAMING lub OVERRUN. W każdym z tych dwóch błędnych przypadków procedura odpowiednio ustawia DSKSTATUS.

Gdy znak jest wpisany jako prawidłowo przeniesiony, jest przez BUFFULL sprawdzane, czy wszystkie znaki do odczytania zostały napotkane. Jeśli tak, a więc BUFFULL nie jest zero, to znak, który jest jeszcze, jest interpretowany jako suma kontrolna i po odczycie znaków sprawdzane jest, czy zgadza się ona z obliczoną sumą kontrolną. Jeżeli suma kontrolna nie jest w porządku, to błąd sumy kontrolnej jest sygnalizowany w DSKSTATUS. Niezależnie od tego jest sygnalizowane przez ustawienie RECEIVEND, że trzeba przerwać RECEIVE.

Jeżeli BUFFULL jest zero, to znak jest odczytywany, dodawany do sumy kontrolnej i zapisywany do bufora od (DSKBUFPTR). Potem DSKBUFPTR jest zwiększany i sprawdza się, czy został osiągnięty BUFENDPTR. Gdy DSKBUFPTR jest nadal mniejszy niż BUFENDPTR, to procedura przerwania jest opuszczana bez zmiany flag.

Przy równości obu wskaźników następuje koniec właściwego przesyłania danych. Może być teraz odebrana warunkowo suma kontrolna. Jeśli tak, a więc NOCHKSUM jest zero, to BUFFULL jest ustawiany na FF dla następnego przerwania i procedura przerwania jest opuszczana.

Jeżeli według NOCHKSUM nie należy już oczekiwać od nadajnika sumy kontrolnej, to NOCHKSUM jest kasowany dla następnego wywołania RECEIVE i przed opuszczeniem procedury RECEIVED ustawiany jest na FF (odbiór całkowicie zakończony).

EB87 60295 EB6E 60270 LOADPTR

Są tu przeprowadzane dwa obliczenia:

DSKBUFPTR := DSKBUFFER

BUFENDPTR := DSKBUFFER + DSKBYTCNT

Nie oznacza to nic innego jak tylko to, że konieczne dla SIO dane początku i końca bufora są przesyłane z DCB (Disk Control Block) do bloku kontroli SIO.

EB9D 60317 EB84 60292 CASENTER

Skok do tego punktu następuje, gdy chodzi o I/O magnetofonu kasetowego.

W tym celu najpierw jest rozróżniane, czy chodzi o odczyt, czy zapis, ponieważ przy odczycie prędkość wejściowa jest mierzona, natomiast przy zapisie jest ustalona.

Jeżeli więc chodzi o zapis, to asynchroniczne liczniki wyjściowe 3 i 4 POKEY-a są programowane na prędkość 600 bitów/sek. Wtedy blok kontroli SIO jest inicjowany przez LOADPTR i blok danych jest nadawany przez SEND. CASENTER służy również do włączania i wyłączania silnika magnetofonu.

Jeżeli natomiast chodzi o odczyt, to CASFLAG jest ustawiany i, również po ustawieniu wskaźnika bufora SIO przez LOADPTR, ustalana jest prędkość odczytu dla tego bloku danych przez wywołanie procedury BEGIREAD. Poza tym TIMER1 jest inicjowany w taki sposób, że funkcjonuje jako źródło przerwania Timeout.

Po wywołaniu RECEIVE, w którym blok albo jest prawidłowo odczytany, albo też odpowiednio ustawiane są flagi, silnik magnetofonu jest wyłączany, gdy jest to potrzebne (tylko po przesłaniu ostatniego bloku End Of File) i następuje powrót do miejsca wywołania.

EC11 60433 EBF0 60400 TIMER1INT

Skok do tej procedury następuje wtedy, gdy przy przerwaniu wygaszania pionowego TI::COC::T1 przechodzi na zero. Procedura ta jest wykorzystywana do pilnowania Timeout tego urządzenia, które nie odpowiedziało lub nie może odpowiedzieć, rozpoznania tego Timeout i spowodowania przerwania operacji przez odpowiedni raport błędu. W tym celu do TIMEFLAG jest po prostu ładowane zero.

Aby TIMER1 mógł znaleźć tę procedurę, w procedurze SETTIM1V ładowany jest do TIMER1VEC adres TIMER1INT.

EC17 60439 EBF6 60406 SENDENABL

Tu jest inicjowane nadawanie przez złącze szeregowe w SKCNTL i jego rejestrze-cieniu. Następnie przy korzystaniu z kaset ładowane są rejestr dźwięku 2 niższą częstotliwością (3995Hz) i rejestr dźwięku 1 wyższą częstotliwością (5327Hz) dla używanego przy tym procesu dwutonowego. Potem kasowane są w IRQEN$ ewentualne stare przerwania i zezwalane jest przerwanie wyjścia danych (Output Data Interrupt). Z kolei wykonywany jest skok do środka procedury RECEIVEN, gdzie pozostałe rejestry POKEY-a są inicjowane dla przesyłania danych.

EC40 60480 EC1F 60447 RECEIVEC

Tu jest, analogicznie do SENDENABL, ustawiany POKEY na asynchroniczne przesyłanie danych i zezwolenie przerwania odbioru w IRQEN.

Poza tym zapisywany jest SKSTATRES, aby skasować ewentualne stare meldunki błędów.

Następnie znajduje się tu punkt "wskoku" dla SENDENABL, przy którym kanał 3 taktu jest ustawiany na 1,77MHz, a kanał 4 na wyjście kanału 3.

Potem rejestry kontroli AUDICNTL1 - AUDICNTL4 są odpowiednio do wartości IOSOUNDEN ustawiane tak, że "dźwięk przesyłania" występuje tylko przy ustawionym IOSOUNDEN.

Jeżeli nie będą przeprowadzane żadne operacje z magnetofonem, to kanały dźwiękowe 1 i 2 są blokowane. Są one tylko konieczne dla dwutonowego procesu obsługi kasety.

EC84 60548 EC63 60515 SENDDIS

Tu są kasowane bity 3, 4 i 5 w IRQEN$, co wstrzymuje wszelki odczyt i zapis przez przerwanie. Następnie są kasowane 4 rejestry kontroli dźwięku AUDCNTLx.

EC9A 60570 EC79 60537 SETTIMOUT

Ta procedura dostarcza produkt z młodszego bajta DSKTIMOUT i stałej 32. Daje ona starszy bajt wyniku w rejestrze X i młodszy bajt w rejestrze Y.

ECA9 60585 EC88 60558 INTTAB

Tu znajdują się trzy wektory przerwań ISRSIR, ISRODN i ISRXD.

ECAF 60591 EC8E 60558 SENDINIT

Po krótkiej zwłoce, która daje układom I/O okazję do ustawienia się wywoływany jest SCEND, a następnie WAIT.

Po powrocie z WAIT wartość rejestru Y jest kopiowana w akumulatorze, aby ustawić flagi. Jeżeli wystąpi Timeout lub urządzenie nie może prawidłowo przeprowadzić operacji, to ustawiana jest flaga Zero.

ECC8 60616 ECA7 60583 COMPUTE

Ta procedura oblicza wartość dla rejestrów częstotliwości 3 i 4 POKEY-a przy obsłudze kasety. Wywołuje ona przy tym ADJUST i sama jest pobudzana wyłącznie z BEGINREAD.

ED2E 60718 ED08 60680 ADJUST

Ten podprogram jest wykorzystywany do ustawiania zawartości POKEY-a. Przy użyciu urządzeń PAL zawartość akumulatora jest podwyższana o 20, w przypadku, gdy nie osiąga ona jeszcze 7C. Gdy zawartość akumulatora jest większa od 7B, to odejmowane jest 7C.

Przy systemie NTSC zawartość ta jest podwyższana jedynie o 07.

ED3D 60733 ED14 60692 BEGINREAD

Ta część programu ustawia POKEY na pomierzoną prędkość przesyłania danych z magnetofonu. W tym celu zakłada się, że dwa pierwsze bajty czytanego bloku mają wartość AA (10101010 bin), która doskonale nadaje się do synchronizacji.

Gdy synchronizacja jest zakończona, dwa bajty 55 są umieszczane w buforze i inicjują sumę kontrolną zakładając, że zgadza się ona z obydwoma bajtami 55. Gdy wystąpi Timeout lub naciśnięty zostanie klawisz BREAK, to pojawi się odpowiedni meldunek w DSKSTATUS, a silnik magnetofonu jest wyłączany.

EDE2 60898 EDBD 60692 SETTIM1V

Ta procedura przez JUMPTAB+0C ustawia wektor skoku dla TIMER1 (TIMER1VEC) na TIMOUTINT, a sam TIMER1 (TIMCOUNT1) na wartość podaną X (starszy bajt) i w Y (młodszy bajt).

EDF9 60921 EDD2 60882 POKTAB

Tabela przeliczeń dla COMPUTE.

EE11 60945 ---- ----- TABELLE

EEB1 61105 ---- ----- ADJTAB

W ADJTAB znajduje się wartość czasowa dla ADJUST przy wersji NTSC, a w ADJTAB+1 przy wersji PAL.

EEBC 61116 ---- ----- NEWDEVICE

Ta procedura szuka od adresu HATABS urządzenia o nazwie podanej w X. Jeżeli znajdzie taki wpis, to następuje powrót z ustawioną flagą Carry, przy czym Rejestr X wskazuje adres manipulatora znalezionego wpisu (a więc poniżej nazwy!), a Rejestr Y wskazuje stan wejścia.

Gdy poszukiwane urządzenie nie zostało znalezione, procedura sprawdza jeszcze raz wszystkie wpisy, czy jest tam wpis pusty. Jeśli tak, to zawartość akumulatora jest interpretowana jako starszy bajt, a Y jako młodszy bajt adresu tabeli manipulatorów i razem z podaną w X nazwą są rejestrowane jako nowy wpis HATABS.

W takim przypadku bit Carry jest kasowany. Jeżeli nie ma żadnej wolnej nazwy, to Rejestr Y jest ustawiany na FF, a Carry na 1.

EEF9 61177 ---- ----- SPECHANDL

Ten manipulator specjalny wywołuje DCBINIT z następującymi parametrami:

Akumulator - znak z adresu (IOCBBUFAZ)

Rejestr Y - wartość IOCBDSKNZ (numer stacji dysków)

Jeżeli DCBINIT wraca z negatywną wartością powrotną, to Y jest ładowany przez NON_EXISTENT_DEVICE i procedura się kończy.

W przeciwnym wypadku są ładowane następujące rejestry wartościami:

IOCBCHDIZ	7F
IOCBPUTBZ	adres z PUTBYTE-1
IOCBSPARE+IOCBCHIDZ	znak z (IOCBBUFAZ)
IOCBSPARE+IOCBCHIDZ+1	CHAINTMP
Rejestr Y	01

EF26 61222 ---- ----- PUTBYTE1

Natychmiast po rozpoczęciu tej procedury sprawdzane są trzy możliwe źródła błędów:

1) Cztery niższe bity akumulatora mogą być różne od zera. Wtedy do Y ładowany jest INVALID_IOCB_NUMBER i obsługa jest przerywana.
2) Taką samą wartość powrotną zawiera Y, gdy zawartość X >= 80.
3) Jeżeli STARTTST ma wartość zera, to w rejestrze Y jest meldowane NON_EXISTENT_DEVICE

Gdy nie występuje żaden z tych przypadków, wartość X jest wykorzystana jako IOCBNUMZ i IOCB określony przez IOCBCHIDZ jest kopiowany w zarezerwowanym dla IOCB obszarze strony zerowej, aby z tą nową wartością wywołać PREPLINK. Gdy ta procedura wraca z negatywną wartością, także procedura PUTBYTE jest przerywana z wartością Y FUNCTION_NOT_IMPLEMENTED_IN_HANDLER.

W innym przypadku zawartość IOCBPUTZ (16 bitów) jest umieszczana na stosie i wykonywany jest skok do tego wektora przez RTS.

EE65 61285 ---- ----- FREE1

Tu jest 6 wyzerowanych i niewykorzystanych bajtów wolnej pamięci programowej do wykorzystania. Przy ich zajmowaniu należy pamiętać o skorygowaniu sumy kontrolnej (CHECKROM1 i innych!).

EF6B 61291 ---- ----- CASMOTOFFC

Bezpośredni skok do CASMOTOFF.

EF6E 61924 F3E4 62436 POWERONA

Tu jest kasowany LSTCH (ustawiany na FF), RAMTOP jest ustawiany na RAMSIZE, SHIFTLOCK na 40 (tylko duże litery), KEYDEFPTR na KEYDEF FKTDEFPTR FKDEF.

EF8E 61326 ---- ----- INITSOME

W tym miejscu podczas zimnego i gorącego startu inicjowana jest większość zmiennych systemowych, np. CHARBASE, CHARSTPTR, DMACNTL$, CHARCNTL, MONSTATUS, IRQST$, IRQST, CURSORINH, TEXTINDEX, ADDRESS, TABMAP, COLPFO$...COLBAK$, TEXTSTART i inne.

F180 61824 F593 62867 GETCH

Czyta znak z podanej pozycji kursora i przekazuje go do akumulatora. Pozycja kursora jest przy tym zwiększana.

F18F 61839 F18F 61839 GETPLT

Czyta znak leżący pod ADRESS i zamienia go z kodu ekranowego na ATASCII.

F1A4 61860 F5BD 62909 OUTCH

Znak podany do akumulatora jest sprawdzany, czy jest on CLEAR_ _SCREEN(7D) czy NEWLINE (9B) i jeśli tak, to jest wywoływana odpowiednia procedura, zaś w przeciwnym wypadku zarządzanie znakiem przejmuje OUTPLT, a przedtem kursor jest zwiększany.

F1CA 61898 F5E0 62944 OUTPLT

Gdy flaga wyjścia obrazu STARTSTOP jest różna od zera, to Atari chodzi w pętli tak długo, aż flaga zostanie skasowana.

Potem znak podany w akumulatorze jest umieszczany na pozycji kursora.

W tej procedurze jest punkt OUTCH2 (F1E9), do którego często są wykonywane skoki z innych modułów.

F20B 61963 F621 63009 RETURM

RETURn from Monitor stara się o to, żeby przy włączonym trybie graficznym nie był widoczny kursor i przy skutecznie przeprowadzonej operacji I/O zaopatruje DSKSTATUS w wartość SUCCESS.

W tym miejscu nie można pominąć milczeniem, że tu, w środku procedury (krótko przed końcem) trzeba omijać następną etykietę. Uważamy to za potknięcie dobrze uporządkowanego systemu operacyjnego dla ludzi, którzy niechętnie kupują ideały. Tak więc procedura RETURNM kończy się pod adresem F22D, a nie przed poniższym skokiem.

Nieistotne jest również, że adres F21E jest wykorzystywany jako punkt wskoku z SWAP. Jest to miejsce, na którym jest sygnalizowany meldunek powodzenia wywołanej zmiany. Ten adres w 400/800 jest F634.

F223 61987 ---- ----- TESTROMEN

Jest to jedynie bezpośredni skok do SWITCHROM. Przed ewentualną zmianą proszę przeczytać poprzedni punkt!

F22E 61998 ---- ----- SCROLFINE

Jeżeli 7 bit FINESCROL jest zero, to obsługa procedury jest przerywana, w przeciwnym wypadku wyłączane jest przerwanie Display List, FINESCROL jest zerowany, a wektor programu ANTIC-u na C0CE, a więc na MASKTAB-1, a następnie wykonywany jest skok do procedury INITSOME.

F24A 62026 F63E 63038 EGETCH

Ta procedura jest wykorzystywana, aby móc podać i wyedytować pełną listę logiczną, a więc max 120 znaków. Jako wartość powrotna jest podawany w akumulatorze pierwszy znak linii, jeśli klawisz BREAK nie został naciśnięty. Gdy wystąpił ten przypadek, to w akumulatorze jest podawany NEWLINE (9B), a rejestr Y zawiera wartość 80 jako status.

Gdy wejście było w porządku (bez BREAK), to Y ma wartość 01. Ponieważ przy pierwszym wywołaniu może być przekazany tylko pierwszy znak podanej linii, trzeba teraz dla każdego następnego znaku wywołać EGETCH. Wtedy automatycznie zawsze dostarczony będzie następny znak. Koniec linii logicznej jest wskazywany wywołującemu modułowi również przez NEWLINE.

F2AD 62125 F6A1 63137 JSRIND

Następuje tu pośredni skok do (ADDRESS).

F2B0 62128 F6A4 63140 EOUTCH

Znak podany w akumulatorze jest umieszczany na pozycji kursora. Znaki sterujące są wykonywane.

F2F8 62200 F6DD 63197 KGETC2

Nie jest to żaden punkt skoku, to etykieta konieczna dla następnej procedury KGETCH.

F302 62210 F6EA 63202 KBGETCHAR

Tu jest odczytywany znak z klawiatury i sprawdzany na różne znaki specjalne. Jednym z takich znaków był np. znak o kodzie 89. Ten znak specjalny, już niedostępny w 600XL/800XL, włączał "klick" klawiatury, gdy był on wyłączony, i odwrotnie. Poza tym procedura obsługuje też dostępne w 600XL/800XL znaki specjalne, np. CONTROL-1.

F3E0 62432 F779 63353 ESCAPE

ESCAPEFLAG jest ustawiany na 80. Jest to konieczne, aby rozkazy kursora można było przedstawić na ekranie jako znaki specjalne.

F3E6 62438 F77F 63359 CURSORUP

Kursor przenoszony jest o jeden wiersz do góry lub z najwyższego wiersza na najniższy.

F3F3 62451 F780 63372 CURSORDWN

Kursor przenoszony jest o jeden wiersz do dołu lub z najniższego wiersza na najwyższy.

F400 62464 F799 63385 CURSORLFT

Kursor przenoszony jest o jedną pozycję w lewo lub ze skrajnego lewego położenia na skrajne prawe w tym samym wierszu.

F411 62481 F7AA 63402 CURSORRIG

Kursor przenoszony jest o jedną pozycję w prawo lub ze skrajnego prawego położenia na skrajne lewe w tym samym wierszu.

F420 62496 F7B9 63417 CLEARSCREEN

Obraz jest kasowany razem z LOGICMAP i wywoływana funkcja CURSORHOM.

F440 62528 F7D6 63446 CURSORHOM

Kursor jest umieszczany w lewym górnym rogu (pozycja HOME). Zawartość ekranu nie zmienia się.

F450 62544 F7E6 63462 CURSORBS

Przeprowadzana jest funkcja BACK SPACE. Znak stojący bezpośrednio z lewej strony kursora zostaje skasowany i kursor zajmuje jego miejsce. Jeśli kursor stoi na początku linii to rozkaz jest ignorowany.

F47A 62586 F810 63504 CURSORTAB

W BITMASK wyszukiwana jest następna pozycja tabulacji i wykonywany jest do niej skok.

Jeżeli nie ma następnej pozycji tabulacji w wierszu, to jest wykonywany skok na początek następnego wiersza.

F495 62613 F82D 63533 CURSOSTAB

Kolumna, w której znajduje się kursor przy wywołaniu tej procedury, jest pozycją jest pozycją tabulacji przez stawienie odpowiedniego bitu w BITMASK.

F49A 62618 F832 63538 CURSOCTAB

Ewentualne istniejąca w kolumnie kursora pozycja tabulacji jest kasowana.

F49F 62623 F837 63543 INSCHAR

Na pozycji kursora umieszczana jest spacja.

F4D5 62677 F86D 63597 DELCHAR

Znak stojący logicznie z prawej strony kursora jest kasowany, a wszystkie następne są przesuwane o jedno miejsce w lewo.

F50C 62732 F8A5 63653 INSLINE

Na pozycji kursora jest wstawiana nowa linia logiczna.

F520 62752 F8D4 63700 DELLINE

Wskazywana przez pozycję kursora linia logiczna jest kasowana. Wszystkie następne linie przesuwane są do góry.

F556 62806 F90A 63754 BELL

Tu jest 32 razy wywołany KEYCLICK. Daje to dźwięk o czasie trwania ok. 0,25sek.

F55F 62815 ---- ----- BOTTOMLIN

Ta procedura umieszcza kursor w najniższej linii obrazu i w pierwszej kolumnie.

F565 62821 7917 63767 DBDDEC

ADDRESS jest dwukrotnie zmniejszany i sprawdzany, czy rejestr znajduje się jeszcze w podanych granicach lub czy wystąpił SCREEN_ERROR.

F5AC 62892 F947 65815 CONVERT

Adres kursora określony przez numer wiersza i kolumny jest przeliczany na rzeczywisty adres w pamięci.

F60A 62986 F9D4 63956 INCRSB

Po zwiększeniu pozycji kursora jest sprawdzane, czy znajduje się on na końcu wiersza lub obrazu. Jeżeli jest to ten przypadek, to automatycznie uruchamiany jest przesuw (scrolling).

F6AE 63150 FA7A 64122 SUBEND

Według wartości X (0C lub 02) ustalany jest ENDPOINTR z PLOTROWAC lub z PLOTCOLAC.

F6BC 63164 FA88 64136 ERANGE

ERANGE jest punktem skoku dla edytora. Jest tu sprawdzane, czy edytor jest otwarty i przy negatywnym wyniku dokonuje otwarcia.

Następnie jest sprawdzane, czy kursor leży wewnątrz swoich normalnych granic. Jeśli nie, to wywoływany jest CURSORHOM i sygnalizowany błąd CURSOR_OVERRUN oraz pobierany jest ze stosu ostatni adres powrotny. W ten sposób błąd przekazywany do modułu wywołującego CIOMAIN.

F715 63253 ---- ----- JMPF21E

Następuje bezpośredni skok do środka procedury RETURN. Ten skok jest wykorzystywany przez SWAP.

F718 63256 FAE4 64228 OFFCURSOR

Kursor jest wyłączany.

Wszystkie następujące teraz procedury z nazwami BITxxx odnoszą się do obsługi tabulacji:

F723 63267 FAEB 64235 BITCON

Maska od MASKTAB+ Akumulator jest umieszczana w BITMASK i w X jest podawana 1/8 akumulatora.

F732 63282 FAFA 64250 BITROL

LOGICMAP do LOGICMAP+2 są przesuwane o jeden bit w lewo. Najwyższy Bi z LOGICMAP jest umieszczany w Carry.

F73C 63292 FB04 64260 BITPUT

Ustawia określony przez akumulator bit w TABMAP.

F74A 63306 FB12 64274 BITCLR

Kasuje określony przez akumulator bit w TABMAP.

F758 63320 FB20 64288 LOGGET

Ładuje do akumulatora CURSROW+120 i przechodzi do następnej procedury.

F75D 63325 FB25 64293 BITGET

Daje z powrotem ustawioną flagę Carry, gdy określony przez akumulator bit jest ustawiony.

F76A 63338 FB32 64306 INATAC

Podany w akumulatorze kod ekranowy znaku zamieniany jest na kod ATASCII, o ile nie jest to symbol graficzny.

F78E 63374 FB4E 64334 LINEINSERT

Tu zachodzi zależne od sprzętu przesunięcie fizycznego wiersza. Są wykorzystywane rejestry ADDRESS, MLTTMP i SAVEADR.

F7C2 63426 FB7B 64379 EXTEND

Ta procedura przedłuża linię logiczną o dalszą fizyczną.

F7E2 63458 FB9B 64411 CLEARLINE

Ta procedura wykonuje zależne od sprzętu kasowanie (nie usunięcie!) fizycznego wiersza.

F7F7 63479 ---- ----- DOSCROLL

Ta procedura wykonuje "delikatny" przesuw na ekranie, jak również część obsługi grafiki.

F8B1 63665 FC00 64512 DOBUFC

Ta procedura oblicza długość bufora chwilowej linii logicznej, przy czym ostatnie spacje są ignorowane.

F90C 63756 FC5C 64604 STRBEG

BUFSTR jest ustawiany na początek wiersza określonego przez pozycję kursora.

F918 63768 FC68 64616 DELTIA

Ta procedura kasuje pusty wiersz fizyczny, o ile jest on ostatni fizycznie w wierszu logicznym.

F93C 63804 FC8D 64653 TESTCNTL

Ta procedura przeszukuje tabelę znaków kontrolnych od CONTROLS. Gdy znak zostanie znaleziony, to X otrzymuje wartość odgałęzienia na znaleziony wpis odnośnie CONTROLS. W przypadku powodzenia jest ustawiana flaga Zero.

F94C 63820 F09D 64669 PHACRS

CURSROW, CURSCOL i GRAPHEMUL są umieszczane od TEMPROW w dół.

F957 63831 FCA8 64680 PLACRS

Umieszczone przez PHACRS pozycje kursora są przywracane.

F962 63842 FCB3 64691 SWAP

Zmienne od CURSROW do OLDGRADR+1 są zamieniane z TEXTROW do TEXTGRAD+1 i SWAPFLAG jest odwracany. Ta zamiana musi być wykonana, gdy obraz jest wewnętrznie przełączany z tekstu na grafikę lub odwrotnie.

F983 63875 FCD8 64728 KEYCLICK

W tym miejscu jest generowany "klick" klawiatury.

F997 63895 FCE4 64740 COLCR

CURSCOL jest zero gdy SWAPFLAG jest zero, a GRAPHEMUK różne od zera. W przeciwnym razie CURSCOL jest ustawiany na wartość z LEFTMARGIN.

F9A6 63910 FCF3 64755 PUTMSC

Wartość z SCRNSTART jest ładowana do ADDRESS.

F9AF 63919 FCFC 64764 DRAWTO

Zależnie od rozkazu w IOCBCMDZ albo jest kreślona linia (DRAW), albo wypełniany obszar (FILL). Linia jest kreślona od OLDGRROW, OLDGRCOL do adresu podanego w CURSROW i CURSROW.

FB04 64260 FE45 65093 TABELLE

FB0D 64269 FEC6 65222 CONTROLS

Ta tabela zawiera wszystkie konieczne znaki sterowania i odpowiednie adresy manipulatorów.

FB11 64273 ---- ----- FKDEF

Tu znajduje się 16 bajtów dla każdego klawisza funkcyjnego F1-F4 Atari 1200XL (w 600XL/800XL nie wbudowane).

FB51 64357 ---- ----- KEYDEF

Znajdują się tu wszystkie 129 kody klawiszowe wzgl. ich znaczenia.

FC1A 64538 FFBE 65470 CPIRQQ

Tu leży przerwanie klawiatury. Przy naciśnięciu CONTROL-1 modyfikowana jest flaga STARTSTOP, rejestr ATRACT jest kasowany i SRTIMER jest ustawiany na wcześniej wybraną wartość KEYRPDELY.

FCD6 64726 ---- ----- FREE2

Tu stoją dwa bajty inicjowane przez ?????

FCD8 64728 ---- ----- KEYCLICKC

Stąd startuje bezpośredni skok do procedury KEYCLICK.

FCDB 64731 EF41 61249 INIT

Ustawia POKEY-a na prędkość 600 bitów/sek.

FCE6 64742 EF4C 61260 CASOPEN

Tu jest magnetofon kasetowy przygotowywany dla wejścia lub wyjścia. Jeśli urządzenie jest już otwarte, to meldowany jest błąd.

Gdy błąd nie wystąpi, to zależnie od rozkazu (zapis lub odczyt) wywoływany jest BEEPWAIT dla jednego lub dwóch dźwięków, co tworzy jeden lub dwa dźwięki rozkazowe do obsługi magnetofonu. Następnie po naciśnięciu klawisza włączany jest silnik i krótko oczekuje się, aż osiągnie prawidłową prędkość. Przy zapisie podawany jest ton startowy przez 20 sekund i ustawiane są wskaźniki bufora i długości bufora.

FD7A 64890 EFD6 61398 CASRDBYTE

Tu jest czytany bajt z kasety. Gdy bufor jest pusty, lecz jeszcze nie zakończony zbiór, to wczytywany jest nowy blok danych. Po prawidłowym odczytaniu dana jest przekazywana do akumulatora., a w Y podawany jest status operacji.

FD8D 64909 EFE9 61417 CASREADBL

Pełny blok jest czytany z kasety i sprawdzany, czy nie jest on ostatnim. Jeśli tak, to sprawdzane jest, ile bajtów jest w nim zawartych. Liczba ta jest podawana w CASBUFLIM. Jeżeli odczytany został cały zbiór, następuje raport END_OF_FILE.

FDB4 64948 F010 61456 CASPUTBYT

Znak podany w akumulatorze jest umieszczany w buforze od CASDATA do CASDATA+7F. Jeśli bufor jest pełny (CASBUFPTR wskazuje w buforze na następne wolne miejsce), to automatycznie jest zapisywany i inicjowany na nowo.

FDCC 64972 F028 61480 STATUS

Tu jest sygnalizowany w Y meldunek powodzenia (SUCCESS).

FDCF 64975 F02B 61483 CASCLOSE

Tu wyłączana jest obsługa magnetofonu. Przy czytaniu jest to bardzo proste, przy zapisie trzeba uważać, czy wszystkie bajty zostały zapisane. Po ostatnim bloku zapisany zostaje blok END_OF_TEXT i wyłączany jest silnik.

FDFC 65020 F058 61528 BEEPWAIT

W akumulatorze jest podana liczba dźwięków, które mają być nadane. Każdy dźwięk ma długość ok. 0,25sek, potem przerwa 0,1sek. Po ostatnim dźwięku jest wywoływany KBGETCHAR, a więc oczekuje się na naciśnięcie klawisza.

FE3F 65087 F095 61589 SIOSYSBUF

Tu jest przygotowywany bufor SIO od DSKDEVICE dla operacji kasetowych odczytu lub zapisu. Przykładowo: bufor jest ustawiany na 03FD, a liczba oczekiwanych lub zapisywanych znaków na 131. Wtedy jest przez JUMPTAB+09 wywoływany SIOINTERF.

Każdorazowy kod rozkazu jest podawany w akumulatorze. Gdy SIOSYSBUF jest wywołany rozkazem Write, to prawdopodobnie jest wzywana z WSIOSB.

FE7C 65148 FOD2 61650 WSIOSB

Jest to procedura zapisu na kasecie. W akumulatorze jest podany typ zapisywanych bloków:

FC ten blok składa się ze 128 bajtów.

FA ten blok zawiera mniej niż 128 znaczących bajtów. Dokładna liczba zawarta jest w 128 bajcie bloku. Oznacza to, że odczytuje się wprawdzie 128 bajtów, ale wykorzystuje się ich najwyżej 127.

FE Blok END_OF_FILE, wszystkie 128 bajtów mają wartość zero.

Do tej procedury nie może być podany żaden rozkaz. Jest ona wywoływana tylko w przypadku zapisu.

FE8D 65165 ---- ----- TABELLE

Są tu zawarte następujące pary wartości:

04	03
80	C0
02	01
40	E0
1E	19
0A	08

FE99 65177 EE78 61048 PHINIT

Wartość Timeout dla drukarki(PTIMOUT) jest ustawiona na 30.

FE9F 65183 EE7E 61054 PHSTLO

Tu znajduje się wartość adresowa 02EA (DEVICSTAT) do pośredniego wykorzystania.

FEA1 65185 EE80 61050 PHCHLO

Tu znajduje się wartość adresowa 03C0 (PRINTBUF) do pośredniego wykorzystania.

FEA3 65187 EE81 61057 PHSTAT

Jest przeszukiwane, gdy została wezwana drukarka. Jeśli się nie powiedzie, to ustawiana jest flaga Negative CPU.

FEC2 65218 EE9F 61087 PHOPEN

Po wywołaniu PHSTAT długość bufora drukarki jest ustawiana na zero(PRTBUFPTR). Status z PHSTAT leży jeszcze w rejestrze Y, flagi nie są znaczące.

FECB 65212 EEA7 61095 PHWRITE

Bufor drukarki PRINTBUF jest sukcesywnie wypełniany aż do znaku NEWLINE. Gdy nie jest jeszcze pełny (128 bajtów), uzupełniany jest spacjami. Gdy bufor jest pełny, jest przesyłany do drukarki przez JUMPTAB+09 i SIOINTERF.

FF02 65282 EEB0 61148 PHCLOSE

Po wywołaniu PRMODE jest sprawdzane, czy PRINTBUF jest pusty i jeśli nie, to jest opróżniany przez PHWRITE. PHWRITE nie jest wywoływany na właściwy punkt skoku.

FF0F 65295 EEE6 61158 SETDBC

W X i Y podane są młodszy i starszy bajt adresu bufora z PHCHLO, aby dzięki tym i dalszym zewnętrznie przesłanym wartościom zainicjować SIO dla rozkazu drukowania.

FF3F 65343 EF1A 61210 PHPUT

Wartość z PRTTIME jest przesyłana do PTIMOUT.

FF46 65350 EF1A 61210 PRMODE

Według trybu drukowania (normalny, podwójna szerokość lub wąski druk) są inicjowane DSKCMD i DSKAUX1.

FF6E 65390 ---- ----- CHECKROM1

Oblicza sumę kontrolną następujących obszarów pamięci:

C002–CFFF (C000 i C001 nie, gdyż same tworzą sumę kontr.)

5000–57FF (TEST ROM)

D800–DFFF (ROM matematyczny)

Jeżeli ta obliczona w CHECKSUM (młodszy bajt) i CHECKSUM+! (starszy bajt) suma kontrolna jest identyczna z zaprogramowaną w CHECKSR0 i CHECKSR1 wartością, następuje pozytywny raport przez skasowanie flagi Carry, w przeciwnym razie flaga jest ustawiana. Ta procedura może być wykorzystana do obliczenia wartości przy nowoprogramowanych częściach systemu operacyjnego. Wartości te po obliczeniu są zawsze do dyspozycji w CHECKSUM! Ta i następna funkcja wykorzystują GETCHECKS.

FF8D 65421 ---- ----- CHECHROM2

Ta funkcja, podobnie jak CHECKROM1, oblicza sumę kontrolną obszarów:

E000–FFF7 (FFF8 i FFF9 są sumą kontrolną)

FFFA–FFFF

Obliczona suma jest umieszczana w CHECKSUM2 (młodszy bajt) i CHECKSUM2+1 (starszy bajt). Pozostałe dane jak dla CHECKSUM1.

FFA4 65444 ---- ----- GETCHECKS

Przy wywołaniu tej procedury oczekuje ona w rejestrze X wartości, według której może rozpoznać obszar pamięci, którego suma kontrolna ma być dodana do wartości CHECKSUM (młodszy bajt) i CHECKSUM+1 (starszy bajt), przy czym jest obojętne, czy wybrany został obszar RAM czy ROM. Odpowiednie obszary pamięci określają następujące wartości X:

00 C002–CFFF
04 5000–57FF
08 D800–DFFF
0C E000–EFF7
10 FFFA–FFFF

Jako efekt uboczny procedura daje podwyższenie o 4 zawartości rejestru X. Ułatwia to wielokrotnie wywoływanie tej funkcji do testowania wielu obszarów pamięci.

FFD2 65490 ---- ----- CHKSUMTAB

Są tu wartości początkowe i podwyższone o 1 wartości końcowe dla testu sumy kontrolnej GETCHECKS. Są to następujące wartości:

C002 D000

5000 5800
D800 E000
E000 FFF8
FFFA FFFF

FFF8 65528 ---- ----- CHECKSUM2

Tu i w następnym bajcie znajduje się suma kontrolna chwilowo implementowanych części systemu operacyjnego w obszarach pamięci:

E000 FFF7
FFFA FFFF

Oba bajty sumy kontrolnej FFF8 i FFF9 nie są oczywiście zawarte. Byłoby to oczywiście obliczeniowo możliwe, jednak wadą jest, że GETCHECKS nie umożliwia wtedy obliczenia sumy kontrolnej nowoimplementowanych części systemu operacyjnego.

FFFA 65530 FFFA 65530 NMIVEC

Ten wektor jest ustalonym wektorem mikroprogramu CPU 6502 dla procedury przerwania niemaskowalnego. Wskazuje on na adres PNMI.

FFFC 65532 FFFC 65532 RESETVEC

Ten wektor jest ustalonym wektorem mikroprogramu CPU 6502 dla procedury resetu sprzętowego. Wskazuje on na adres RESETCOLD.

FFFE 65534 FFFE 65534 INTVEC

Ten wektor jest ustalonym wektorem mikroprogramu CPU 6502 dla procedury przerwania maskowanego. Wskazuje on na adres JMPIRQVEC.

P L A N P A M I Ę C I A T A R I

ADRES		NAZWA	OPIS
BIN	HEX		
0	0000	HELPWORD	Te dwa bajty są wykorzystywane przez procedurę resetowania przy teście pamięci.
1	0001		
2	0002	CASINITV	Po zakończeniu ładowania z kasety następuje skok do podanego tu adresu.
3	0003		
4	0004	RAMTSTPTR	Ten wskaźnik jest wykorzystywany tylko do testu wielkości pamięci.
5	0005		
6	0006	TMPRAMSIZ	Jest wykorzystywany w połączeniu z adresem 0005h, aby ustalić wynik testu.
7	0007	TESTDATA	Tu znajduje się bajt danych, który przed rozpoczęciem testu pamięci ustawiany jest w testowanym miejscu.
8	0008	WARMFLAG	Gdy stoi tu wartość różna od zera, to przy naciśnięciu klawisza RESET następuje tylko start gorący.
9	0009	DOSAKTIV	Jeżeli ten bajt ma wartość jeden, to przy gorącym starcie następuje skok do procedury inicjowania DOS w DOSINIT.
10	000A	DOSVEC	Tu znajduje się adres początkowy oprogramowania DOS lub innego oprogramowania przepływu.
11	000B		
12	000C	DOSINIT	Adres skoku do procedury inicjowania DOS.
13	000D		
14	000E	BASMEMTOP	Znajduje się tu najwyższy adres pamięci wykorzystywany przez program użytkownika. Powyżej leży w większości przypadków obszar obrazu.
15	000F		
16	0010	IRQEN$	Przez ustawienie jakiegokolwiek bitu można tu sterować źródłami przerwań POKEY-a. Jeśli bit jest ustawiony, to odpowiednie źródło jest aktywne.
17	0011	IRQST$	Jeżeli jeden z bitów tego rejestru-cienia jest zero, to należące do niego źródło przerwań w POKEY-u jest aktywne i wywołany jest odpowiedni manipulator.
18	0012	CLOCK	Ta trzybajtowa wartość jest zwiększana co 1/50 sekundy, przy czym 0014 jest najmniej znaczącym bajtem.
19	0013		
20	0014		
21	0015	BUFFERADR	Ten adres służy procedurom SIO jako wskaźnik pomocniczy przy operacjach z dyskami.
22	0016		
23	0017	IOCBCMD	Służy jako pamięć pomocnicza przy operacjach CIO
24	0018	DISKFORM	Jest to wskaźnik pomocniczy przy operacjach z dyskami.
25	0019		
26	001A	DISKUTIL	Jest to również wskaźnik pomocniczy przy operacjach z dyskami.
27	001B		
28	001C	ABUFPTR0	Są to wskaźniki pomocnicze wyłącznie do użytku wewnętrznego.
29	001D	ABUFPTR1	
30	001E	ABUFPTR2	
31	001F	ABUFPTR3	
32	0020	IOCBCHIDZ	W tym bajcie znajduje się symbol rozpoznawczy wzywanego urządzenia ze złącza szeregowego.
			Przy obsłudze dysków identyfikacja "dysk" w

33	0021	IOCBDSKNZ	IOCBCHIDZ nie wystarczy, musi być podany jeszcze numer stacji dysków. Podawany jest tutaj.
34	0022	IOCBCMDZ	Tu znajduje się właśnie wykonywany lub właśnie zakończony rozkaz CIO.
35	0023	IOCBSTATZ	Pod tym adresem procedura CIO umieszcza swój meldunek statusu.
36	0024	IOCBBUFAZ	Adres początkowy bufora danych przy operacjach CIO.
37	0025		
38	0026	IOCBPUTBZ	Adres początkowy 1 procedury PUT_ONE_BYTE odpowiedniego urządzenia.
39	0027		
40	0028	IOCBBUFLZ	Długość bufora rozpoczynającego się od IOCBBUFAZ
41	0029		
42	002A	IOCBAUX1	Informacje pomocnicze dla obsługi rozkazu CIO.
43	002B	IOCBAUX2	
44	002C	IOCBAUX3	
45	002D	IOCBAUX4	
46	002E	IOCBNUMZ	Numer wykorzystywanego IOCB pomnożony przez 16.
47	002F	IOCBCHARZ	Rejestr pomocniczy do przyjęcia przesyłanych znaków przy operacjach zapisu CIO bez bufora danych.

Następne rejestry są przeznaczone wyłącznie do użytku wewnętrznego. Można je wykorzystywać tylko wtedy, gdy zmieniana jest wykorzystywana procedura, ponieważ inaczej nie tylko wykorzystywana procedura, lecz też zmienne nie mogą być opisywane przez ich stan.

48	0030	DSKSTAT	Meldunek statusu z obsługi dysku lub kasety.
49	0031	DSKCFIKSUM	Rejestr sumy kontrolnej dla przesyłania szeregowego
50	0032	DSKBUFPTR	początek bufora dla operacji z kasetą lub dyskiem.
51	0033		
52	0034	BUFENDPTR	Wskazuje koniec bufora DSKBUFPTR,
53	0035		
54	0036	LOADERTMP	Rejestr pomocniczy do wewnętrznego użytku Loadera.
55	0037		
56	0038	BUFFULL	Gdy ta flaga jest różna od zera, to bufor CIO lub SIO jest pełny.
57	0039	RECEIVEND	Gdy ta flaga jest różna od zera, to oznacza, że procedura odbioru CIO została zakończona.
58	003A	XMITEND	Ta flaga informuje, czy sterowana przerwaniami procedura nadawania zakończyła już swoje zadanie.
59	003B	CHKSUMSND	Gdy ustawiony, to suma kontrolna została już nadana.
60	003C	NOCHKSUM	Jeżeli ma wartość różną od zera, to suma kontrolna była nadana.
61	003D	CASBUFPTR	Licznik bajtów dla przesyłania z/do kasety. Jego wartości leżą między zero i CASBUFLIM.
62	003E	GAPTYPE	00h: Normalna długość przerw między blokami na kasecie. 90h: Znacznie dłuższa przerwa na początku przesyłania kasetowego.
63	003F	CASEOF	Jeżeli ten bajt jest ustawiony, to podczas odczytu kasety procedura napotkała zapis END_OF_ FILE.
64	0040	BEEPCOUNT	Znajduje się tu parametr dla procedury BEEPWAIT. Zawiera on liczbę tonów jako znak rozpoznawczy wewnętrznej operacji (np. dwa tony: włącz PLAY i

			RECORD).
65	0041	IOSOUNDEN	Gdy ten bajt jest zero to wyjście dźwięku złącza szeregowego jest stłumione.
66	0042	CRITICIO	Ten bit jest ustawiony wtedy, gdy oczekuje się szybkiej serii przerwań (np. przy szeregowym I/O). Sprawia on, że procedura przerwania wygaszania pionowego jest wykonywana tylko w nieznacznej części, aby przez to zaoszczędzić czas. Jako efekt uboczny otrzymuje się np., że z powodu błędnych wyników liczników nieskuteczne jest automatyczne powtarzanie klawiatury.
67 do 72	0043 do 0048	FILEMNGMT	Wewnętrzny wskaźnik do sterowania dyskami.
73 74 75	0049 004A 004B	ZCHAIN	Wskaźnik dla przetwarzania liniowej listy IOCB.
76	004C	MONSTATUS	Ten status jest konieczny do obsługi monitora.
77	004D	ATTRACT	Jeżeli ten bajt jest mniejszy niż 128 (80h), to następuje normalne wyjście obrazu. Gdy osiągnie wartość 128, jest ustawiany na 254 i powoduje włączenie trybu Attract.
78	004E	ATRACTMSK	FEh: normalna jasność ekranu F6h: zmniejszona jasność AMRACTMSK jest zależny od ATTRACT.
79	004F	COLREGSH	Tak jak ATRACTMSK ten rejestr należy również do obsługi trybu Attract. Następuje tu logiczne połączenie rejestrów koloru i jasności ANTIC-u.
80	0050	MONTEMP	Bajty pomocnicze do przetwarzania obrazu.
81	0051		
82	0052	LEFTMARGIN	Szerokość lewego marginesu przy pisaniu tekstu.
83	0053	RIGMARGIN	Szerokość prawego marginesu przy pisaniu tekstu.
84	0054	CURSROW	Aktualna linia kursora przy obsłudze grafiki w zakresie od 0 do 191.
85	0055	CURSCOL	Aktualna linia kursora przy obsłudze grafiki w zakresie od 0 do 319.
86	0056		
87	0057	GRAPHEMUL	Ta wartość ustala, w którym trybie graficznym odbywa się następne wyjście. Jest to konieczne, aby wyższe programowania mogły łatwo mieszać rodzaje grafiki.
88	0058	SCRNSTART	Adres pierwszego bajta obrazu.
89	0059	OLDGRROW	Te adresy zawierają wszystkie dane, które muszą być przechowane, gdy program przełącza między obsługą grafiki i tekstu. Adresy te są określone tylko do celów wewnętrznych!
90	005A		
91	005B	OLDGRCOL	
92	005C		
93	005D	OLDGRGHR	
94	005E	OLDGRADR	
95	005F		
96	0060	FKTDEFPTR	Adres początkowy 8-bajtowej tabeli do ustalenia kodów klawiszy funkcyjnych 1200XL.
97	0061		
95	0062	PALNTSC	0: system PAL, 1: system NTSC.
99	0063	ACTCURNUM	Aktywna pozycja kursora wewnątrz linii log.
100	0064	ADDRESS	Często wykorzystywane wewnętrznie rejestry do różnorakich zastosowań.
101	0065		
102	0066	MLTTMP	
103	0067		
104	0068	SAVEADR	

105	0069		
106	006A	RAMTOP	Liczba stron RAM do dyspozycji.
107	006B	ACTBUFLEN	Chwilowa wielkość aktualnej linii logicznej
108	006C	BUFSTR	Wskaźnik GETCHARACTER edytora.
109	006D		
110	006E	BITMASK	Rejestr do zarządzania linią logiczną.
111	006F	SHFAMT	Rej. pomocniczy konieczny też dla GETCHAR.
112	0070	PLOTROWAC	Te rejestry razem z rejestrami ROWINC i COLINC są wykorzystywane do obliczania grafiki (np. DRAWTO).
113	0071		
114	0072	PLOTCOLAC	
115	0073		
116	0074	ENDPOINTER	
117	0075		
118	0076	DELTAROW	
119	0077	DELTACOL	
120	0078		
121	0079	KEYDEPTR	Ten wskaźnik wskazuje na tabelę do zamiany kodu klawiatury na ATASCII.
122	007A		
123	007B	SWAPFLAG	Ta flaga różna od zera wskazuje, że obszar zmiennych obrazu jest włączony na grafikę, zero sygnalizuje, że zmienne dla obszaru obrazu tworzą tekst.
124	007C	HOLDCHAR	Wewnętrzne zmienne pomocnicze dla zarządzania obrazem.
125	007D	INSDATA	
126	007E	COUNTER	
127	007F		
128	0080	LOWMEM	Wskaźnik najniższego adresu, który nie jest już wykorzystywany do celów systemu operacyjnego.
129	0081		
139	008B	CHECKSUM	W tych dwóch bajtach jest tworzona suma kontrolna obszaru pamięci. Ponieważ jest ona konieczna tylko w fazie inicjowania, po zimnym starcie może być ta wartość zniszczona.
140	008C		
512	0200	DLIVKT	Wektor dla przerwania programu ANTIC-u. Gdy przerwanie programu ANTIC-u jest wezwane, przeprowadzany jest program przerwania, którego adres stoi w tych dwóch komórkach pamięci.
513	0201		
514	0202	VPRECEDE	Wektor skoku dla przerwania wezwanego z PORT A, który tworzy przerwanie SERIAL_BUS_PROCEED.
515	0203		
516	0204	VINTERRUP	PORT B sygnalizuje żądanie przerwania przesyłania szeregowego. Oba ostatnie wektory przerwań są normalnie ignorowane przez system, ponieważ zamknięte urządzenie nie tworzy przerwań. Linie są planowane dla późniejszych rozszerzeń.
517	0205		
518	0206	VBREAK	Adres skoku wykorzystywany, gdy CPU przeprowadza instrukcję BRK.
519	0207		
520	0208	VKEYBOARD	Wektor dla przypadku, gdy został naciśnięty normalny klawisz.
521	0209		
522	020A	VSERIELIN	Umieszcza znak ze złącza szeregowego w SERIN.
523	020B		
524	020C	VSERREADY	Rejestr SEROUT jest gotowy do przyjęcia następnego znaku.
525	020D		
526	020E	VSERCLOSE	Rejestr równoległo-szeregowy jest pusty i przesyłanie przez POKEY zostało zakończone.
527	020F		
528	0210	VTIMER1	Gdy POKEY sygnalizuje, że licznik 1 jest zero, wykonywany jest skok do tego wektora.
529	0211		
530	0212	VTIMER2	Gdy POKEY sygnalizuje, że licznik 2 jest zero, wykonywany jest skok do tego wektora.
531	0213		

532	0214	VTIMER4	Gdy POKEY sygnalizuje, że licznik 1 jest zero, a odpowiednie przerwanie ma zezwolenie, wykonywany jest skok do tego wektora.
533	0215		
534	0216	VIMMEDIRQ	Adres właściwego manipulatora przerwania, który najpierw podejmuje podział na poszczególne źródła przerwań.
535	0217		
536	0218	TIMCOUNT1	Zegar programowy nr 1. Wartość jest zmniejszana przy każdym przerwaniu wygaszania pionowego. Jeżeli jest ona po tym zero, to wywołuje się procedurę stojącą w (TIMER1VEC).
537	0219		
538	021A	TIMCOUNT2	Podobny zegar programowy jak TIMCOUNT1, z tą tylko różnicą, że zliczanie w tym i trzech następnych licznikach nie zachodzi, gdy CRITICIO jest ustawiony.
539	021B		
540	021C	TIMCOUNT3	Dla tego licznika obowiązuje wszystko to, co dla TIMCOUNT2, tylko że przy wartości zero nie jest wykonywany skok do żadnego wektora, lecz jedynie ustawiana jest flaga TIMER3SIG.
541	021D		
542	021E	TIMCOUNT4	Dla tego licznika obowiązuje wszystko to, co dla TIMCOUNT3, tylko że przy wartości zero nie jest wykonywany skok do żadnego wektora, lecz jedynie ustawiana jest flaga TIMER4SIG.
543	021F		
544	0220	TIMCOUNT5	Dla tego licznika obowiązuje wszystko to, co dla TIMCOUNT3, tylko że przy wartości zero nie jest wykonywany skok do żadnego wektora, lecz jedynie ustawiana jest flaga TIMER5SIG.
545	0221		
546	0222	VBLKIVEC	Do tego wektora zawsze przy przerwaniu wygaszania pionowego jest wykonywany pierwszy skok.
547	0223		
548	0224	VBLKDVEC	Skok do tego wektora następuje z procedury wskazanej przez VBLKIVEC, gdy CRITICIO jest skasowany. Ta procedura normalnie nie istnieje i jest programowana przez użytkownika.
549	0225		
550	0226	TIMER1VEC	Wektor skoku dla procedury użytkownika, która może być przeprowadzona, gdy TIMCOUNT1 jest zero.
551	0227		
552	0228	TIMER2VEC	Wektor skoku do procedury obsługi przy zerowej wartości TIMCOUNT2.
553	0229		
554	022A	TIMER3SIG	Ta flaga jest zero, gdy TIMCOUNT3 jest różny od zera, w przeciwnym razie jest FF.
555	022B	SRTIMER	Istotny, wewnętrznie wykorzystywany zegar do sprawdzania klawiatury, czekania do pierwszego automatycznego powtórzenia i określenia potem prędkości powtarzania.
556	022C	TIMER4SIG	Ta flaga jest zero, gdy TIMCOUNT4 jest różny od zera, w przeciwnym razie jest FF.
557	022D	IANTEMP	Przejściowy Rejestr pomocniczy.
558	022E	TIMER5SIG	Ta flaga jest zero, gdy TIMCOUNT5 jest różny od zera, w przeciwnym razie jest FF.
559	022F	DMACNTL$	Rejestr-cień DMACNTL. DMACNTL włącza poszczególne rodzaje DMA.
560	0230	DLPTR$	Rejestr-cień wskaźnika programu ANTIC-u. Początek programu ANTIC-u musi się znajdować we wskazanym miejscu, gdy jest przetwarzany przez system operacyjny.
561	0231		
562	0232	SKCNTL$	Sterowanie przesyłaniem szeregowym, odczytem klawiatury i odczytem wejść potencjometrów,
563	0233	LCOUNT	Licznik bajtów dla procedury Loader.
564	0234	LPENH$	Rejestr-cień. Pozioma pozycja pióra świetlnego.
565	0235	LPENV$	Rejestr-cień. Pionowa pozycja pióra świetlnego.

566	0236	VBREAKKEY	Wektor do obsługi klawisza BREAK. Nie pomylić z VBREAK, który obsługuje instrukcję BRK CPU!
567	0237		
568	0238	NEWIOINIV	Wektor do procedury inicjowania jeszcze nie istniejącego sprzętu. Zobacz w tym celu opis systemu operacyjnego!
569	0239		
570	023A	CMDDEVIV	Wewnętrzne zmienne pomocnicze dla obsługi dysków.
571	023B	CMDCMD	
572	023C	CMDAUX1	
573	023D	CMDAUX2	
574	023E	CMDAUX3	
575	023F	ERRORFLAG	
576	0240	DSKFLAG	
577	0241	DSKSECCNT	
578	0242	DSKLDADR	
579	0243		
580	0244	COLDSTART	Jeżeli ta flaga jest ustawiona na wartość różną od zera, to przy skoku do procedury gorącego startu następuje start zimny.
582	0246	DISKTIMCON	Rejestr kontrolny dla obsługi dysków, wartość różna od zera mówi, że wystąpił Timeout.
583	0247	NEWVORHDN	Flaga tylko do użytku wewnętrznego. Jest ustawiana, gdy do szyny równoległej jest dołączone zewnętrzne urządzenie. Normalnie musi być zero, ponieważ inaczej system może runąć!
584	0248	NEWIODREQ	Ta flaga zawiera adres właśnie wywołanego nowego urządzenia.
585	0249	NEWIOMASK	Te adresy także są wykorzystywane sensownie tylko wtedy, gdy dodatkowy sprzęt jest dołączony do szyny równoległej.
586	024A	NEWLDTMP1	
587	024B		
619	026B	CHARSTPTR	Wskaźnik dla wewnętrznej obsługi monitora.
620	026C	FINESCROL	Przejściowa flaga do obsługi delikatnego przesuwu, przy czym nie jest równa przesunięciu pełnego wiersza.
621	026D	KBDISABLE	Gdy ta flaga ma wartość różną od zera, to wejście z klawiatury jest niemożliwe.
622	026E	FINESCRFL	Odpowiedzialny za przesuw obrazu jak FINESCROL.
623	026F	GTIACNTL$	Rejestr-cień do sterowania GTIA i priorytetu graczy i pocisków.
624	0270	PADDL0$	Rejestry-cienie, zawierają wartości wejść Paddle. W Atari 600XL/800XL wykorzystane tylko wejścia 0 - 3, 4 - 7 są kopiowane z 0 - 3.
625	0271	PADDL1$	
626	0272	PADDL2$	
627	0273	PADDL3$	
628	0274	PADDL4$	
629	0275	PADDL5$	
630	0276	PADDL6$	
631	0277	PADDL7$	
632	0278	JOYSTICK0	Te rejestry zawierają stan dżojstików. W nowych modelach Atari są przewidziane tylko dwa dżojstiki (0 i 1), pozostałe są kopiowane.
633	0279	JOYSTICK1	
634	027A	JOYSTICK2	
635	027B	JOYSTICK3	
636	027C	PTRIG0$	Przyciski paddle ("0"-przycisk naciśnięty, "1" - zwolniony). W Atari 600XL/800XL można przyłączyć tylko 4 paddle, więc tylko 0 - 3 są wykorzystane, a pozostałe są kopiowane.
637	027D	PTRIG1$	
638	027E	PTRIG2$	
639	027F	PTRIG3$	
640	0280	PTRIG4$	
641	0281	PTRIG5$	
642	0282	PTRIG6$	
643	0283	PTRIG7$	

644	0284	TRIG0$	Przyciski dżojstików ("0" - przycisk naciśnięty, "1" - zwolniony). W nowych modelach Atari tylko TRIG0 i TRIG1 połączone z portami dżojstików, wartości pozostałych są kopiowane.
645	0285	TRIG1$	
646	0286	TRIG2$	
647	0287	TRIG3$	
648	0288	HIBYTELD	Rejestr pośredni dla Loadera.
649	0289	WRITE MODE	Flaga dla obsługi kasety. Wskazuje, czy aktualna operacja to zapis (80h) czy odczyt (00h). Tylko do użytku wewnętrznego.
650	028A	CASBUFLIM	Adres końca bufora dla operacji kasetowych.
651	028B		
652	028C	NEWIOPTR	Wskaźnik adresu początkowego obsługi I/O nowego sprzętu na szynie równoległej.
653	028D		
654	028E	NEWADRLOD	Adres ładowania dla I/0 z nowym sprzętem dodatkowym.
655	028F		
656	0290	TEXTROW	Zmienne do obsługi mieszanej tekstowo-graficznej zawartości obrazu.
657	0291	TEXTCOL	
658	0292		
659	0293	TEXTINDEX	
660	0294	TEXTMSC	
661	0295		
662	0296	TEXTOLD	
663	0297		
666	029A	TEXTGRAD	
667	029C	CRETRY	Maksymalna liczba powtórzeń prób podania rozkazu na urządzenie, które melduje się z powrotem. Jeżeli wartość jest jeden, to me miejsce jedno powtórzenie.
668	029E	SUBTEMP	Zmienne pomocnicze do użytku wewnętrznego.
669	029F	HOLD2	
670	02A0	DISPLYMSK	Zmienne do przetwarzania tekstu przez obraz lub edytor.
671	02A1	TEMPLBT	
672	02A2	ESCAPEFLAG	
673 do 689	02A3 do 02B1	TABMAP	Tabela, w której każdy pojedynczy bit tworzy pozycję kursora wewnątrz linii logicznej na ekranie. Jeżeli bit jest ustawiony, to ta pozycja jest miejscem tabulacji.
690 do 693	02B2 do 02B5	LOGICMASK	Tabela do łączenia linii logicznej i fizycznej na aktualnie wyświetlanym obrazie. Jest konieczna wewnętrznie, by przy kasowaniu linii zawsze była kasowana pełna linia logiczna.
694	02B6	XORKEYMSK	Ta zmienna jest używana, aby umożliwić zmianę funkcji grup klawiszy klawiatury. Każdy ustawiony bit w tym bajcie podaje, że dostarczany z POKEY-a bit danych z kodu klawiatury musi być odwrócony. Można tak cały rząd klawiszy zamienić na inny. Należy jednak zauważyć, że odwrócenie bitu zachodzi w kodzie klawiatury, a więc przed zamianą odczytanej wartości na kod ATASCII.
701	02BD	DRETRY	Liczba, od której są powtarzane próby wezwania określonego urządzenia. Normalnie 13 powtórzeń.
702	02BE	SHIFTLOCK	00h: Aktywne małe litery (tryb maszyny do pisania), 40h: tylko duże litery (tryb normalny), 80h: naciśnięty klawisz CONTROL.
703	02BF	NUMNXTLIN	Ten rejestr zawiera liczbę linii tekstu przetwarzanych na ekranie. Może mieć wartości 24, 4 lub 0, inne są ignorowane przez system operacyjny.
704	02C0	COLPM0$	

705	02C1	COLPM1$	Rejestry-cienie rejestrów koloru dla graczy względnie pocisków.
706	02C2	COLPM2$	
707	02C3	COLPM3$	
708	02C4	COLPF0$	Rejestry-cienie rejestrów koloru dla pola gry względnie tła.
709	02C5	COLPF1$	
710	02C6	COLPF2$	
711	02C7	COLPF3$	
712	02C8	COLBAK$	
713	02C9	RUNADRLOD	Te zmienne są wykorzystywane do wewnętrznych zadań przy ładowaniu programów późniejszego sprzętu dołączanego do szyny równoległej.
714	02CA		
715	02CB	HIUSEDLOD	
716	02CC		
717	02CD	LODZHIUSE	
718	02CE		
719	02CF	LODGBYTEA	
720	02D0		
721	02D1	LODADRESS	
722	02D2		
723	02D3	LODZLOADA	
724	02D4		
725	02D5	DSKECLEN	Ta wartość może być normalnie 0080h lub 0100h, zależnie od formatu dysku.
726	02D6		
727	02D7	ACMISR	Ma wewnętrzne zastosowanie.
728	02D8		
729	02D9	KEYRPDELY	Ta wartość określa czas między naciśnięciem klawisza a pierwszym automatycznym powtórzeniem. Wartość standardowa jest 40.
730	02DA	KEYREP	Ta wartość, w przeciwieństwie do KEYRPDELY, określa czas między kolejnymi automatycznymi powtórzeniami. Wartość standardowa jest 5.
731	02DB	CLICKDISA	Gdy różny od zera, to "klick" klawiatury jest wyłączony.
732	02DC	HELPFLAG	Ten rejestr wskazuje, czy był naciśnięty klawisz HELP. Po puszczeniu naciśniętego klawisza rejestr nie zmienia ponownie zawartości!
733	02DD	DMASAVE	Informacja o statusie bezpośredniego dostępu do pamięci. Normalnie nie ma zastosowania dla użytkownika.
734	02DE	PRTBUFPTR	Numer bajta wewnątrz bufora drukarki.
735	02DF	PRTBUFSIZ	Maksymalna wielkość bufora drukarki w bajtach.
740	02E4	RAMSIZE	Liczba będących do dyspozycji stron RAM systemu.
741	02E5	MEMTOP	Wskaźnik górnej granicy przestrzeni pamięci wykorzystywanej przez użytkownika. Normalnie powyżej leży obraz.
742	02E6		
743	02E7	MEMLO	Wskaźnik dolnej granicy obszaru pamięci wykorzystywanej przez użytkownika. Normalnie leży on bezpośrednio po zmiennych systemu operacyjnego względnie DOS, ewentualnie jeszcze DUP.
744	02E8		
745	02E9	HNDLRLOAD	Pomocnicze zmienne do wewnętrznego użytku przy obsłudze listy liniowej i biegnących przez nią przejść ładowania jeszcze nie istniejącego sprzętu.
746	02EA	DEVICSTAT	
747	02EB		
748	02EC	CHAINTMP	
749	02ED		
750	02EE	CASSPEED	Wartość czasowa dla okresu nadawania na kasetę.
751	02EF		
752	02F0	CURSORINH	Jeśli wartość tej zmiennej jest różna od zera, to kursor nie jest wyświetlany na ekranie.

753	02F1	KEYDELAY	Ta wartość, standardowo ustawiona na 3, określa czas między puszczeniem klawisza i naciśnięciem nowego klawisza, tak żeby jego kod był przyjęty. Przy bardzo szybkim pisaniu wartość ta może być zmniejszona.
755	02F3	CHARCNTL$	Rejestr-cień, steruje wyglądem znaków w więk-szości trybów tekstowych ANTIC-u.
756	02F4	CHARBASE$	Rejestr-cień, zawiera starszy bajt adresu bazowego generatora znaków.
757	02F5	NEWGRROW	Wartości dla rozkazu DRAWTO.
758	02F6	NEWGRCOL	
759	02F7		
760	02F8	ROWINC	Zmienne wykorzystywane do obliczeń trasy DRAWTO.
761	02F9	COLINC	
762	02FA		
763	02FB	ATASCICHR	Ostatni podany znak w kodzie ATASCII.
764	02FC	KBCODE$	Rejestr-cień, zawiera kod klawiatury, tzn., który znak był naciśnięty.
765	02FD	FILEDAT	Zarezerwowany do wewnętrznego wykorzystania systemów graficznych.
766	02FE	DISPLYFLG	Jeżeli ta flaga jest różna od zera, to następny znak, gdy jest on znakiem kontroli, nie jest przeprowadzany, lecz zamieniany na odpowiedni kod ATASCII.
767	02FF	STARTSTOP	Jeżeli ten bajt jest równy FFh, to zatrzymuje wyjście obrazu, przy 00h idzie ono dalej. Flaga ta jest zmieniana przez naciśnięcie CONTROL 1. Ważne jest, że przy aktualnie implementowanym systemie operacyjnym system "stoi", gdy oczekuje na wartość 0 w STARTSTOP. Przeprowadzane są tylko istotne przerwania, właściwa praca jest jednak zatrzymana.

Poniżej są opisane ważne zmienne wykorzystywane przy obsłudze dysków. Są one przeważnie same ustawiane przy obsłudze dysków, gdy programowanie następuje w językach wyższego poziomu, jak na przykład BASIC lub PASCAL.

768	0300	DSKDEVICE	Ogólny kod rozpoznawczy stacji dysków (30h) względnie magnetofonu kasetowego (E0h).
769	0301	DSKUNIT	Numer wezwanej stacji dysków, może mieć wartość z zakresu od 01h do 09h, lecz obecny sprzęt rozpoznaje jedynie wartości od 01h do 04h.
770	0302	DSKCMD	Przeprowadzany rozkaz. Zobacz IOCB na stronie zerowej od IOCBCHIDZ.
771	0303	DSKSTATUS	Zameldowany z powrotem status operacji. Zobacz także IOCDSTATZ.
772	0304	DSKBUFFER	Adres początkowy bufora danych.
773	0305		
774	0306	DSKTIMOUT	Liczba sekund do meldunku błędu Timeout.
775	0307		
776	0308	DSKBYTCNT	Liczba bajtów będących do dyspozycji w buforze od DSKBUFFER.
777	0309		
778	030A	DSKAUX1	Pomocnicze informacje do obsługi dysków. Rozkaz OPEN ma tu cechy szczególne (np. dostęp tylko do odczytu). Przy operacjach zapisu lub odczytu znajduje się tu numer bloku od 01h do MAXBLOCK.
779	030B	DSKAUX2	
780	030C	INTERVTI1	Zegar odstępów nr 1, tworzy jedną jednostkę z INTERVTI2.
781	030D		
			Ten adres jest zaznaczony oficjalnie przez Atari

782	030E	OPTIONJMP	jako istniejący, nie jest jednak wykorzystany nigdzie w systemie operacyjnym. Przewidziany jest dla późniejszego sprzętu. Ze względu na późniejszą kompatybilność oprogramowania, nie należy tej zmiennej wykorzystywać.
783	030F	CASFLAG	Gdy ta flaga nie jest zero, to chodzi o operację dla kasety. Tylko do użytku wewnętrznego.
784	0310	INTERVTI2	INTERVTI1 i INTERVTI2 są wykorzystywane, aby określić prędkość odczytu z kasety.
785	0311		
786	0312	CHAINTP1	Wskaźnik pomocniczy do obsługi list liniowych.
787	0313		
788	0314	PTIMOUT	Wartość Timeout dla drukarki.
791	0317	TIMEFLAG	Wartość Timeout dla określenia prędkości odczytu
792	0318	STACKSAVE	Rejestr pomocniczy dla zachowania wskaźnika stosu przy operacjach SIO.
793	0319	TEMPSTAT	Rejestr do krótkotrwałego zapamiętywania informacji statusu SIO.
794 do 831	031A do 033F	HATABS	Tabela do przyporządkowania urządzeń ich manipulatorom. Każdy wpis tabeli składa się z trzech bajtów nazwy (1) i adresu początkowego tabeli manipulatora (2). Przewidziane są urządzenia: kaseta, edytor, ekran, drukarka i klawiatura. Wpis jest pusty wtedy, gdy jego nazwa jest zero.
832 do 847	0340 do 034F	IOCB0	Od tego adresu znajduje się 8 wpisów IOCB, z których wpis 0 jest wykorzystywany przez system do obsługi edytora. Poszczególne wpisy proszę zobaczyć w opisie IOCB strony zerowej, który znajduje się od IOCBCHIDZ. Przy wykorzystaniu jednego z tych IOCB zawartość jego rejestrów jest kopiowana na stronie zerowej i potem znowu przenoszona z powrotem.
848 do 863	0350 do 035F	IOCB1	
864	0360	IOCB2	
do			
879	036F		
880	0370	IOCB3	
do	do		
895	037F		
896	0380	IOCB4	
do	do		
911	038F		
912	0390	IOCB5	
do	do		
927	039F		
928	03A0	IOCB6	
do	do		
943	03AF		
944	03B0	IOCB7	
do	do		
959	03BF		
960 do 999	03C0 do 03E7	PRINTBUF	Znajduje się tu bufor drukarki do wyjścia na drukarkę przez złącze szeregowe.
1000	03E8	SUPERFLAG	Informacje pomocnicze dla edytora.
1001	03E9	STARTTST	Ta flaga ma wartość różną od zera, gdy podczas

			zimnego startu naciśnięty był klawisz START.
1002	03EA	CASSTART	Flaga, czy właśnie jest przepływ z/do kasety i czy do podanego tam adresu początkowego ma być wykonany skok.
1003	03EB	CARTCKSUM	Suma kontrolna kartridża w obszarze C000-CFFF.
1004	03EC		
1005 do 1015	03ED do 03F7	ACMVAR	Zmienna pomocnicza do celów wewnętrznych.
1016	03F8	X64KBFLAG	Flaga, czy najwyższe 16KB jest RAM czy ROM.
1017	03F9	MINTLK	Rejestr pomocniczy, nie wykorzystywać.
1018	03FA	TRIG3$	Jest to kopia TRIG3, na którą normalnie wpływa tylko wyjście RD5. Jeżeli kartridż jest włożony (01h) lub wyjęty (00h), to wartość TRIG3 zmienia się. Jest wykorzystywany do ustalenia, czy chodzi o start zimny czy gorący.
1019	03FB	CHAINLINK	Wskaźnik przetwarzania list liniowych. Gdy jest ustawiony, a więc tylko przy specjalnych zastosowaniach dla jeszcze nie istniejącego sprzętu.
1020	03FC		
1021 do 1151	03FD do 047F	CASBUFFER	Obszar bufora dla przesyłania danych z/do kasety, przy czym trzy pierwsze bajty służą tylko do synchronizacji i ustalenia typu i ilości danych. Właściwy bufor danych rozpoczyna się od adresu 1024 (04C0h) i rozciąga się do końca CASBUFFER.

W tym miejscu kończą się zmienne systemu operacyjnego. Z pewnością są jeszcze wykorzystywane różne inne miejsca pamięci na stronie zerowej i w obszarze do 0480h. Ponieważ jednak ich użycie jest ograniczone jedynie do BASIC-u, DOS-u i innego oprogramowania, nie będzie to tutaj zawarte.

Dalej następują adresy sprzętowe GTIA, POKEY-a, PIA i ANTIC-u:

W - znaczenie przy zapisie R - znaczenie przy odczycie

53248	D000	HPOSP0	W	Poziome pozycje graczy.
53249	D001	HPOSP1	W	
53250	D002	HPOSP2	W	
53251	D003	HPOSP3	W	
53252	D004	HPOSM0	W	Poziome pozycje pocisków.
53253	D005	HPOSM1	W	
53254	D006	HPOSM2	W	
53255	D007	HPOSM3	W	
53248	D000	KOLM0PF	R	Rejestry kolizji pocisk/pole gry.
53249	D001	KOLM1PF	R	
53250	D002	KOLM2PF	R	
53251	D003	KOLM3PF	R	
53252	D004	KOLP0PF	R	Rejestry kolizji gracz/pole gry.
53253	D005	KOLP1PF	R	
53254	D006	KOLP2PF	R	
53255	D007	KOLP3PF	R	
53256	D008	SIZEP0	W	Rejestry wielkości graczy.
53257	D009	SIZEP1	W	
53258	D00A	SIZEP2	W	
53259	D00B	SIZEP3	W	
53260	D00C	SIZEM	W	Rejestr wielkości pocisków.
53256	D008	KOLM0PL	R	Rejestry kolizji graczy/pocisków.
53257	D009	KOLM1PL	R	
53258	D00A	KOLM2PL	R	
53259	D00B	KOLM3PL	R	
53261	D00D	GRAFP0	W	

53262	D00E	GRAFP1	W	Rejestry graficzne graczy.
53263	D00F	GRAFP2	W	
53264	D010	GRAFP3	W	
53265	D011	GRAFM	W	Rejestr graficzny pocisków.
53260	D00C	KOLP0PL	R	Rejestry kolizji pomiędzy graczami.
53261	D00D	KOLP1PL	R	
53262	D00E	KOLP2PL	R	
53263	D00F	KOLP3PL	R	
53264	D010	TRIG0	R	Rejestry przycisków dżojstików.
53265	D011	TRIG1	R	
53266	D012	TRIG2	R	
53267	D013	TRIG3	R	
53266	D012	COLPM0	W	Rejestry koloru graczy i pocisków.
53267	D013	COLPM1	W	
53268	D014	COLPM2	W	
53269	D015	COLPM3	W	
53270	D016	COLPF0	W	Rejestry koloru pola gry.
53271	D017	COLPF1	W	
53272	D018	COLPF2	W	
53273	D019	COLPF3	W	
53274	D01A	COLBAK	W	Rejestr koloru tła.
53275	D01B	GTIACNTL	W	Steruje GTIA, określa kolejność priorytetów między elementami obrazu i steruje tworzeniem graczy i pocisków.
53276	D01C	VDELAY	W	Umożliwia przesuwanie graczy i pocisków o pojedyncze linie obrazu, gdy jest wybrana dwuwierszowa rozdzielczość PMG.
53277	D01D	PMCNTL	W	Włącza graczy i pociski; umożliwia zatrzymanie statusu wejść przycisków.
53278	D01E	HITCLR	W	Jakikolwiek zapis do tego rejestru kasuje rejestry kolizji.
53279	D01F	CONSOL	RW	Rejestr statusu klawiszy konsoli START, SELECT OPTION
53759	D1FF	NEWPORT	Ten rejestr służy rozszerzeniom I/O. Zobacz opis systemu operacyjnego!	
53760	D200	AUDIFREQ1	W	Rejestry te określają częstotliwość generowanego dźwięku
53761	D201	AUDIFREQ2	W	
53762	D202	AUDIFREQ3	W	
53763	D203	AUDIFREQ4	W	
53764	D204	AUDICNTL1	W	Te rejestry sterują poszczególnymi generatorami dźwięku. Określają one głośność, zniekształcenia względnie tryb VOLUME_ONLY.
53765	D205	AUDICNTL2	W	
53766	D206	AUDICNTL3	W	
53767	D207	AUDICNTL4	W	
53760	D200	POT0	R	Te rejestry zawierają wartości wejść potencjometrów POKEY-a. W 600XL/800XL są wykorzystane tylko wejścia 0-3. Pozostałe rejestry zawierają kopie pierwszych czterech.
53761	D201	POT1	R	
53762	D202	POT2	R	
53763	D203	POT3	R	
53764	D204	POT4	R	
53765	D205	POT5	R	
53766	D206	POT6	R	
53767	D207	POT7	R	
53768	D208	AUDICOM	W	Rejestr sterujący tworzeniem dźwięków.
53768	D208	POTSTAT	R	Dzięki temu rejestrowi można ustalić, czy dla określonego wejścia potencjometru została zakończone przetwarzanie analogowo-cyfrowe. Gdy bit POTSTAT jest "0", oznacza to, że wartość należącego do niego wejścia POT jest ważna. Wejścia POT są przyporządkowane bitom

				w ten sposób, że numer bitu jest taki sam jak numer wejścia potencjometru.
53769	D209	STIMER	W	Przez wybranie tego adresu w trybie zapisu dzielniki częstotliwości są ponownie ustawiane na wartość AUDIFREQ.
53769	D209	KBCODE	R	Ten rejestr zawiera kod klawiatury, tzn. użytkownik może tu sprawdzić, który klawisz jest naciśnięty.
53770	D20A	SKSTATRES	W	Przez wpisanie dowolnej wartości do tego rejestru są kasowane bity 5, 6 i 7 SKSTAT.
53770	D20A	RANDOM	R	Ten rejestr zawiera 8-bitową liczbę pseudo-losową, która zgadza się z licznikiem 17- wzgl. 9-bitowym.
53771	D20B	POTGO	W	Wpisanie do tego rejestru dowolnej wartości uruchamia odczyt wejść potencjometrów.
53773	D20D	SEROUT	W	Rejestr wyjściowy złącza szeregowego.
53773	D20D	SERIN	R	Rejestr wejściowy złącza szeregowego.
53774	D20E	IRQEN	W	Bity w tym rejestrze włączają i wyłączają poszczególne rodzaje przerwań.
53774	D20E	IRQSTAT	R	Dzięki temu rejestrowi można określić pochodzenie żądania przerwania maskowalnego.
53775	D20F	SKCNTL	W	Poprzez ten rejestr można wybierać poszczególne tryby pracy złącza szeregowego, zamiany wartości POT i sprawdzania klawiatury.
53775	D20F	SKSTAT	R	Status złącza szeregowego i klawiatury.
54016	D300	PORT A	RW	Port A układu PIA
54017	D301	PORT B	RW	Port B układu PIA
54018	D302	PORTACNTL	Rejestr sterujący i statusu portu A	
54019	D303	PORTBCNTL	Rejestr sterujący i statusu portu A	
54272	D400	DMACNTL	W	Sterowanie DMA ANTIC-u.
54273	D401	CHARCNTL	W	Sterowanie wyglądem znaków w trybach ANTIC-u.
54274	D402	DLPTRL	16-bitowy adres programu ANTIC-u.	
54275	D403	DLPTRH		
54276	D404	HSCROL	W	Ten rejestr określa, o ile cykli koloru ma być przesunięta określona część obrazu.
54277	D405	VSCROL	W	Ten rejestr określa, o ile linii obrazu ma być przesunięta określona część obrazu.
54279	D407	PMBASE	W	Starszy bajt adresu bazowego obszaru pamięci dla PMG
54281	D409	CHARBASE	W	Starszy bajt adresu bazowego generatora znaków.
54282	D40A	WAITHSYNC	W	Wezwanie tego rejestru zatrzymuje CPU do początku następnej synchronizacji poziomej.
54283	D40B	VCOUNT	R	Numer właśnie tworzonej linii obrazu podzielony przez 2.
54284	D40C	LPENH	R	Pozioma pozycja pióra świetlnego.
54285	D40D	LPENV	R	Pionowa pozycja pióra świetlnego.
54286	D40E	NMIEN	W	Włączanie i wyłączanie przerwań przy synchronizacji pionowej i przerwań programu ANTIC-u.
54287	D40F	NMIST	R	Dzięki temu rejestrowi można określić pochodzenie żądania przerwania niemaskowalnego.
54287	D40F	NMIRES	W	Dowolny zapis powoduje skasowanie NMIST.
65530	FFFA	NMIVEC	Po przerwaniu niemaskowalnym CPU wykonuje skok do znajdującego się tu adresu.	
65531	FFFB			
65532	FFFC	RESETVEC	Po RESET CPU wykonuje skok do znajdującego się tu adresu.	
65533	FFFD			
65534	FFFE	IRQVEC	Po przerwaniu maskowalnym CPU wykonuje skok do znajdującego się tu adresu.	
65535	FFFF			

LISTA ETYKIET

Etykieta	Adres (hex)	Etykieta	Adres (hex)
ABUFPTR1	001D	CASDATA	0400
ABUFPTR2	001E	CASENTER	EB9D
ABUFPTR3	001F	CASEOF	003F
ACMISR	02D7	CASFLAG	030F
ACMVAR	03ED	CASINIT	C6AE
ACTCHRNUM	0063	CASINITV	0002
ADD28E	C87B	CASMOTOFF	FD05
ADD28EGET	C8C3	CASOPEN	FCE6
ADD28EPUT	C8E0	CASPUTBYT	FDB4
ADD28EWRD	C8A0	CASRDBYTE	FD7A
ADDRESS	0064	CASREADBL	FD8D
ADJUST	ED2E	CASSPEED	02EE
ADJUSTTAB	EE11	CASSTART	03EA
AKTBUFLEN	006B	CHAINLINK	03FB
ATASCICHR	02FB	CHAINTMP	02EC
ATRACTMSK	004E	CHAINTP1	0312
ATTRACT	004D	CHAINTP1H	0313
AUDICNTL1	D204	CHARBASE	D409
AUDICNTL2	D205	CHARBASE$	02F4
AUDICNTL3	D206	CHARCNTL	D401
AUDICNTL4	D207	CHARCNTL$	02F3
AUDICOM	D208	CHARSTPTR	026B
AUDIFREQ1	D200	CHECKFF	CB64
AUDIFREQ2	D201	CHECKNEWP	C9EA
AUDIFREQ3	D202	CHECKROM1	FF6E
AUDIFREQ4	D203	CHECKROM2	FF8D
BASMEMTOP	000E	CHECKSRO	C000
BEEPCOUNT	0040	CHECKSUM	008B
BEEPWAIT	FDFC	CHECKSUM2	FFF8
BEGINREAD	ED3D	CHKSUMSND	003B
BELL	F556	CHKSUMTAB	FFD2
BITCLR	F74A	CIOCLOSE	E57C
BITCON	F723	CIOINIT	E4C1
BITGET	F75D	CIOJUMP	E6F4
BITMASK	006E	CIOMAIN	E4DF
BITMASKE	CA2F	CIONOTOPN	E4DC
BITPUT	F73C	CIOREAD	E5B2
BITROL	F732	CIORETURN	E670
BLOAD	C637	CIOSTATSP	E597
BLOCK1	C5C9	CIOWRITE	E61E
BOOT	C599	CLEARLINE	F7E2
BOTTOMLIN	F55F	CLEARSCRN	F420
BRKEVENT	C092	CLICKDISA	02DB
BUFENDPTR	0034	CLOCK	0012
BUFFERADR	0015	CMDAUX1	023C
BUFFULL	0038	CMDAUX2	023D
BUFSTR	006C	CMDCMD	023B
CALLTAB	E48F	CMDDEVIC	023A
CALLVKT	E900	COLBAK$	02C8
CALLVKTC1	E894	COLCR	F997
CARTCKSUM	03EB	COLDCARTC	C431
CARTGO	C47F	COLDSTART	0244
CASBOOT	C67C	COLINC	02F9
CASBUFLIM	028A	COLPF0	D016
CASBUFPTR	003D	COLPF0$	02C4
CASCLOSE	FDCF	COLPF1	D017

COLPF1$	02C5	DMASAVE	02DD
COLPF2	D018	DOBUFC	F8B1
COLPF2$	02C6	DOSAKTIV	0009
COLPF3	D019	DOSCROLL	F7F7
COLPF3$	02C7	DOSINITC	C649
COLPM0	D012	DOSINITV	000C
COLPM0$	02C0	DOSVKT	000A
COLPM1	D013	DOSVKTC	C434
COLPM1$	02C1	DRAWTO	F9AF
COLPM2	D014	DRETRY	02BD
COLPM2$	02C2	DSKAUX1	030A
COLPM3	D015	DSKAUX2	030B
COLPM3$	02C3	DSKBUFFER	0304
COLREGSH	004F	DSKBUFPTR	0032
COMENT	E695	DSKBYTCNT	0308
COMPUTE	ECC8	DSKCHKSUM	0031
COMTAB	E72D	DSKCMD	0302
CONSOL	D01F	DSKDEVICE	0300
CONVERT	F5AC	DSKFLAG	0240
CPIRQQ	FC1A	DSKLDADR	0242
CRETRY	029C	DSKRDERR	C64C
CRITICIO	0042	DSKSECCNT	0241
CURSCOL	0055	DSKSECLEN	02D5
CURSOCTAB	F49A	DSKSTAT	0030
CURSORBS	F450	DSKSTATUS	0303
CURSORDWN	F3F3	DSKTIMCON	0246
CURSORHOM	F440	DSKTIMOUT	0306
CURSORINH	02F0	DSKUNIT	0301
CURSORLFT	F400	EDITORVKT	E400
CURSORRIG	F411	EGETCH	F24A
CURSORTAB	F47A	ENDPOINTR	0074
CURSORUP	F3E6	EOUTCH	F2B0
CURSOSTAB	F495	ERANGE	F6BC
CURSROW	0054	ERRORFLAG	023F
DBDDEC	F565	ESCAPE	F3E0
DCBINIT	E7BE	ESCAPEFLG	02A2
DCBXINIT	E833	EXITVBL	C298
DECBFP	E6C8	EXTEND	F7C2
DECBUFL	E6BB	FILEDAT	02FD
DECTIMER	C263	FILEMNGMT	0043
DELCHAR	F4D5	FINESCRFL	026E
DELLINE	F520	FINESCROL	026C
DELTIA	F918	FKDEF	FB11
DERRMSG	C44B	FKTDEFPTR	0060
DEVICSRCH	E712	FREE0	CB73
DEVICSTAT	02EA	FREE1	EF65
DEVS2PL3	E6FF	FREE2	FCD6
DISKFORM	0018	GAPTYPE	003E
DISKINIT	C6B1	GETBLOCK	C667
DISKINTERF	C6C1	GETBYTEAC	C7DD
DISKUTIL	001A	GETCH	F180
DISPLYFLG	02FE	GETCHECKS	FFA4
DISPLYMSK	02A0	GETLOWEST	C9BD
DLIVKT	0200	GETPLT	F18F
DLPTR$	0230	GETRAMHI	C4B7
DLPTRH	D403	GOHANDLER	E6EA
DLPTRL	D402	GOMEMTEST	C3BD
DMACNTL	D400	GRAFM	D011
DMACNTL$	022F	GRAFP0	D00D

GRAFP1	D00E	IOCBDSKNZ	0021
GRAFP2	D00F	IOCBNUMZ	002E
GRAFP3	D010	IOCBPUTBZ	O026
GRAPHEMUL	0057	IOCBSTATZ	0023
GTIACNTL	D01B	IOPORTINI	C4E8
GTIACNTL$	026F	IOSOUNDEN	0041
HANDLERTB	E440	IRQEN	D20E
HATABS	031A	IRQEN$	0010
HELPFLAG	02DC	IRQST$	0011
HIBYTELD	0288	IRQSTAT	D20E
HELPWORD	0000	ISRODN	EAAD
HITCLR	D01E	ISRSIR	EB2C
HIUSEDLOD	02CB	ISRXD	EAEC
HNDLRLOAD	02E9	JMPCASOFF	EF6B
HOLD2	029F	JMPF21E	F715
HPOSM0	D004	JMPF983	FCD8
HPOSM1	D005	JMPIRQVKT	C02C
HPOSM2	D0O6	JMPSIOINIT	E959
HPOSM3	D007	JMPTAB	E450
HPOSP0	D000	JMPTIMER1	C25D
HPOSP1	DC01	JMPTIMER2	C260
HPOSP2	D002	JOYSTICK0$	0278
HPOSP3	D003	JOYSTICK1$	0279
HSCROL	D404	JOYSTICK2$	027A
INATEMP	022D	JOYSTICK3$	027B
INATAC	F76A	JSRIND	F2AD
INCBFP	F60A	KBCODE	D200
INCLOAD	E816	KBCODE$	02FC
INCRSB	F6OA	KBDISABLE	026D
INDSETVKT	E912	KBGETCHAR	F302
INIT	FCDB	KBVKT	E420
INIT200	C459	KEYCLICK	F903
INIT31A	C34C	KEYDEF	FB51
INITCARTC	C437	KEYDEFPTR	0079
INITLOAD	E7DE	KEYDELAY	02F1
INITPOTS	C239	KEYREP	02DA
INITSOME	EF8E	KEYRPDELY	02D9
INSCHAR	F49F	KOLM0PF	D000
INSLINE	F50C	KOLN0PL	D008
INTERCHAR	CC00	KOLM1PF	DO01
INTTAB	ECA9	KOLM1PL	D009
INTVKT	FFFE	KOLM2PF	D002
IOCB0	0340	KOLM2PL	D00A
IOCB1	0350	KOLM3PF	D003
IOCB2	0360	KOLM3PL	D00B
IOCB3	0370	KOLP0PF	D004
IOCB4	0380	KOLP0PL	D00C
IOCB5	0390	KOLP1PF	D005
IOCB6	03A0	KOLP1PL	D00D
IOCB7	03B0	KOLP2PF	D006
IOCBAUX1Z	002A	KOLP2PL	D00E
IOCHAUX2Z	0023	KOLP3PF	DC07
ICCBAUX3Z	002C	KOLP3PL	D00F
IOCHAUX4Z	002D	LCOUNT	0233
IOCBBUFAZ	002F	LFTMARGIN	0052
IOCHBUFLZ	0020	LINEINSRT	F78E
IOCBCHARZ	0024	LINK	E898
IOCBCHIDZ	0028	LINKSOMETCH	E739
IOCBCMD	0017	LOADER	C753
IOCBCMDZ	0022	LOADERTMP	0036

LOADPTR	EB87	OUTPLT	F1CA
LODADRESS	02D1	PADDLE0$	0270
LODGBYTEA	02CF	PADDLE1$	0271
LODZHIUSE	02CD	PADDLE2$	0272
LODZLOADA	02D3	PALDLE3$	0773
LOGGET	F758	PADDLF4$	0274
LOGICMASK	02B2	PADDLE5$	0275
LOWMEM	0080	PADDLE6$	0276
LPENH	D40C	PADDLS7$	0277
LPENH$	0234	PADTRIGS	C23C
LPENV	D40D	PHACRS	F94C
LPENV$	0235	PHCLOSE	FF02
LPTVKT	E430	PHOPEN	FEC2
MASKTAB	C0CF	PHPUT	FF3F
MATHMUELL	D805	PHSTAT	FEA3
MEMLO	02E7	PHWIRITE	FECB
MEMTOP	02E5	PLACRS	F957
MINTLK	03F9	PLOTCOLAC	0072
MLTTMP	0066	PLOTROWAC	0070
MONSTATUS	004C	PMBASE	D407
MONTEMP1	0051	PMCNTL	D01D
NEUDEVC1	C99F	PNMI	C018
NEUDEVC3	C9A4	POKTAB	EDF9
NEUDEVC5	C9A9	PORTA	D30C
NEUDEVC7	C9AE	PORTACNTL	D302
NEUDEVC9	C9B3	PORTB	D301
NEUDEVCB	C9B8	PORTBCNTL	D303
NEUINIT	C91A	POT0	D200
NEUINITC	E49H	POT1	D201
NEUIODREQ	0248	POT2	D202
NEUIOINIV	0238	POT3	D203
NEUIOMASK	0249	POT4	D204
NEUIOPTR	028C	POT5	D205
NEUIOREQ	C97C	POT6	D206
NEUPORT	D1FF	POT7	D207
NEUPORTERR	C9D8	POTGO	D20B
NEUPORTST	CA11	POTSTAT	D208
NEUVECTOR	C8F2	POWUPBYT1	033D
NEWVORHDN	0247	POWUPBYT2	033E
NEWADRLOD	028E	POWUPBYT3	033F
NEWCART	C4D7	PREPLINK	CA37
NEWDEVICE	EEBC	PRINTBUF	03C0
NEWGRCOL	02F6	PRMODE	FF46
NEWGRROW	02F5	PRTBUFPTR	02DE
NEWLDTMP1	024A	PRTBUFSIZ	02DF
NMIEN	D40E	PRWONA	EF6E
NMIENABLE	C00C	PTIMOUT	0314
NMIRES	D40F	PTRIG0$	027C
NMIST	D40F	PTRIG1$	027D
NMIVKT	FFFA	PTRIG2$	027E
NOCHKSUM	003C	PTRIG3$	027F
NTSCPAL$	OO62	PTRIG4$	0280
NUMTXTLIN	02BF	PTRIG5$	0281
OFFCURSOR	F718	PTRIG6$	0282
OLDGRADR	005E	PTRIG7$	0283
OLDGRCHR	005D	PUTADRESS	C748
OLDGRCOL	005B	PUTBYTE1	EF26
OLDGRROW	005A	PUTCHAR	C7E3
OPTIONJMP	030E	PUTLINE	C650
OUTCH	F1A4	PUTMSC	F9AE

RAMSIZE	02E4	STACKSAVE	0318
RAMTOP	00EA	STANDCHAR	E000
RAMTSTPTR	0004	STARTSTOP	02FF
RANDOM	D20A	STARTTST	03E9
READJOY0	C20E	STATUS	FDCC
READJOY1	C201	STIMER	D209
READPOTS	C223	STRBEG	F90C
READTRIG0	C219	SUBBFL	E6D8
READTRIG1	C222	SUBEND	F6AE
RECEIVE	EAFD	SUBTEMP	029E
RECEIVEN	EC40	SUPERFLAG	03E8
RECEIVEND	0039	SWAP	F962
RECLEN	0245	SWAPANSPR	F21E
RESET	C2D6	SWAPFLAG	007B
RESETCOLD	C2B8	SWITCHROM	C90A
RESETVKT	FFFC	SYSINIT	C543
RESETWARM	C29E	SYSTEMVBL	C0F0
RETURM	F20B	TABELLE	CA65
RIGMARGIN	0053	TABELLE	FB04
ROMFLAG	03FA	TABELLE	FE8D
ROWINC	02F8	TABMAP	02A3
RTS	E4C0	TABSIOINI	E7D4
RUNADRC	C7E0	TEMPLBT	02A1
RUNADRLOD	02C9	TEMPROW	02B8
RUNLOADED	C7A3	TEMPSTAT	0319
SAVEADR	0068	TESTCNTL	F93C
SCREENVKT	E410	TESTDATA	0007
SCRNSTART	0058	TESTROMEN	F223
SCROLFINE	F22E	TEXTBUF	0580
SEARCHLNK	E85D	TEXTCOL	0291
SEND	EA88	TEXTGRAD	029A
SENDDIS	EC84	TEXTINDEX	0293
SENDENABL	EC17	TEXTOLD	0296
SENDINIT	ECAF	TEXTROW	0290
SERIN	D20D	TEXTSTART	0294
SEROUT	D20D	TIMCOUNT1	0218
SETDCB	FFOF	TIMCOUNT2	021A
SETTIM1V	EDE2	TIMCOUNT3	021C
SETTIMOUT	EC9A	TIMCOUNT4	021E
SETVBLVKT	C280	TIMCOUNT5	0220
SHFAMT	006F	TIMEFLAG	0317
SHIFTLOCK	02BE	TIMER1VKT	0226
SINRDYIRQ	C030	TIMER2VKT	0228
SIO	E971	TIMER3SIG	022A
SIOINIT	E95C	TIMER4SIG	022C
SIOINTERF	C941	TIMER5SIG	022E
SIOSYSBUF	FE3F	TMOUTINT	EC11
SIOTAB	E851	TMPRAMSIZ	0006
SIZEM	D00C	TRIG0$	0284
SIZEP0	D008	TRIG0	D010
SIZEP1	D009	TRIG1$	0285
SIZEP2	D00A	TRIG1	D011
SIZEP3	D00B	TRIG2$	0286
SKCNTL	D20F	TRIG2	D012
SKCNTL$	0232	TRIG3$	0287
SKSTAT	D20F	TRIG3	D013
SKSTATRES	D20A	UNLINK	E915
SPECHANDL	EEF9	VBLKDVKT	0224
SRTIMER	022B	VBLKIVKT	0222

VBREAK	0206	VTIMER2	0212
VBREAKKEY	0236	VTIMER4	0214
VCOUNT	D40B	WAIT	EA37
VDELAY	D01C	WAITHSYNC	D40A
VECTAB	C0D7	WAITRESET	C0DF
VIMMEDIRQ	0216	WARMFLAG	0008
VINTERRUP	0204	WRITEMODE	0289
VKEYBOARD	0208	WSIOSB	FE7C
VPRECEDE	0202	X64KBFLAG	03F8
VSCROL	D405	XMITEND	003A
VSERCLOSE	020F	XORKEYMSK	02B6
VSERIELIN	020A	ZCHAIN	004A
VSERREADY	020C	ZCHAINH	004B
VTIMER1	0210		