Stołeczny Ośrodek
Elektronicznej
Techniki Obliczeniowej

INFORMATYKA
mikrokomputerowa

Wojciech Zientara

Mapa pamięci Atari XL/XE: Podstawowe procedury systemu operacyjnego

ATARI

WARSZAWA 1988

Spis treści

PRZEDMOWA

Początkujący programiści piszą zwykle programy bardzo zbliżone do siebie, niezależnie od typu komputera. W miarę nabywania doświadczenia i poznawania swojego komputera pojawia się coraz więcej instrukcji związanych ze sprzętem. Najpierw są to proste PEEK i POKE, później wykorzystuje się poszczególne procedury systemu operacyjnego. W tym miejscu wielu użytkowników napotyka na poważną przeszkodę - brak odpowiedniej literatury.

Niniejszy opis podstawowych procedur systemu operacyjnego komputerów Atari serii XL i XE powinien pomóc użytkownikom tego sprzętu w pełniejszym wykorzystaniu jego możliwości. W najbliższej przyszłości przewidywane jest wydanie kolejnych książek poświęconych pozostałym procedurom systemu operacyjnego Atari XL/XE.

Książka ta jest przeznaczona przede wszystkim dla użytkowników znających już programowanie w języku maszynowym mikroprocesora 6502. Osoby nie znające tego języka także będą mogły korzystać z zamieszczonych tu opisów procedur, choć w ograniczonym zakresie. Podane adresy rejestrów wykorzystywanych przez system operacyjny mogą być użyte również w programach napisanych w Basicu poprzez instrukcje PEEK, POKE i USR.

Osobom nie znającym języka maszynowego mogę polecić książkę Jana Ruszczyca "Asembler 6502" wydaną przez SOETO. Stanowi ona podstawowy podręcznik zarówno dla początkujących, jak i zaawansowanych programistów.

Wszystkie zamieszczone w książce procedury zostały napisane w formacie asemblera MAC/65. W przypadku posiadania innego asemblera konieczne będzie dokonanie drobnych poprawek w niektórych słowach kluczowych procedur, aby mogły one działać prawidłowo.

Na końcu książki zamieszczone zostały dodatki ułatwiające korzystanie z niej oraz słownik niektórych użytych terminów i bibliografia pozwalająca na poszerzenie wiedzy o systemie operacyjnym Atari. Osoby, które nie mają wprawy w posługiwaniu się innymi systemami liczbowymi niż dziesiętny, znajdą tam także tabelę przeliczeń pomiędzy systemami dziesiętnym, dwójkowym i szesnastkowym.

WSTĘP

System operacyjny (OS - Operating System) jest zapisanym w pamięci ROM zestawem procedur nadzorujących pracę całego systemu komputerowego, to znaczy samego komputera oraz urządzeń peryferyjnych. W komputerach Atari OS jest dokładnie przemyślanym, samodzielnym pakietem programowym, który umożliwia całkowitą ubsługę systemu bez odwoływania się do dodatkowych procedur znajdujących się np. w interpreterze Basica, jak to ma miejsce w ZX-Spectrum.

Cały system operacyjny Atari zajmuje 16 KB pamięci ROM, podzielone na kilka części. Dwie podstawowe części OS zajmują obszary $C000-$CFFF (49152-53247) oraz $E000-$FFFF (57344-65535). W tych obszarach znajdują się wszystkie procedury oraz tabele niezbędne do działania systemu.

Trzecim blokiem systemu operacyjnego jest pakiet procedur zmiennoprzecinkowych służących do wykonywania obliczeń na liczbach rzeczywistych. Zajmuje on 2 KB pamięci i znajduje się w obszarze $D800-$DFFF (55296-57343).

Kolejny blok 2 KB pamięci ROM od $D000 do $D7FF (53248-55295) jest zajęty przez program testowania komputera. Normalnie jest on niedostępny dla użytkownika, gdyż w tej samej przestrzeni adresowej znajdują się rejestry sprzętowe specjalizowanych układów scalonych komputera (procesor 6502 nie ma wydzielonej przestrzeni adresowej dla układów I/O). W chwili wywołania tego programu następuje sprzętowe przełączenie linii adresowych i pozorne przeniesienie tego programu do obszaru $5000-$57FF (20480-22527).

System operacyjny wykorzystuje także dla swoich potrzeb część pamięci RAM. Jest to połowa strony zerowej ($00-$7F, 0-127) oraz obszar od $0200 do $047F (512-1151). Oprócz tego obszar pierwszej strony pamięci ($0100-$01FF, 256-511) zajęty jest przez stos mikroprocesora.

Kolejne modernizacje 8-bitowej rodziny Atari pozostawiały ślad także w systemie operacyjnym, który jest spotykany w kilku odmianach. Rozpoznanie wersji jest możliwe dzięki informacjom zawartym w dwóch obszarach pamięci - $C002-$C00B (49154-49163) i $FFEE-$FFF7 (65518-65527):

```
$C002-$C004 (49154-49156)     Revision Date
$C005       (49157)           Option Byte
$C006-$C00A (49158-49162)     Part Number
$C00B       (49163)           Revision Number
$FFEE-$FFF0 (65518-65520)     Revision Date
$FFF1       (65521)           Option Byte
$FFF2-$FFF6 (65522-65526)     Part Number
$FFF7       (65527)           Revision Number
```

Opis będzie dotyczył najczęściej spotykanej w Polsce wersji B (O.S. Revision B). Poniżej podane są wartości znajdujące się w obszarach identyfikacyjnych tej wersji.

Revision Date jest to data zatwierdzenia wersji OS zapisana w kodzie BCD. Są tam wpisane wartości $10, $05 i $83. Part Number jest numerem serii zapisanym w formacie AANNNNNN, gdzie A jest znakiem ASCII, a N - liczba w kodzie BCD. Znajdują się tam kolejno wartości: $42, $42, $00, $00 i $01 (BB000001). Numer wersji (Revision Number) ma wartość $02 w 800XL, 65XE i 130XE. Atari 600XL zawiera w $FFFB wartość $01, a 1200XL - $11. Option Byte w pierwszym bloku ($C005) jest równy $00, a w drugim ($FFF1) $02 (w 1200XL - $01).

Jeżeli po sprawdzeniu zawartości powyższych komórek okaże się, że są one inne, nie należy się tym szczególnie przejmować. Wszystkie wersje Atari mają podobny system operacyjny. Różnice najczęściej sprowadzają się do rozmieszczenia procedur w pamięci (przesunięcie o kilka bajtów) i drobnej kosmetyki. Adresy początkowe wielu procedur można w takim przypadku odnaleźć dzięki wektorom umieszczonym w pamięci RAM i ROM oraz przy pomocy tablicy skoków. W najgorszym przypadku można odszukać procedurę przez poszukiwanie w pamięci RAM określonej sekwencji bajtów.

Programy pisane dla komputerów Atari powinny w zasadzie korzystać wyłącznie z tablicy skoków. Zapewnia to ich prawidłowe działanie na wszystkich istniejących modelach. Ponieważ w Polsce znajdują się praktycznie jedynie modele XL i XE, to oprogramowanie może także korzystać bezpośrednio z większości procedur systemu operacyjnego.

Rozdział 1

TABLICA SKOKÓW

Skoro istnieje wiele wersji systemu operacyjnego, to jak w takim razie zapewniona jest pełna zgodność oprogramowania dla różnych modeli? Osiągnięte jest to dzięki tak zwanej tablicy skoków. Ma ona ustalone miejsce w pamięci (w każdej wersji takie samo). Dzięki temu odwołania do procedur, które korzystają z tej tabeli, zawsze trafią w odpowiednie miejsce, ponieważ zawiera ona aktualny adres procedury.

Wyjaśnia to również, dlaczego niektóre programy wywagają użycia "Translatora". Po prostu autor programu nie korzystał z tablicy skoków, lecz odwoływał się bezpośrednio do procedur OS. "Translator" (lub mający identyczne działanie "XL Fix") odłącza pamięć ROM zawierającą system operacyjny i wpisuje w RAM znajdującą się w tym obszarze poprzednią wersję OS (z modeli 400/800).

Tablica skoków w modelach XL/XE wygląda następująco:

```
0100 ;JUMP TABle
0110 ;
0120 CASOPIN = $FCF7
0130 CASRDBL = $FD8D
0140 CIOINIT = $E4C1
0150 CIOMAIN = $E4DF
0160 DSKINIT = $C6A3
0170 DSKINT = $C6B3
0180 EXITVBL = $C28A
0190 LINK = $E898
0200 NEWDEVC = $EEBC
0210 NMIENBL = $C00C
0220 RESETCD = $C2C8
0230 RESETWM = $C290
0240 SETVBLV = $C272
0250 SIOINIT = $E95C
0260 SIOINT = $C933
0270 SNDENBL = $EC17
0280 SYSVBL = $C0E2
0290 TESTROM = $F223
0300 TESTST = $5000
0310 UNLINK = $E915
0320 ;
0330     *=  $E450
0340 ;
0350     JMP DSKINIT
0360     JMP DSKINT
0370     JMP CIOMAIN
0380     JMP SIOINT
0390     JMP SETVBLV
0400     JMP SYSVBL
0410     JMP EXITVBL
```

```
0420     JMP SIOINIT
0430     JMP SNDENBL
0440     JMP NMIENBL
0450     JMP CIOINIT
0460     JMP TESTROM
0470     JMP RESETWM
0480     JMP RESETCD
0490     JMP CASRDBL
0500     JMP CASOPIN
0510     JMP TESTROM
0520     JMP TESTST
0530     JMP NEWDEVC
0540     JMP UNLINK
0550     JMP LINK
```

Poprzednie modele (400/800) nie zawierały w tablicy odwołań do procedur TESTROM, TESTST, NEWDEVC i LINK.

Dwukrotne umieszczenie w tablicy skoku do procedury TESTROM nie jest pomyłką. Pierwszy skok jest przewidziany do obsługi niektórych błędów systemu i w przyszłości może ulec zmianie, gdy zostanie dodana nowa procedura.

Wydruk tablicy skoków jest tak prosty, że nie wymaga właściwie żadnego dodatkowego komentarza. Można jedynie zauważyć, że ujęte są tu wszystkie procedury istotne dla pracy systemu komputerowego oraz wszystkich możliwych translatorów języków programowania.

Rozdział 2

PROCEDURY STARTU KOMPUTERA

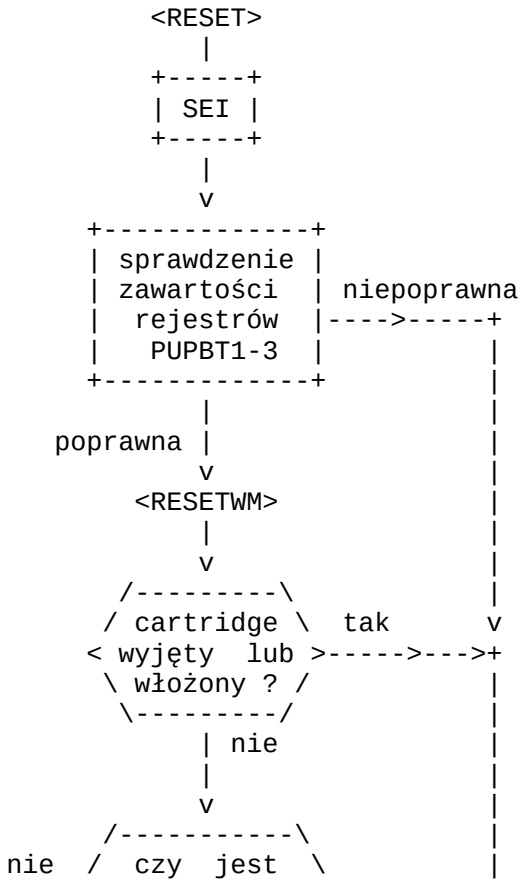
2.1. Główna procedura RESET

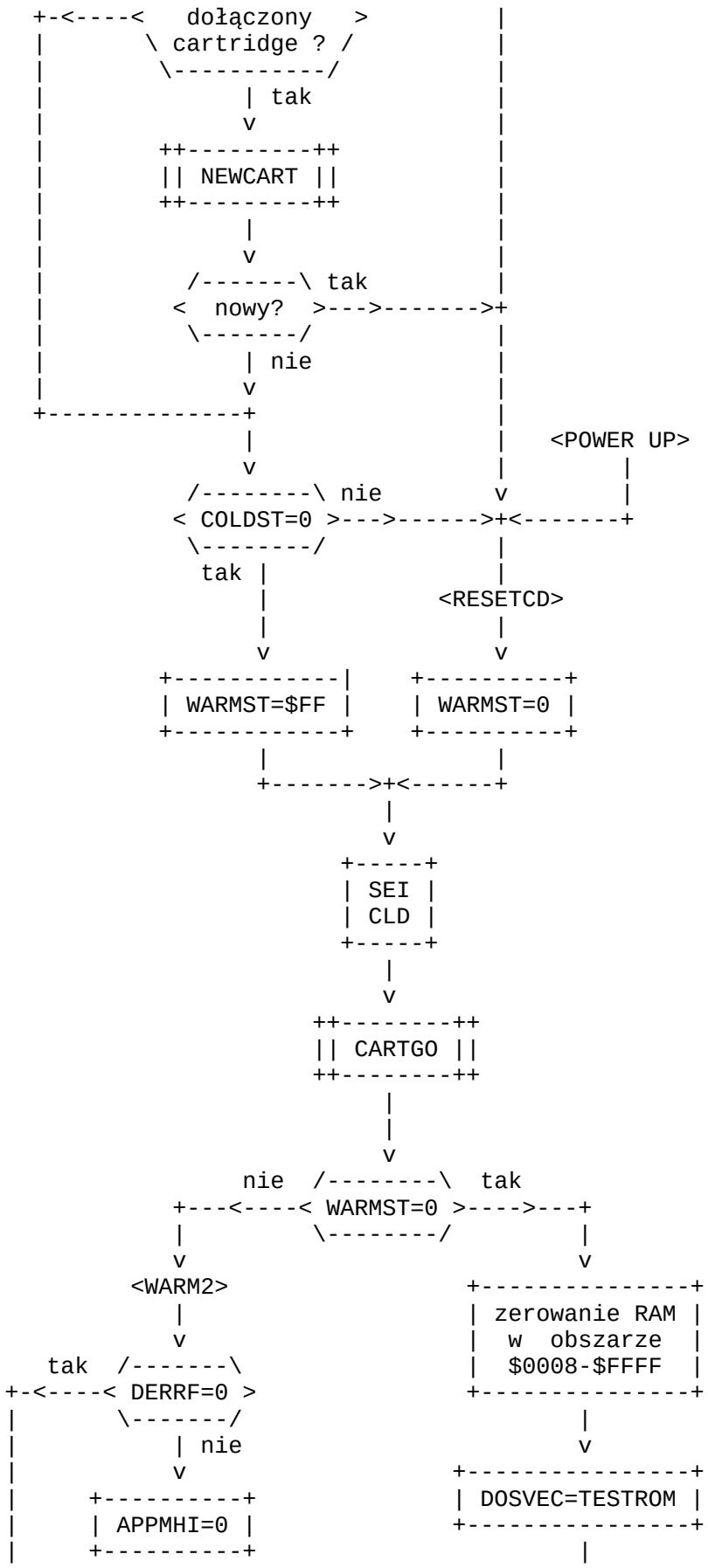
Po włączeniu zasilania komputera oraz po naciśnięciu klawisza RESET, który zeruje układy komputera, konieczne jest zainicjowanie pracy systemu. Zadanie to wykonuje procedura RESET mająca dwa warianty przebiegu.

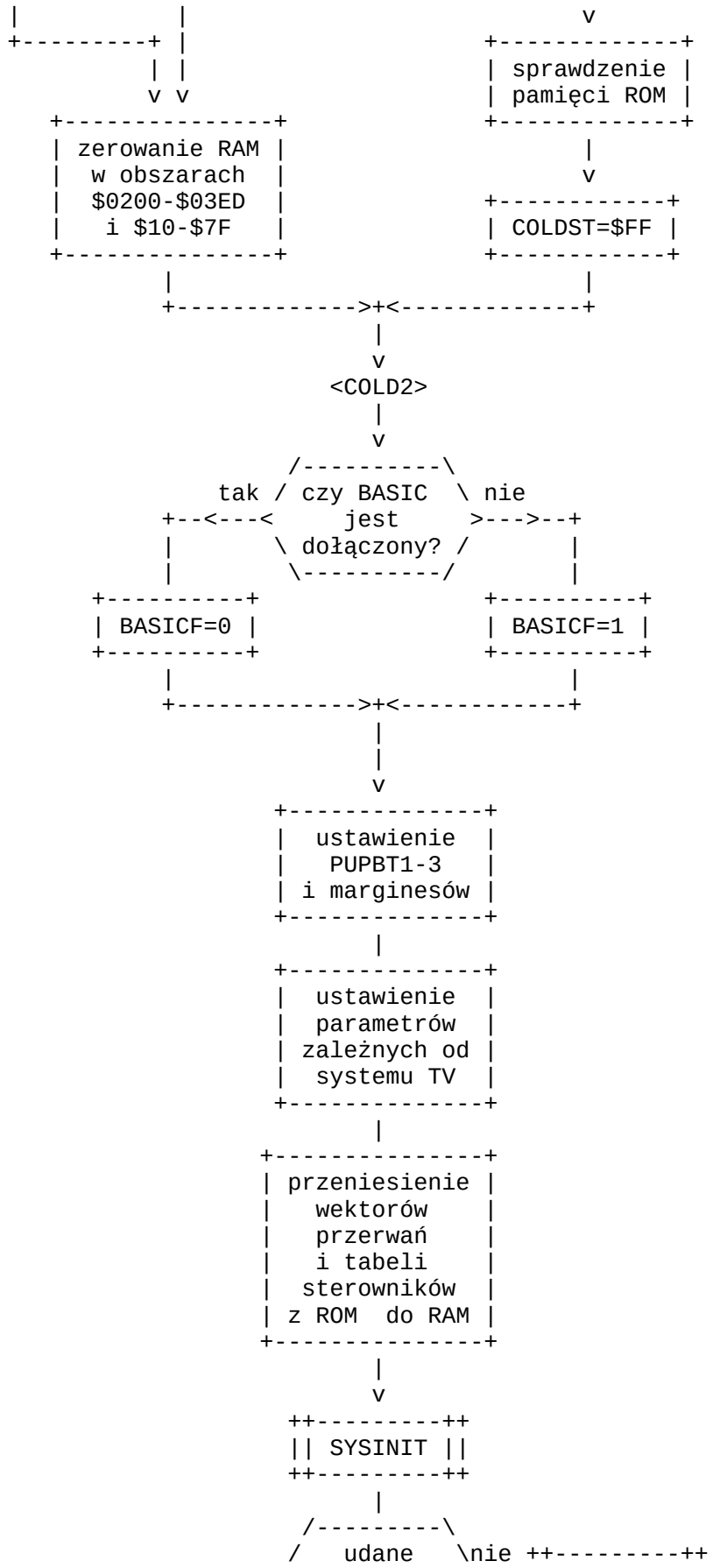
Wariant zimnego startu (RESET COLD) jest wykonywany zawsze po włączeniu zasilania (tzw. Power Up). Po naciśnięciu klawisza RESET wykonywany jest wariant startu gorącego (RESET WARM). Należy pamiętać, że dokonanie zmian w niektórych rejestrach systemowych w obszarze RAM może spowodować przejście z procedury startu gorącego do startu zimnego. Odwrotna zmiana wariantu procedury jest niemożliwa.

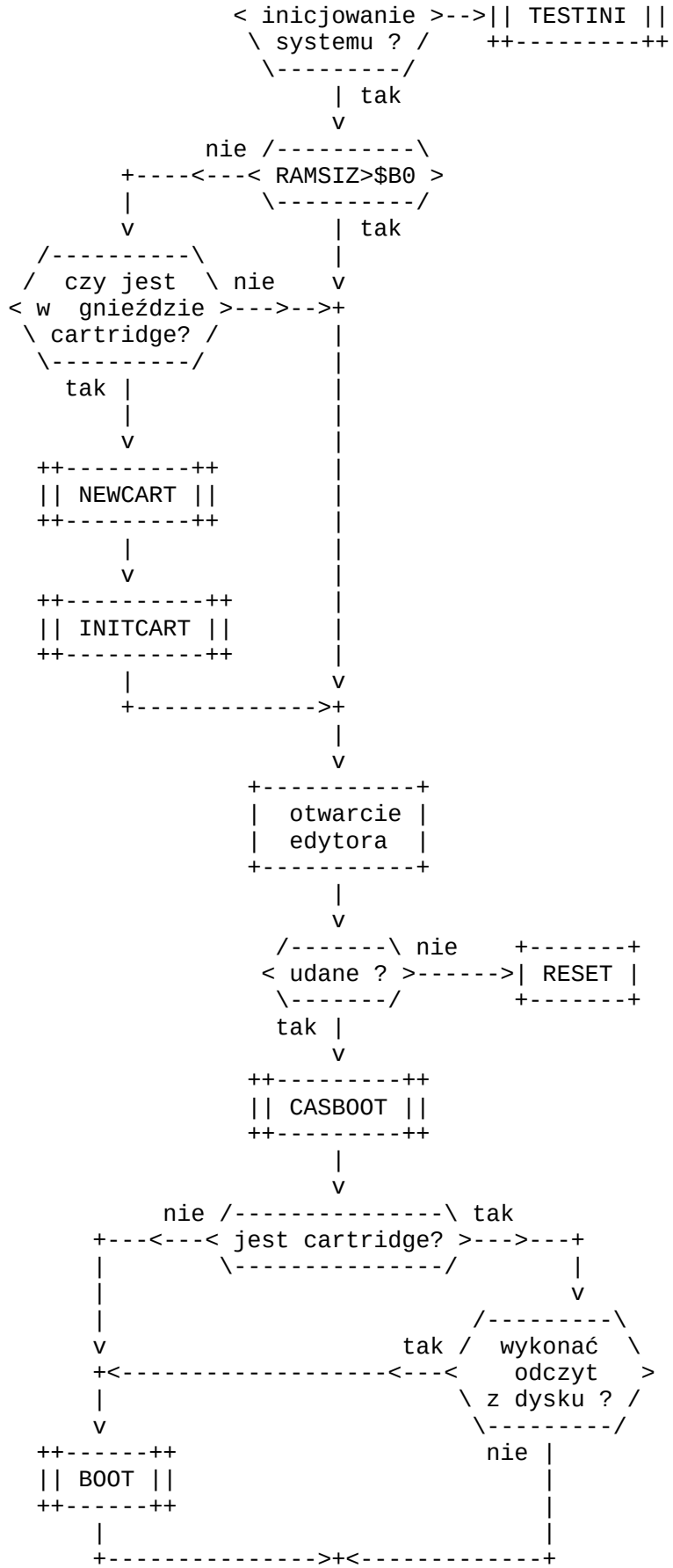
2.1.1. Struktura procedury RESET

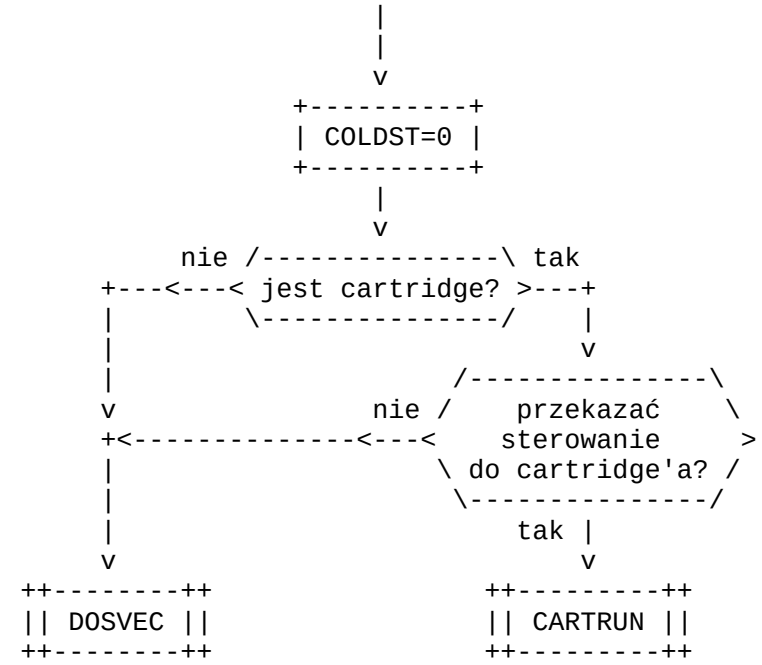
Cały przebieg procedury startu jest bardzo skomplikowany, zostanie więc przedstawiony etapami poczynając od schematu blokowego.




dołączony
cartridge ?
tak
NEWCART
nowy?
tak
nie
<POWER UP>
COLDST=0
tak
<RESETCD>
WARMST=$FF
WARMST=0
SEI
CLD
CARTGO
nie
WARMST=0
tak
<WARM2>
zerowanie RAM
w obszarze
$0008-$FFFF
tak
DERRF=0
nie
APPMHI=0
DOSVEC=TESTROM


sprawdzenie
pamięci ROM
zerowanie RAM
w obszarach
$0200-$03ED
i $10-$7F
COLDST=$FF
<COLD2>
tak / czy BASIC \ nie
jest
\ dołączony? /
BASICF=0
BASICF=1
ustawienie
PUPBT1-3
i marginesów
ustawienie
parametrów
zależnych od
systemu TV
przeniesienie
wektorów
przerwań
i tabeli
sterowników
z ROM do RAM
SYSINIT
udane
nie


< inicjowanie >-->|| TESTINI ||
\ systemu ? / ++---------++
\---------/
| tak
v
nie /----------\
+----<---< RAMSIZ>$B0 >
| \----------/
v | tak
/----------\ |
/ czy jest \ nie v
< w gnieździe >--->-->+
\ cartridge? / |
\----------/ |
tak | |
| |
v |
++---------++ |
|| NEWCART || |
++---------++ |
| |
v |
++----------++ |
|| INITCART || |
++----------++ |
| v
+------------->+
|
v
+-----------+
| otwarcie |
| edytora |
+-----------+
|
v
/-------\ nie +-------+
< udane ? >------>| RESET |
\-------/ +-------+
tak |
v
++---------++
|| CASBOOT ||
++---------++
|
v
nie /---------------\ tak
+---<---< jest cartridge? >--->---+
| \---------------/ |
| v
| /---------\
v tak / wykonać \
+<-------------------<---< odczyt >
| \ z dysku ? /
v \---------/
++------++ nie |
|| BOOT || |
++------++ |
| |
+--------------->+<-------------+



2.1.2. Przebieg procedury RESET

Po włączeniu zasilania układy wewnętrzne komputera ustawiają (na drodze sprzętowej) licznik programu procesora na wartość RESETCD ($C2C8). Po przekazaniu sterowania procesorowi zaczyna on wykonywanie programu od tego właśnie miejsca.

Odmiennie zachowuje się procesor po otrzymaniu sygnału RESET. Pobiera on dwa bajty z rejestru RESETVEC ($FFFC) i umieszcza je w liczniku programu. Ta operacja także dokonywana jest sprzętowo. W komputerach Atari w rejestrze RESETVEC znajduje się adres procedury RESET ($C2AA). Od tego więc miejsca kontynuowane jest wykonywanie programu.

Zasadnicza procedura RESET przedstawia się następująco:

```
0100 ; RESET
0110 ;
0120 APPMHI = $0E
0130 BASICF = $03F8
0140 BOOT = $C58B
0150 CARTGO = $C471
0160 CARTINI = $BFFE
0170 CARTINS = $BFFC
0180 CARTOPT = $BFFD
0190 CARTRUN = $BFFA
0200 CASBOOT = $C66E
0210 CHBAS = $02F4
0220 CHACTL = $02F3
0230 CKROM1 = $FF73
0240 CKROM2 = $FF92
0250 COLDST = $0244
0260 DDEVIC = $0300
0270 DERRF = $03EC
0280 DLIV =  $0200
0290 DOSVEC = $0A
0300 GINTLK = $03FA
0310 HATABS = $031A
0320 ICAX1 = $034A
0330 ICBUFA = $0344
0340 ICCMD = $0342
0350 INIT31A = $C42E
0360 INIT200 = $C44B
0370 JCIOMAIN = $E456
0380 KEYREP = $02DA
0390 KRPDEL = $02D9
0400 LINKSOM = $E739
0410 LMARGIN = $52
0420 LNFLG = $00
0430 NEWCART = $C4C9
```

```
0440 NGFLAG = $01
0450 PAL =   $D014
0460 PALNTS = $62
0470 PORTB = $D301
0480 PUPBT1 = $033D
0490 PUPBT2 = $033E
0500 PUPBT3 = $033F
0510 RAMLO = $04
0520 RAMSIZ = $02E4
0530 RMARGIN = $53
0540 SYSINIT = $C535
0550 TESTINI = $5003
0560 TESTROM = $F223
0570 TRAMSZ = $06
0580 TRIG3 = $D013
0590 WARMST = $08
0600 ;
0610     *=  $C290
0620 ;
0630 RESETWM SEI
0640     LDA TRIG3
0650     CMP GINTLK
0660     BNE RESETCD
0670     ROR A
0680     BCC OLDCART
0690     JSR NEWCART
0700     BNE RESETCD
0710 OLDCART LDA COLDST
0720     BNE RESETCD
0730     LDA #$FF
0740     BNE CONT
0750 RESET SEI
0760     LDX #$8C
0770 DELAY1 DEY
0780     BNE DELAY1
0790     DEX
0800     BNE DELAY1
0810     LDA PUPBT1
0820     CMP #$5C
0830     BNE RESETCD
0840     LDA PUPBT2
0850     CMP #$93
0860     BNE RESETCD
0870     LDA PUPBT3
0880     CMP #$25
0890     BEQ RESETWM
0900 RESETCD LDA #$00
0910 CONT STA WARMST
0920     SEI
0930     CLD
0940     LDX #$FF
0950     TXS
0960     JSR CARTGO
0970     LDA #$01
0980     STA NGFLAG
0990     LDA WARMST

1560     LDA PORTB
1570     AND #$02
```

```
1000     BNE WARM2
1010     LDA #$00
1020     LDY #$08
1030     STA RAMLO
1040     STA RAMLO+1
1050 LOOP LDA #$FF
1060     STA (RAMLO),Y
1070     CMP (RAMLO),Y
1080     BEQ OK1
1090     LSR NGFLAG
1100 OK1 LDA #$00
1110     STA (RAMLO),Y
1120     CMP (RAMLO),Y
1130     BEQ OK2
1140     LSR NGFLAG
1150 OK2 INY
1160     BNE LOOP1
1170     INC RAMLO+1
1180     LDX RAMLO+1
1190     CPX TRAMSZ
1200     BNE LOOP1
1210     LDA # <TESTROM
1220     STA DOSVEC
1230     LDA # >TESTROM
1240     STA DOSVEC+1
1250     LDA PORTB
1260     AND #$7F
1270     STA PORTB
1280     JSR CKROM1
1290     BCS BAD
1300     JSR CKROM2
1310     BCC ROMOK
1320 BAD LSR NGFLAG
1330 ROMOK LDA PORTB
1340     ORA #$80
1350     STA PORTB
1360     LDA #$FF
1370     STA COLDST
1380     BNE COLD2
1390 WARM2 LDX #$00
1400     LDA DERRF
1410     BEQ LOOP2
1420     STX APPMHI
1430     STX APPMHI+1
1440     TXA
1450 LOOP2 STA DLIV,X
1460     CPX #$ED
1470     BCS BYPASS
1480     STA DDEVIC,X
1490 BYPASS DEX
1500     BNE LOOP2
1510     LDX #$10
1520 LOOP3 STA LNFLG,X
1530     INX
1540     BPL LOOP3
1550 COLD2 LDX #$00

2060 NOERR LDX #$00
2070     STX TRAMSZ
```

```
1580     BEQ BASICON
1590     INX
1600 BASICON STX BASICF
1610     LDA #$5C
1620     STA PUPBT1
1630     LDA #$93
1640     STA PUPBT2
1650     LDA #$25
1660     STA PUPBT3
1670     LDA #$02
1680     STA LMARGIN
1690     LDA #$27
1700     STA RMARGIN
1710     LDA PAL
1720     AND #$0E
1730     BNE TVPAL
1740     LDA #$05
1750     LDX #$01
1760     LDY #$28
1770     BNE SETTV
1780 TVPAL LDA #$06
1790     LDX #$00
1800     LDY #$30
1810 SETTV STA KEYREP
1820     STX PALNTS
1830     STY KRPDEL
1840     LDX #$25
1850 LOOP4 LDA INIT200,X
1860     STA DLIV,X
1870     DEX
1880     BPL LOOP4
1890     LDX #$0E
1900 LOOP5 LDA INIT31A,X
1910     STA HATABS,X
1920     DEX
1930     BPL LOOP5
1940     JSR SYSINIT
1950     CLI
1960     LDA NGFLAG
1970     BNE NOERR
1980     LDA PORTB
1990     AND #$7F
2000     STA PORTB
2010     LDA #$02
2020     STA CHACTL
2030     LDA #$E0
2040     STA CHBAS
2050 GOMEMTST JMP TESTINI
```

```
2080     LDX RAMSIZ
2090     CPX #$B0
2100     BCS NOCART
2110     LDX CARTINS
2120     BNE NOCART
2130     INC TRAMSZ
2140     JSR NEWCART
2150     JSR INITCART
2160 NOCART LDA #$03
2170     LDX #$00
2180     STA ICCMD,X
2190     LDA # <NAME
2200     STA ICBUFA,X
2210     LDA # <NAME
2220     STA ICBUFA+1,X
2230     LDA #$0C
2240     STA ICAX1,X
2250     JSR JCIOMAIN
2260     BPL DELAY2
2270     JMP RESET
2280 DELAY2 INX
2290     BNE DELAY2
2300     INY
2310     BPL DELAY2
2320     JSR CASBOOT
2330     LDA TRAMSZ
2340     BEQ DBOOT
2350     LDA CARTBOOT
2360     ROR A
2370     BCC NOBOOT
2380 DBOOT JSR BOOT
2390     JSR LINKSOM
2400 NOBOOT LDA #$00
2410     STA COLDST
2420     LDA TRAMSZ
2430     BEQ DOSVECC
2440     LDA CARTOPT
2450     AND #$04
2460     BEQ DOSVECC
2470 COLDCART JMP (CARTRUN)
2480 DOSVECC JMP (DOSVEC)
2490 INITCART JMP (CARTINI)
2500 CLCRTS CLC
2510     RTS
2520 ;
2530     *= $C448
2540 NAME .BYTE "E:",$9B
```

Przebieg procedury RESET zależy w znacznym stopniu od rodzaju startu (zimny czy gorący). Przy gorącym starcie przede wszystkim kontrolowane są różne rejestry systemowe w celu sprawdzenia, czy nie zachodzi konieczność wykonania zimnego startu.

Najpierw sprawdzane są trzy rejestry PUPBT (Power UP Byte), które muszą zawierać ściśle określone wartości - kolejno $5C, $93 i $25. W następnej kolejności kontrolowana jest obecność cartridge'a oraz jego rodzaj. Jeżeli został on wyjęty, włożony lub zamieniony, to cały system

inicjowany jest ponownie. Wykorzystywany tu rejestr TRIG3 jest ustawiany na drodze sprzętowej i zawiera 1, jeśli jest dołączony zewnętrzny cartridge. Wartość rejestru GINTLK jest ustalana według TRIG3 podczas inicjowania komputera. Niezgodność ich zawartości powoduje więc przejście do zimnego startu.

Na końcu tego etapu kontrolowany jest wskaźnik COLDST. Jeżeli jego wartość jest różna od zera, także następuje przejście do zimnego startu. Gdy wszystkie sprawdzenia wypadły pomyślnie, wskaźnik WARMST uzyskuje wartość $FF (służy on później do rozróżniania rodzaju startu.

Jeżeli zostało włączone zasilanie lub system rozpoznał konieczność dokonania zimnego startu, to wskaźnik WARMST jest ustawiany na wartość zero. Od tego miejsca obie procedury przebiegają wspólnie.

Następnie wywoływana jest procedura CARTGO, która sprawdza obecność cartridge'a i rozpoznaje jego rodzaj, inicjuje porty I/O, oraz sprawdza wielkość dostępnej pamięci RAM. Obliczona wielkość RAM (w stronach) jest umieszczana w rejestrze TRAMSZ. Kolejnym etapem jest zerowanie pamięci RAM. Przy starcie zimnym kasowana jest zawartość całej pamięci od adresu $08 do wartości umieszczonej w TRAMSZ. Podczas startu gorącego wyzerowaniu ulegają jedynie obszary od $10 do $7F oraz od $0200 do $03ED. Procedura zimnego startu sprawdza jeszcze poprawność pamięci ROM przez przeliczenie sum kontrolnych oraz ustawia wskaźnik COLDST na wartość $FF i wektor DOSVEC na adres procedury TESTROM.

Teraz następuje ustawienie wartości w rejestrach systemu operacyjnego w pamięci RAM. Najpierw wskaźnik BASICF otrzymuje wartość 0, gdy interpreter Basica jest dołączony lub 1, gdy jest odłączony. Do rejestrów PUPBT wpisywane są odpowiednie wartości (zob. wyżej) oraz ustawiane są marginesy ekranu. Po rozpoznaniu według zawartości rejestru PAL rodzaju systemu telewizyjnego (PAL czy NTSC) ustalane są wartości zmiennych systemowych zależnych od częstotliwości zegara taktującego (PAL ma 50 Hz, a NTSC - 60 Hz). Są to wielkości określające szybkość powtarzania klawisza (KEYREP) oraz czas rozpoczęcia powtarzania przy stałym jego wciśnięciu (KRPDEL).

Wektory systemu znajdujące się na drugiej stronie pamięci są teraz przepisywane z tabeli INIT200, która jest umieszczona w ROM-ie. Tabela ta zawiera systemowe wektory przerwań maskowalnych i niemaskowalnych, zajmujące obszar $0200-$0225.

```
0100 ;INITiation table $200
0110 ;
0120 CPUIRQ = $FC19
0130 EXITVBL = $C28A
0140 ISRODN = $EAAD
0150 ISRSIR = $EB2E
0160 ISRXD = $EAEC
0170 PLARTI = $C0CD
0180 RTI =   $C0CE
```

```
0190 SINDRYI = $C030
0200 SYSVBL = $C0E2
0210 ZERO =  $00
0220 ;
0230     *=  $C44B
0240 ;
0250 INIT200 .WORD RTI,PLARTI
0260     .WORD PLARTI,PLARTI
0270     .WORD CPUIRQ,ISRSIR
0280     .WORD ISRODN,ISRXD
0290     .WORD PLARTI,PLARTI
0300     .WORD PLARTI,SINDRYI
0310     .WORD ZERO,ZERO
0320     .WORD ZERO,ZERO
0330     .WORD ZERO,SYSVBL
0340     .WORD EXITVBL
```

Następnie przenoszona jest tabela adresów procedur obsługi urządzeń zewnętrznych (HATABS). Nie jest ona wypełniana w całości, gdyż tabela INIT31A zawiera jedynie adresy procedur zapisanych w ROM-ie. Znajdują się tam adresy procedur obsługi edytora (E:), ekranu (S:), klawiatury (K:), drukarki (P:) i magnetofonu (C:).

```
0100 ;INITiation table $31A
0110 ;
0120 CASVEC = $E440
0130 EDTVEC = $E400
0140 KBDVEC = $E420
0150 PRTVEC = $E430
0160 SCRVEC = $E410
0170 ;
0180     *= $C42E
0190 ;
0200 INIT31A .BYTE 'P
0210     .WORD PRTVEC
0220     .BYTE 'C
0230     .WORD CASVEC
0240     .BYTE 'E
0250     .WORD EDTVEC
0260     .BYTE 'S
0270     .WORD SCRVEC
0280     .BYTE 'K
0290     .WORD KBDVEC
```

Procedury obsługi innych urządzeń (np. stacji dysków lub modemu) są zapisywane w pamięci RAM i dla udostępnienia ich systemowi operacyjnemu tabela HATABS jest uzupełniana podczas późniejszego instalowania tych urządzeń.

Po tych wszystkich operacjach wywoływana jest procedura SYSINIT, która przeprowadza inicjowanie pozostałej części systemu oraz urządzeń zewnętrznych. Będzie ona dokładnie opisana w dalszej części.

Błędy występujące podczas wykonywania powyższych procedur są sygnalizowane w rejestrze

NGFLAG. Po zakończeniu SYSINIT zawartość tego rejestru jest kontrolowana i gdy wskazuje błędy, to program przechodzi do wbudowanej procedury testowania komputera (tzw. SELFTEST). W przeciwnym wypadku kontynuowana jest procedura RESET.

Jeżeli dołączony jest cartridge (rejestr CARTINS ma wartość 1, gdy brak cartridge'a i Basic jest odłączony), to teraz jest on inicjowany (lecz jeszcze nie uruchamiany). Na zakończenie tej części kanał IOCBO jest otwierany dla edytora. Gdy przy otwieraniu wystąpi błąd, następuje skok do początku procedury RESET.

W tym momencie cały system jest już gotowy do pracy. Teraz poszukiwane jest wyjście z procedury RESET do programu użytkownika. Najpierw przez wywołanie CASBOOT dokonywana jest próba odczytu z kasety. Następnie, po sprawdzeniu, czy nie zabrania tego zainstalowany cartridge (CARTOPT=0 oznacza zakaz odczytu), podejmowana jest próba odczytu z dyskietki. Dopiero po tym wskaźnik COLDST jest ustawiany na wartość 0, która spowoduje po naciśnięciu klawisza RESET wykonanie gorącego startu.

Na zakończenie sterowanie komputerem zostaje przekazane pod adres zawarty w rejestrze DOSVEC (jeżeli nie ma cartridge'a i nie został dokonany odczyt z kasety lub dyskietki, to nadal jest tam adres TESTROM) albo, jeśli zainstalowany cartridge zabraniał odczytu, pod adres zawarty w rejestrze CARTRUN. Pewną ciekawostką jest umieszczenie na końcu procedury RESET zupełnie niepotrzebnych tu instrukcji CLC i RTS. Przewidywano wykorzystanie ich przez różne procedury systemu operacyjnego, lecz mogą one z powodzeniem zostać użyte w programie użytkownika.

2.2. Obliczanie sum kontrolnych

Podczas inicjowania komputera sprawdzana jest poprawność układów pamięci ROM zawierających system operacyjny. Dokonywane jest to przez zsumowanie zawartości wszystkich bajtów pamięci i porównanie otrzymanego wyniku z zapisaną w ROM sumą kontrolną.

Ponieważ system operacyjny jest zapisany w dwóch oddzielnych blokach pamięci, które posiadają odrębne sumy kontrolne, to sprawdzenie poprawności ROM wykonywane jest przez dwie procedury - CKROM1 i CKROM2.

Procedura CKROM1 kontroluje pamięć ROM znajdującą się w obszarach $C002-$CFFF, $5000-$57FF i $D800-$DFFF. Obszary $E000-$FFF7 i $FFFA-$FFFF są kontrolowane przez procedurę CKROM2.

Porównanie podanych obszarów ROM ze wskazanymi na początku książki wykazuje dwie dwubajtowe luki. Są to miejsca umieszczenia sum kontrolnych, które oczywiście nie podlegają sprawdzaniu. Pierwszy blok pamięci ma sumę kontrolną pod adresem $C000 (CHSRO1), a drugi pod $FFF8 (CHSRO2).

```
0100 ;Check ROM
0110 ;
0120 CHSRO1 = $C000
0130 CHSRO2 = $FFF8
0140 CKSUM = $8B
0150 GETCKS = $FFA9
0160 ;
0170     *=  $FF73
0180 ;
0190 CKROM1 LDX #$00
0200     STX CKSUM
0210     STX CKSUM+1
0220 LOOP JSR GETCKS
0230     CPX #$0C
0240     BNE LOOP
0250     LDA CHSRO1
0260     LDX CHSRO1+1
0270 CHECK CMP CKSUM
0280     BNE BAD
0290     CPX CKSUM+1
0300     BNE BAD
0310     CLC
0320     RTS
0330 BAD SEC
0340     RTS
0400 CKROM LDX #$00
0410     STX CKSUM
0420     STX CKSUM+1
0430     LDX #$0C
0440     JSR GETCKS
0450     JSR GETCKS
0460     LDA CHSRO2
0470     LDX CHSRO2+1
0480     JMP CHECK
```

Przebieg obu procedur jest bardzo prosty. Po wyzerowaniu rejesrtu CKSUM, w którym będzie zliczana suma bajtów pamięci ROM, wywoływana jest właściwa procedura obliczająca sumę kontrolną - GETCKS. Następnie obliczona suma jest porównywana z sumą zapisaną w ROM. Jeżeli są one równe, to bit przeniesienia (Carry) w rejestrze statusu procesora jest kasowany, w przeciwnym razie jest on ustawiany.

Powrót do procedury RESET z ustawionym bitem Carry powoduje zmianę zawartości rejestru NGFLAG, który służy do kontroli poprawności inicjowania systemu.

Bezpośrednio obliczająca sumę kontrolną procedura GETCKS wymaga przy wywołaniu przekazania w rejestrze X numeru sprawdzanego bloku pamięci. Adres początkowy i końcowy tego bloku jest pobierany z tabeli CKSTAB.

```
0100 ;GET CheCK Sum
0110 ;
0120 CKSUM = $8B
0130 TMPREG = $9E
0140 ;
```

```
0150     *=  $FFA9
0160 ;
0170     LDY #$00
0180 LOOP LDA CKSTAB,X
0190     STA TMPREG,Y
0200     INX
0210     INY
0220     CPY #$04
0230     BNE LOOP
0240     LDY #$00
0250 NEXT CLC
0260     LDA (TMPREG),Y
0270     ADC CKSUM
0280     STA CKSUM
0290     BCC NXT1
0300     INC CKSUM+1
0310 NXT1 INC TMPREG
0320     BNE NXT2
0330     INC TMPREG+1
0340 NXT2 LDA TMPREG
0350     CMP TMPREG+2
0360     BNE NEXT
0370     LDA TMPREG+1
0380     CMP TMPREG+3
0390     BNE NEXT
0400     RTS
0410 ;
0420 CKSTAB
0430     .WORD $C002,$D000
0440     .WORD $5000,$5800
0450     .WORD $D800,$E000
0460     .WORD $E000,$FFF8
0470     .WORD $FFFA,$00
```

GETCKS najpierw pobiera z CKSTAB adres początkowy i końcowy sprawdzanego bloku (według wartości X) i umieszcza je w tymczasowym rejestrze TMPREG. Następnie korzystając z podanych adresów odczytuje zawartości kolejnych komórek pamięci i dodaje je do zawartości CKSUM. Po osiągnięciu adresu końcowego procedura się kończy.

Warto zauważyć, że adres końcowy w tabeli CKSTAB wskazuje na pierwszą komórkę pamięci nie należącą do badanego bloku. Jest to spowodowane zwiększaniem adresu na końcu pętli.

Ostatnia informacja dotyczy rejestrów w pamięci RAM wykorzystywanych przez powyższe procedury. Znajdują się one w obszarze przeznaczonym dla oprogramowania i po procedurze inicjowania systemu nie są więcej wykorzystywane przez OS.

Oprócz procedur sprawdzających poprawność pamięci ROM system operacyjny zawiera jeszcze procedurę obliczającą sumę kontrolną obszaru $BFF0-$C0EF. Sumuje ona wartości bajtów w tym obszarze i umieszcza je w rejestrze CARTCK. Ponieważ przedtem dokonywane jest porównanie obliczonej sumy z aktualną zawartością tego rejestru, umożliwia to także sprawdzenie, czy została dokonana zamiana cartridge'a. W tym właśnie celu procedura NEWCART jest wywoływana na

początku procedury gorącego startu. W przypadku stwierdzenia takiej zamiany następuje przejście do zimnego startu systemu.

```
0100 ;NEW CARTridge ?
0110 ;
0120 CART = $BFF0
0130 CARTCK = $03EB
0140 ;
0150     *=  $C4C9
0160 ;
0170     LDA #$00
0180     TAX
0190     CLC
0200 LOOP ADC CART,X
0210     INX
0220     BNE LOOP
0230     CMP CARTCK
0240     STA CARTCK
0250     RTS
```

Niezależnie od rodzaju startu procedura ta jest wywoływana - gdy cartridge jest dołączony - po zainicjowaniu systemu, a przed zainicjowaniem cartridge'a.

2.3. Procedury inicjowania bloków systemu

Prawidłowa praca całego systemu komputerowego wymaga umieszczenia w rejestrach znajdujących się w pamięci RAM oraz w rejestrach sprzętowych określonych wartości, które będą wykorzystywane potem przez OS. Czynność ta nazywana jest inicjowaniem i wykonywana jest przez kilkanaście procedur wywoływanych kolejno podczas startu systemu.

Jako ostatnia wykonywana jest procedura inicjowania cartridge'a. (jeśli jest dołączony). Jest ona umieszczona w cartridge'u, a jej wektor zapisany jest pod adresem $BFFE. Ponieważ znajduje się ona w cartridge'u, to nie może być tutaj opisana.

2.3.1. Procedury rozpoznania cartridge'a i RAM

Przed zainicjowaniem system musi sprawdzić, jakimi dysponuje obszarami pamięci RAM i dodatkowej pamięci ROM (cartridge). Do tego celu służą procedury CARTGO i GRAMHI.

Procedura CARTGO sprawdza najpierw obecność cartridge'a w gnieździe i jeśli jest, to jego rodzaj. W przypadku stwierdzenia obecności cartridge'a diagnostycznego (ustawiony bit 7 rejestru CARTOPT) dalsze sterowanie jest przekazywane do niego. W przeciwnym razie dokonywane jest inicjowanie portów I/O poprzez procedurę IOPORTINI.

Teraz określany jest status wbudowanego interpretera Basica. Jest on sterowany poprzez bit 1 w PORTB. Jeżeli bit ten jest skasowany (równy zero), to pamięć ROM zawierająca interpreter jest dostępna w obszarze $A000-$BFFF. Gdy bit 1 w PORTB jest ustawiony (równy jeden), układ zarządzania pamięcią blokuje dostęp do interpretera.

W tej fazie interpreter jest najpierw odłączany. Jeżeli jest to start zimny, sprawdzany jest rejestr CONSOL sygnalizujący stan klawiszy konsoli (START, SELECT i OPTION). Klawiszom tym są przyporządkowane odpowiednio bity 0, 1 i 2. Normalnie bity te są ustawione i rejestr CONSOL zawiera wartość $07. Naciśnięcie któregoś klawisza kasuje odpowiadający mu bit. Gdy bit 2 CONSOL jest ustawiony (klawisz OPTION nie wciśnięty), interpreter Basica jest ponownie włączany przez skasowanie bitu 1 PORTB.

Podczas startu gorącego włączenie Basica następuje po stwierdzeniu wartości zero w rejestrze BASICF, który otrzymał odpowiednią wartość podczas wcześniejszej fazy procedury RESET.

```
0100 ;CARTridge GOes ?
0110 ;
0120 BASICF = $03F8
0130 CARTINI = $BFFE
0140 CARTINS = $BFFC
0150 CARTOPT = $BFFD
0160 CONSOL = $D01F
0170 IOPORTINI = $C4DA
0180 PORTB = $D301
0190 RAMLO = $04
0200 TRAMSZ = $06
0210 TRIG3 = $D013
0220 WARMST = $08
0230 ;
0240     *=  $C471
0250 ;
0260     LDA TRIG3
0270     ROR A
0280     BCC PINI
0290     LDA CARTINS
0300     BNE PINI
0310     LDA CARTOPT
0320     BPL PINI
0330     JMP (CARTINI)
0340 PINI JSR IOPORTINI
0350     LDA PORTB
0360     ORA #$02
0370     STA PORTB
0380     LDA WARMST
0390     BEQ CNSL
0400     LDA BASICF
0410     BNE GRAMHI
0420     BEQ ON
0430 CNSL LDA CONSOL
0440     AND #$04
0450     BEQ GRAMHI
0460 ON  LDA PORTB
0470     AND #$FD
0480     STA PORTB
0490 ;
0500 ;Get RAM HIgh
0510 ;
0520 GRAMHI LDA #$00
0530     TAY
```

```
0540     STA RAMLO+1
0550     LDA #$28
0560     STA TRAMSZ
0570 LOOP LDA (RAMLO+1),Y
0580     EOR #$FF
0590     STA (RAMLO+1),Y
0600     CMP (RAMLO+1),Y
0610     BNE END
0620     EOR #$FF
0630     STA (RAMLO+1),Y
0640     CMP (RAMLO+1),Y
0650     BNE END
0660     INC TRAMSZ
0670     BNE LOOP
0680 END RTS
```

CARTGO nie kończy się instrukcją RTS (powrót z procedury), lecz przechodzi bezpośrednio w procedurę GRAMHI. Sprawdzana jest tu wielkość dostępnej pamięci RAM mierzona w stronach (po 256 B). Pierwszy bajt każdej strony pamięci jest dwukrotnie poddawany operacji EOR (exclusive-or) z wartością $FF i za każdym razem porównywany ze swoją poprzednią wartością. Jeśli operacja EOR nie powoduje zmiany zawartości komórki, oznacza to, że nie jest to komórka pamięci RAM. Aktualnie obliczona liczba stron pamięci RAM jest zapisywana tymczasowo do rejestru TRAMSZ.

Warto zauważyć, że przyjęty sposób testowania pamięci RAM nie powoduje zmiany jej zawartości, dzięki czemu zapisany w pamięci program nie ulega skasowaniu podczas gorącego startu. Samo sprawdzanie wielkości RAM jest konieczne z powodu istnienia różnych odmian komputerów w serii XL i XE oraz możliwości zainstalowania cartridge'a o wielkości 8 lub 16 KB. Na przykład model 600XL, mający identyczny system operacyjny, posiada jedynie 16 KB RAM.

2.3.2. Procedura inicjowania portów I/O.

Gdy nie została stwierdzona obecność cartridge'a diagnostycznego, to z CARTGO jest wywoływana procedura IOPORTINI. Jej zadaniem jest zainicjowanie portów I/O. Są to sprzętowe rejestry wewnętrznych, specjalizowanych układów komputera, które służą do obsługi urządzeń zewnętrznych i odciążają w tej pracy jednostkę centralną (procesor 6502).

```
0100 ;I/O PORT INItiation
0110 ;
0120 ANTIC = $D400
0130 AUDC3 = $D205
0140 AUDC4 = $D207
0150 AUDCTL = $D208
0160 GTIA = $D000
0170 PACTL = $D302
0180 PBCTL = $D303
0190 PIA = $D300
0200 POKEY = $D200
0210 PORTA = $D300
0220 PORTB = $D301
0230 SEROUT = $D20D
```

```
0240 SKCTL = $D20F
0250 ;
0260     *=  $C4DA
0270 ;
0280     LDA #$00
0290     TAX
0300     STA PBCTL
0310 NEXT STA GTIA,X
0320     STA ANTIC,X
0330     STA POKEY,X
0340     CPX #$01
0350     BEQ BYPASS
0360     STA PIA,X
0370 BYPASS INX
0380     BNE NEXT
0390     LDA #$3C
0400     STA PBCTL
0410     LDA #$FF
0420     STA PORTB
0430     LDA #$38
0440     STA PACTL
0450     STA PBCTL
0460     LDA #$00
0470     STA PORTA
0480     LDA #$FF
0490     STA PORTB
0500     LDA #$3C
0510     STA PACTL
0520     STA PBCTL
0530     LDA PORTB
0540     LDA PORTA
0550     LDA #$22
0560     STA SKCTL
0570     LDA #$A0
0580     STA AUDC3
0590     STA AUDC4
0600     LDA #$28
0610     STA AUDCTL
0620     LDA #$FF
0630     STA SEROUT
0640     RTS
```

W pierwszej fazie procedury zerowane są wszystkie rejestry układów GTIA, ANTIC i POKEY oraz rejestry układu PIA z wyjątkiem PORTB. Wyzerowanie tego rejestru, sterującego zarządzaniem obszarami pamięci komputera, spowodowałoby zawieszenie się systemu z powodu odłączenia pamięci ROM zawierającej system operacyjny.

Najdłuższa część procedury przeznaczona jest na ustalenie stanu początkowego rejestrów układu PIA. Znaczna długość tej fazy jest spowodowana specyficznym działaniem układu PIA, w którym rejestry PORTA i PORTB są sterowane pośrednio poprzez rejestry PACTL i PBCTL. W rezultacie przeprowadzonych operacji port A ustawiany jest jako rejestr wejściowy, a port B jako wyjściowy.

Ostatnią czynnością jest nadanie wartości początkowych rejestrom POKEY-a, które obsługują

zegary transmisji danych przez złącze szeregowe oraz z klawiatury. Są to rejestry kontroli generatorów dźwięku AUDC3 i AUDC4 oraz rejestr kontrolujący wszystkie generatory AUDCTL (dokładny opis ich funkcji znajduje się w książce "Mapa pamięci Atari XL/XE. Procedury wejścia/wyjścia" (SOETO, 1988). Od tej chwili komputer może komunikować się z urządzeniami zewnętrznymi. Do prawidłowej pracy każde z tych urządzeń wymaga jednak jeszcze oddzielnego zainicjowania, co wykonuje procedura SYSINIT.

2.3.3. Procedura inicjowania systemu

Najważniejszą z procedur wywoływanych przez RESET jest procedura SYSINIT, której zadaniem jest właściwe zainicjowanie pracy całego systemu operacyjnego. Po jej wykonaniu komputer powinien być gotowy do pracy.

```
0100 ;SYStem INITiation
0110 ;
0120 BREAKIRQ = $C092
0130 CASVEC = $E440
0140 CKEY = $03E9
0150 CONSOL = $D01F
0160 EDTVEC = $E400
0170 IRQSTAT = $11
0180 JCIOINIT = $E46E
0190 JDSKINIT = $E450
0200 JNMIEN = $E46B
0210 JSIOINIT = $E465
0220 KBDVEC = $E420
0230 MEMLO = $02E7
0240 MEMTOP = $02E5
0250 NEWINITC = $E49B
0260 NEWIOREQ = $C96E
0270 PRTVEC = $E430
0280 RAMSIZ = $02E4
0290 SCRVEC = $E410
0300 TRAMSZ = $06
0310 VBRKEY = $0236
0320 VPIRQ = $0238
0330 ;
0340     *=  $C535
0350 ;
0360     DEC IRQSTAT
0370     LDA # <BREAKIRQ
0380     STA VBRKEY
0390     LDA # >BREAKIRQ
0400     STA VBRKEY+1
0410     LDA TRAMSZ
0420     STA RAMSIZ
0430     STA MEMTOP+1
0440     LDA #$00
0450     STA MEMTOP
0460     LDA #$00
0470     STA MEMLO
0480     LDA #$07
0490     STA MEMLO+1
0500     JSR EDTVEC+$0C
0510     JSR SCRVEC+$0C
```

```
0520     JSR KBDVEC+$0C
0530     JSR PRTVEC+$0C
0540     JSR CASVEC+$0C
0550     JSR JCIOINIT
0560     JSR JSIOINIT
0570     JSR JNMIEN
0580     JSR JDSKINIT
0590     LDA # <NEWIOREQ
0600     STA VPIRQ
0610     LDA # >NEWIOREQ
0620     STA VPIRQ+1
0630     JSR NEWINITC
0640     LDA CONSOL
0650     AND #$01
0660     EOR #$01
0670     STA CKEY
0680     RTS
```

W pierwszym kroku procedury kasowane są wszystkie żądania przerwań maskowalnych IRQ przez ustawienie rejestru IRQSTAT na wartość $FF ($00-1=$FF). Następnie wektor przerwania wywoływanego przez klawisz BREAK ustawiany jest na adres procedury BREAKIRQ.

Ustalona poprzednio wielkość pamięci RAM (zapisana w TRAMSZ) jest teraz przepisywana do rejestrów MEMTOP i RAMSIZ, przy czym ten drugi zawiera tylko liczbę stron pamięci. Do rejestru MEMLO jest natomiast wpisywana wartość $0700, która określa najniższy adres pamięci dostępny dla oprogramowania. Ma to na celu ochronę obszaru zmiennych oraz buforów systemowych, a także rejestrów wykorzystywanych przez oprogramowanie.

Teraz kolejno wywoływane są procedury inicjowania urządzeń zewnętrznych, a następnie systemowych procedur wejścia/wyjścia oraz przerwań niemaskowalnych. Do wektora VPIRQ (Vector Parallel IRQ) jest przy tym wpisywany adres procedury NEWIOREQ, która obsługuje przerwania wywoływane przez nowe urządzenia.

Ostatnią czynnością jest sprawdzenie w rejestrze CONSOL, czy został wciśnięty klawisz START. Jeżeli tak, to znacznik CKEY otrzymuje wartość 1, w przeciwnym wypadku 0.

2.3.4. Procedura inicjowania edytora

Procedura ta jest wywoływana trzykrotnie w celu zainicjowania kolejno edytora, klawiatury i ekranu. Jej wywołanie odbywa się pośrednio poprzez adresy skoku umieszczone w tablicy procedur obsługi. Dla wymienionych urządzeń adresy te są aktualnie jednakowe i wskazują procedurę POWERON, jednak takie rozwiązanie umożliwia łatwe wprowadzenie zmian w ewentualnych, następnych wersjach OS. Jest to jeden z wielu przykładów dalekowzroczności projektantów Atari.

```
0100 ;POWER ON action
0110 ;
0120 FKDEF = $FC11
0130 FKDEFP = $60
0140 KBCODES = $02FC
```

```
0150 KEYDEF = $FB51
0160 KEYDEFP = $79
0170 RAMSIZ = $02E4
0180 RAMTOP = $6A
0190 SHFOLK = $02BE
0200 ;
0210    *=  $EF6E
0220 ;
0230    LDA #$FF
0240    STA KBCODES
0250    LDA RAMSIZ
0260    STA RAMTOP
0270    LDA #$40
0280    STA SHFOLK
0290    LDA # <KEYDEF
0300    STA KEYDEFP
0310    LDA # >KEYDEF
0320    STA KEYDEFP+1
0330    LDA # <FKDEF
0340    STA FKDEFP
0350    LDA # >FKDEF
0360    STA FKDEFP+1
0370    RTS
```

Procedura POWERON kolejno ustawia wszystkie parametry niezbędne do pracy edytora (klawiatury i ekranu). Do rejestru KBCODES wpisywana jest wartość $FF, która oznacza, że nie został naciśnięty żaden klawisz. Wielkość dostępnej pamięci (w stronach) przepisywana jest z RAMSIZ do RAMTOP. Wartość $40 umieszczona w SHFOLK oznacza, że klawiatura pracuje w trybie dużych liter.

Na zakończenie wektory KEYDEFP i FKDEFP są ustawiane na wartości adresów zawartych w pamięci ROM tabel zawierających kody klawiszy. Tabela FKDEF dotyczy klawiszy funkcyjnych F1-F4, które występują jedynie w modelu 1200XL. Ponieważ system operacyjny był pisany z myślą o całej serii XL, to tabela ta znajduje się we wszystkich modelach, lecz nie jest wykorzystywana poza 1200XL.

2.3.5. Procedura inicjowania drukarki

Zainicjowanie drukarki polega jedynie na ustawieniu w rejestrze PTIMOT wartości $1E. Określa ona tzw. Timeout, czyli czas oczekiwania komputera na odpowiedź po wysłaniu polecenia do urządzenia zewnętrznego i jest różna dla różnych urządzeń. Jeżeli po odliczeniu wskazanej przez Timeout liczby cykli zegarowych komputer nie otrzyma odpowiedzi z urządzenia, to wywoływany jest błąd o kodzie 138 (przekroczony czas oczekiwania). Używanie wartości Timeout przy komunikacji z urządzeniami peryferyjnymi pozwala na współpracę komputera z urządzeniami o różnej szybkości pracy i jest kolejnym przykładem elastyczności systemu.

```
0100 ;PRinter INITiation
0110 ;
0120 PTIMOT = $0314
0130 ;
```

```
0140     *=  $FE99
0150 ;
0160     LDA #$1E
0170     STA PTIMOT
0180     RTS
```

2.3.6. Procedura inicjowania magnetofonu

Ponieważ magnetofon jest jedynym "nieinteligentnym" urządzeniem zewnętrznym, tzn. nie ma wbudowanych układów do komunikacji z komputerem, to komputer wysyła polecenia i dane bez oczekiwania na odpowiedź lub potwierdzenie. Z tego powodu nie jest określany czas oczekiwania na odpowiedź (Timeout). Inicjowanie magnetofonu polega więc jedynie na ustawieniu w rejestrze CBAUD (Cassette BAUD rate) szybkości transmisji na 600 bodów (bitów na sekundę).

```
0100 ;CASsette INITiation
0110 ;
0120 CBAUD = $02EE
0130 ;
0140     *=  $FCDB
0150 ;
0160     LDA #$CC
0170     STA CBAUD
0180     LDA #$05
0190     STA CBAUD+1
0200     RTS
```

Stosowana w Atari szybkość transmisji z magnetofonu nie jest oszałamiająca (np. w Commodore 64 stosuje się 3800 bodów). Dlatego też istnieje wiele programów przyspieszających zapis i odczyt. Ingerują one między innymi w zawartość rejestru CBAUD.

2.3.7. Inicjowanie bloku kontroli urządzeń

Blok kontroli urządzeń (Device Control Block) jest wykorzystywany przez procedurę SIO przede wszystkim do komunikacji ze stacją dysków elastycznych. Przygotowanie systemu do współpracy ze stacją dysków wymaga określenia czasu oczekiwania na odpowiedź (Timeout), tak jak dla drukarki. Dodatkowo ustalana jest na 128 bajtów długość sektora zapisanego na dyskietce.

```
0100 ;DiSK INITiation
0110 ;
0120 DSCTLN = $02D5
0130 DSKTIM = $0246
0140 ;
0150     *=  $C6A3
0160 ;
0170     LDA #$A0
0180     STA DSKTIM
0190     LDA #$80
0200     STA DSCTLN
0210     LDA #$00
0220     STA DSCTLN+1
0230     RTS
```

2.3.8. Inicjowanie centralnej procedury I/O

Wykonywane przez procedurę CIOINIT zainicjowanie centralnej procedury obsługi operacji wejścia/wyjścia polega jedynie na wpisaniu do wszystkich bloków kontroli (IOCB) dwóch wartości. Identyfikator urządzenia (ICCHID) otrzymuje wartość $FF, co oznacza urządzenie nieistniejące. Jako adres procedury Put Byte (odczytu lub zapisu bajtu - zależnie od sposobu dostępu) umieszczany jest adres procedury CIONOPN.

```
0100 ;Central I/O INITiation
0110 ;
0120 ICCHID = $0340
0130 ICPUTB = $0346
0140 ;
0150     *=  $E4C1
0160 ;
0170     LDX #$00
0180 LOOP LDA #$FF
0190     STA ICCHID,X
0200     LDA # <CIONOPN-1
0210     STA ICPUTB,X
0220     LDA # >CIONOPN
0230     STA ICPUTB+1,X
0240     TXA
0250     CLC
0260     ADC #$10
0270     TAX
0280     CMP #$80
0290     BCC LOOP
0300     RTS
0310 ;
0320 ;CIO Not OPeN
0330 ;
0340 CIONOPN LDY #$85
0350     RTS
```

Próba odczytu lub zapisu przez wywołanie procedury CIO spowoduje teraz umieszczenie w rejestrze Y wartości $85, czyli wystąpieniu błędu "NOT OPEN" (kanał nie został otwarty).

2.3.9. Inicjowanie szeregowej procedury I/O

Przygotowanie do pracy procedury SIO, która bezpośrednio obsługuje transmisję poprzez szynę szeregową, wymaga jedynie wpisania odpowiednich wartości do rejestru kontroli złącza szeregowego i klawiatury (Serial and Keyboard ConTroL) oraz jego odpowiednika w pamięci RAM (tzw. rejestr-cień). Poza tym procedura inicjowania ustawia znacznik zezwolenia na dźwięk podczas transmisji, a do rejestrów kontroli portów układu PIA wpisuje wartości, które ustawiają PORTA i PORTB jako rejestry przesyłania danych.

```
0100 ;Serial I/O INITiation
0110 ;
0120 IOSNDEN = $41
0130 PACTL = $D302
0140 PBCTL = $D303
0150 SKCTL = $D20F
```

```
0160 SKCTLS = $0232
0170 ;
0180     *=  $E95C
0190 ;
0200     LDA #$3C
0210     STA PACTL
0220     LDA #$3C
0230     STA PBCTL
0240     LDA #$03
0250     STA SKCTLS
0260     STA IOSNDEN
0270     STA SKCTL
0280     RTS
```

2.3.10. Inicjowanie nowych urządzeń

System operacyjny Atari był projektowany tak, aby umożliwić maksymalną elastyczność. Szybki rozwój techniki czynił oczywistym powstanie w przyszłości urządzeń, których działanie trudno było przewidzieć podczas tworzenia systemu. Z tego powodu tabela procedur obsługi urządzeń zewnętrznych została umieszczona w pamięci RAM i przewidziano możliwość jej rozszerzania do jedenastu wpisów.

Sposób ten jest stosowany także w innych komputerach, projektanci Atari poszli jednak jeszcze dalej! Założyli powstanie w przyszłości urządzeń zewnętrznych korzystających z szyny równoległej komputera i nazwali je "nowymi urządzeniami" (new device). Do ich obsługi przewidziano liczne procedury systemu operacyjnego. Procedury te przez kilka lat stanowiły powód do zdziwienia, a nawet kpin. Przewidywania twórców Atari okazały się jednak słuszne: jest już pierwsze urządzenie korzystające z tych procedur - twardy dysk 20 MB firmy Supra Corp.

Chociaż na etapie projektowania nie można było przewidzieć nawet zasady działania tych urządzeń, to jednak trzeba było ustalić kilka podstawowych warunków. Przede wszystkim w obszarze portów I/O wydzielono część przestrzeni adresowej dla rejestrów nowych urządzeń ($D100-$D1FF). Następnie ustalono, że nowe urządzenie będzie powodowało odłączenie obszaru ROM zawierającego procedury operacji zmiennoprzecinkowych ($D800-$DFFF). W uzyskanej w ten sposób przestrzeni adresowej zostanie sprzętowo (przez przełączenie linii magistrali adresowej) umieszczona część pamięci ROM urządzenia.

Po tej długiej, lecz koniecznej dygresji czas na opis procedury inicjowania nowych urządzeń.

```
0100 ;NEW device INITiation
0110 ;
0120 DEVID1 = $D803
0130 DEVID2 = $D80B
0140 DEVINIT = $D819
0150 PDVREG = $D1FF
0160 PDVRS = $0248
0170 ;
0180     *=  $C90C
```

```
0190 ;
0200     LDA #$01
0210     STA PDVRS
0220 DEV LDA PDVRS
0230     STA PDVREG
0240     LDA DEVID1
0250     CMP #$80
0260     BNE NEXT
0270     LDA DEVID2
0280     CMP #$91
0290     BNE NEXT
0300     JSR DEVINIT
0310 NEXT ASL PDVRS
0320     BNE DEV
0330     LDA #$00
0340     STA PDVREG
0350     RTS
```

Procedura kolejno aktywizuje nowe urządzenia (może być ich osiem) i sprawdza dwa bajty identyfikacyjne (DEVID1=$80 i DEVID2=$91). Jeżeli wynik jest pozytywny, to wywoływana jest procedura inicjowania urządzenia zapisana w jego pamięci (od adresu DEVINIT). Uaktywnienie urządzenia polega na ustawieniu bitu o odpowiadającym mu numerze (od 0 do 7) w rejestrze PDVREG (Parallel DeVice REGister) znajdującym się w tym urządzeniu. Rejestr ten posiada w obszarze zmiennych systemowych komputera swój rejestr-cień PDVRS.

2.3.11. Inicjowanie przerwań niemaskowalnych

Po przygotowaniu do pracy całego systemu wykonywana jest procedura uruchamiająca przerwania niemaskowalne (NMI - Non Maskable Interrupt). Umieszcza ona w rejestrze NMIEN (NMI ENable) wartość $40, co oznacza zezwolenie na przerwania wywoływane przez impuls synchronizacji pionowej obrazu (zob. rozdział 3.1.1.). Dodatkowo przepisuje ona wartość wskaźnika TRIG3, sygnalizującego obecność cartridge'a do rejestru GINTLK.

```
0100 ;NMI ENaBLe
0110 ;
0120 GINTLK = $03FA
0130 NMIEN = $D40E
0140 TRIG3 = $D013
0150 ;
0160     *=  $C00C
0170 ;
0180     LDA #$40
0190     STA NMIEN
0200     LDA TRIG3
0210     STA GINTLK
0220     RTS
```

2.4. Procedury odczytu wstępnego (boot)

Po zainicjowaniu pracy wszystkich niezbędnych elementów systemu procedura RESET musi przekazać sterowanie komputera do programu użytkownika. W tym celu po zimnym starcie próbuje dokonać odczytu z magnetofonu i stacji dysków, a po starcie gorącym sprawdza, czy taki odczyt był

uprzednio wykonany. Pozytywny wynik (odczytu lub sprawdzenia) powoduje skok do adresu wskazanego przez wektor inicjowania odczytanego programu. W tym miejscu pojawia się największa możliwość ingerencji w procedurę RESET i wykonania programu wskazanego przez programistę.

2.4.1. Procedura odczytu z magnetofonu

Jako pierwsza wywoływana jest procedura odczytu z magnetofonu.

```
0100 ;CASsette BOOT
0110 ;
0120 BLOCK1 = $C5BB
0130 BOOT? = $09
0140 CASINI = $02
0150 CASSBT = $03EA
0160 CKEY =  $03E9
0170 DOSINI = $0C
0180 GAPTYP = $3E
0190 JCASOPIN = $E47D
0200 WARMST = $08
0210 ;
0220     *=  $C66E
0230 ;
0240     LDA WARMST
0250     BEQ RD
0260     LDA BOOT?
0270     AND #$02
0280     BEQ EXIT
0290     JMP CASINITC
0300 RD  LDA CKEY
0310     BEQ EXIT
0320     LDA #$80
0330     STA GAPTYP
0340     INC CASSBT
0350     JSR JCASOPIN
0360     JSR BLOCK1
0370     LDA #$00
0380     STA CASSBT
0390     STA CKEY
0400     ASL BOOT?
0410     LDA DOSINI
0420     STA CASINI
0430     LDA DOSINI+1
0440     STA CASINI+1
0450 EXIT RTS
0460 CASINITC JMP (CASINI)
```

Jeśli jest to start gorący (WARMST=$FF), to kontrolowany jest znacznik BOOT?. Gdy ma on ustawiony bit 1, wykonywany jest skok do programu, którego adres wskazuje wektor CASINI. W przeciwnym wypadku następuje wyjście z procedury CASBOOT i powrót do RESET.

Przy zimnym starcie systemu najpierw sprawdzany jest znacznik CKEY, określający, czy został naciśnięty klawisz START. Zerowa wartość tego znacznika oznacza brak odczytu z magnetofonu i

powoduje opuszczenie CASBOOT.

Po stwierdzeniu konieczności odczytu z magnetofonu rejestr GAPTYP ustawiany jest na $80, co oznacza krótkie przerwy między rekordami zapisanymi na kasecie oraz zwiększana jest wartość wskaźnika CASSBT (z 0 na 1). Następnie wywoływana jest procedura CASOPIN, której zadaniem jest otworzenie kanału IOCB do odczytu z magnetofonu. Teraz poprzez skok do środka procedury BOOT (zob. niżej), w miejsce oznaczone etykietą BLOCK1, wykonywany jest właściwy odczyt z magnetofonu.

Na zakończenie zerowane są wskaźniki CASSBT i CKEY, a BOOT? jest mnożony przez 2. Ponieważ procedura BOOT umieszcza wektor startowy programu w rejestrze DOSINI, a po odczycie z magnetofonu wykorzystywany jest wektor CASINI, to trzeba jeszcze przepisać jego zawartość (z DOSINI do CASINI).

2.4.2. Procedura odczytu ze stacji dysków

Procedura wstępnego odczytu ze stacji dysków rozpoczyna się tak samo jak odczyt z magnetofonu. Jedynie zamiast bitu 1 przy gorącym starcie sprawdzany jest bit 0 wskaźnika BOOT?, sygnalizujący poprawny odczyt z dyskietki. Ustawienie tego bitu powoduje skok do adresu zawartego w rejestrze DOSINI.

```
0100 ;BOOT
0110 ;
0120 BLOAD = $C629
0130 BOOT? = $09
0140 BOOTAD = $0242
0150 CASBUF = $0400
0160 CASSBT = $03EA
0170 DAUX1 = $030A
0180 DAUX2 = $030B
0190 DBUFA = $0304
0200 DCMND = $0302
0210 DFLAG = $0240
0220 DOSINI = $0C
0230 DOSINITC = $C63B
0240 DRDERR = $C63E
0250 DSECCNT = $0241
0260 DUNIT = $0301
0270 GETBLK = $C659
0280 JDSKINT = $E453
0290 RAMLO = $04
0300 WARMST = $08
0310 ;
0320     *=  $C58B
0330 ;
0340     LDA WARMST
0350     BEQ RD
0360     LDA BOOT?
0370     AND #$01
0380     BEQ EXIT
0390     JMP DOSINITC
```

```
0400 RD  LDA #$01
0410     STA DUNIT
0420     LDA #$53
0430     STA DCMND
0440     JSR JDSKINT
0450     BMI EXIT
0460 RP1 LDA #$00
0470     STA DAUX2
0480     LDA #$01
0490     STA DAUX1
0500     LDA # <CASBUF
0510     STA DBUFA
0520     LDA # >CASBUF
0530     STA DBUFA+1
0540 BLOCK1 JSR GETBLK
0550     BPL NER
0560 ERR JSR DRDERR
0570     LDA CASSBT
0580     BEQ RP1
0590 EXIT RTS
0600 NER LDX #$03
0610 LOOP1 LDA CASBUF,X
0620     STA DFLAG,X
0630     DEX
0640     BPL LOOP1
0650     LDA BOOTAD
0660     STA RAMLO
0670     LDA BOOTAD+1
0680     STA RAMLO+1
0690     LDA CASBUF+4
0700     STA DOSINI
0710     LDA CASBUF+5
0720     STA DOSINI+1
0730 NXT LDY #$7F
0740 LOOP2 LDA CASBUF,Y
0750     STA (RAMLO),Y
0760     DEY
0770     BPL LOOP2
0780     CLC
0790     LDA RAMLO
0800     ADC #$80
0810     STA RAMLO
0820     LDA RAMLO+1
0830     ADC #$00
0840     STA RAMLO+1
0850     DEC DSECCNT
0860     BEQ END
0870     INC DAUX1
0880 RP2 JSR GETBLK
0890     BPL NXT
0900     JSR DRDERR
0910     LDA CASSBT
0920     BNE ERR
0930     BEQ RP2
0940 END LDA CASSBT
0950     BEQ BLD
0960     JSR GETBLK
0970 BLD JSR BLOAD
0980     BCS ERR
```

```
0990     JSR DOSINITC
1000     INC BOOT?
1010     RTS
```

W przypadku zimnego startu zawsze podejmowana jest próba odczytu ze stacji dysków numer 1. W tym celu do rejestru DUNIT wpisywany jest numer stacji (1), a do DCMND kod rozkazu odczytu statusu ($53). Potem wywoływana jest procedura DSKINT i po uzyskaniu negatywnego wyniku (brak odpowiedzi od stacji) następuje opuszczenie procedury BOOT.

Gdy została stwierdzona gotowość do pracy stacji numer 1 (status dodatni), numer sektora jest ustawiany na 1, a jako adres bufora do umieszczenia odczytanego bloku wskazywany jest bufor magnetofonu (CASBUF). W tym miejscu (od etykiety BLOCK1) następuje wejście z procedury CASBOOT - ta część jest wspólna dla odczytu z obu urządzeń. Teraz wywoływana jest procedura GETBLK, która pobiera blok bajtów z urządzenia zewnętrznego. Jeśli podczas odczytu wystąpił błąd, to wywoływana jest procedura DRDERR i w przypadku magnetofonu następuje zakończenie odczytu, a w przypadku stacji dysków próba odczytu jest ponawiana (aź do skutku). Do rozróżnienia urządzenia, z którego dokonywany jest odczyt, służy tu znacznik CASSBT (CASSette BooT). Przy odczycie ze stacji dysków ma on wartość zero, a przy odczycie z magnetofonu procedura CASBOOT nadaje mu wartość $01.

Po bezbłędnym odczycie bloku pierwsze sześć bajtów z bufora jest przenoszone do różnych rejestrów: bajt 1 do DFLAG, bajt 2 do DSECCNT, bajty 3 i 4 do BOOTAD i RAMLO oraz bajty 5 i 6 do DOSINI. Dzięki temu DSECCNT (Disc SECtor CouNTer) zawiera liczbę bloków do odczytu, BOOTAD (BOOT ADdress) i RAMLO adres początkowy umieszczenia odczytanych bloków w pamięci, a DOSINI adres inicjowania odczytanego programu. Ostatnim krokiem tego etapu jest przeniesienie zawartości bufora magnetofonu do miejsca ładowania wskazanego wektorem RAMLO.

Dalszy przebieg odczytu jest podobny. Kolejno, aż do wyzerowania licznika DSECCNT, bloki danych są odczytywane do bufora magnetofonu i stąd przenoszone na miejsce wskazane przez RAMLO. Wektor RAMLO jest zwiększany po odczycie każdego bloku o 128, co zapewnia zapisanie odczytanych bloków w pamięci bezpośrednio jeden za drugim. Wystąpienie błędu w którymkolwiek momencie odczytu powoduje wywołanie procedury DRDERR i przerwanie odczytu z magnetofonu lub ponowny odczyt (od początku, tj. od pierwszego sektora) ze stacji dysków.

Po zakończeniu odczytu wywoływana jest procedura BLOAD oraz procedura inicjowania (od adresu umieszczonego w DOSINI). Przed opuszczeniem procedury BOOT zwiększana jest jeszcze o jeden zawartość wskaźnika BOOT?.

2.4.3. Pomocnicze procedury odczytu

Procedury wstępnego odczytu CASBOOT i BOOT wykorzystują kilka procedur pomocniczych. Procedury DSKINT (Disk INTerface), CIOMAIN (CIO MAIN routine), CASOPIN (CASsette OPen for INput) i CASRDBL (CASsette ReaD BLock) są standardowymi procedurami wejścia/wyjścia i są opisane w książce "Mapa pamięci Atari XL/XE. Procedury wejścia/wyjścia" wydanej przez SOETO. Tu przedstawione będą tylko procedury GETBLK, DRDERR i BLOAD.

```
0100 ;GET BlocK
0110 ;
0120 CASSBT = $03EA
0130 DCMND = $0302
0140 DUNIT = $0301
0150 JCASRDBL = $E47A
0160 JDSKINT = $E453
0170 ;
0180     *=  $C659
0190 ;
0200     LDA CASSBT
0210     BEQ DSK
0220     JMP JCASRDBL
0230 DSK LDA #$52
0240     STA DCMND
0250     LDA #$01
0260     STA DUNIT
0270     JMP JDSKINT
```

Procedura GETBLK rozpoznaje według zawartości rejestru CASSBT rodzaj urządzenia zewnętrznego (stacja dysków czy magnetofon) i przechodzi do procedury odczytu z magnetofonu CASRDBL, albo ustawia numer stacji dysków (1) i kod rozkazu odczytu sektora ($52) oraz przechodzi do procedury operacji dyskowych DSKINT.

```
0100 ;Boot LOADer
0110 ;
0120 BOOTAD = $0242
0130 DOSINI = $0C
0140 RAMLO = $04
0150 ;
0160     *=  $C629
0170 ;
0180     CLC
0190     LDA BOOTAD
0200     ADC #$06
0210     STA RAMLO
0220     LDA BOOTAD+1
0230     ADC #$00
0240     STA RAMLO+1
0250     JMP (RAMLO)
0260 DOSINITC JMP (DOSINI)
```

Zadaniem procedury BLOAD jest wykonanie wstępnych czynności wymaganych przez wczytany program. W tym celu adres ładowania programu pobrany z rejestru BOOTAD jest zwiększany o 6 (liczba bajtów nagłówka) i umieszczany w RAMLO. Wektor RAMLO służy następnie do

wykonania skoku do wczytanego programu (do jego części wstępnej - w przypadku magnetofonu jest to zwykle zatrzymanie silnika).

```
0100 ;Disk ReaD ERRor
0110 ;
0120 ICBUFA = $0344
0130 ICBUFL = $0348
0140 ICCMD = $0342
0150 JCIOMAIN = $E456
0160 ;
0170 ;Disk ERRor MeSsaGe
0180 ;
0190     *=  $C43D
0200 ;
0210 DERRMSG .BYTE "BOOT ERROR"
0220     .BYTE $9B   ;End Of Line
0230 ;
0240 ;Disk ReaD ERRor
0250 ;
0260     *= $C63E
0270 ;
0280     LDX # <DERRMSG
0290     LDY # >DERRMSG
0300 PUTLINE TXA
0310     LDX #$00
0320     STA ICBUFA,X
0330     TYA
0340     STA ICBUFA+1,X
0350     LDA #$09
0360     STA ICCMD,X
0370     LDA #$FF
0380     STA ICBUFL,X
0390     JMP JCIOMAIN
```

Wystąpienie błędu w dowolnej fazie procedury BOOT powoduje wywołanie procedury DRDERR. W rejestrze X umieszczany jest młodszy, a w rejestrze Y starszy bajt adresu tekstu "BOOT ERROR". Następnie są one przepisywane do ICBUFA jako adres bufora dla IOCB 0, a do rejestru ICCMD wpisywany jest kod rozkazu "Put Line" (wyślij linię). Po przejściu do procedury CIOMAIN powoduje to wyświetlenie na ekranie wskazanego napisu.

Procedurę tą można łatwo wykorzystać do pisania na ekranie w trybie 0. Należy wpisać do rejestrów X i Y adres tekstu i wywołać procedurę DRDERR od etykiety PUTLINE ($C642) zamiast od początku. Maksymalna długość tekstu (razem ze znakiem końca linii - RETURN) może wynosić 255 bajtów. Tekst jest umieszczany na ekranie począwszy od aktualnej pozycji kursora.

2.5. Procedura testu komputera

W przypadku wystąpienia błędu podczas inicjowania systemu operacyjnego wywoływana jest procedura TESTROM.

```
0100 ;TEST ROM enable
0110 ;
0120 SWITROM = $C8FC
```

```
0130 ;
0140     *=  $F223
0150 ;
0160     JMP SWITROM
```

Jedyną jej funkcją jest bezpośredni skok do procedury SWITROM. Wydaje się to całkiem bezsensowne, lecz w tym miejscu znajdował się w starych modelach Atari 400/800 początek procedury tzw. "memo pad". Skok ten służy więc zachowaniu adresu pomimo zmiany systemu, gdyż adres ten wykorzystywany jest w niektórych programach.

```
0100 ;SWITch ROM
0110 ;
0120 COLDST = $0244
0130 JTESTST = $E483
0140 PORTB = $D301
0150 ;
0160     *=  $C8FC
0170 ;
0180     LDA #$FF
0190     STA COLDST
0200     LDA PORTB
0210     AND #$7F
0220     STA PORTB
0230     JMP JTESTST
```

Właściwe przejście do testowania komputera odbywa się w procedurze SWITROM. Ustawia ona wskaźnik zimnego startu na $FF (zimny start po RESET) oraz przełącza program testowania pamięci (SELFTEST) do obszaru $5000-$57FF. Na zakończenie wykonuje pośredni skok do procedury testującej.

Rozdział 3

PRZERWANIA SYSTEMOWE

Komputery Atari posiadają bardzo rozbudowany system przerwań, który stanowi jeden z najmocniejszych punktów systemu operacyjnego. Część przerwań jest wykorzystywana bezpośrednio przez OS, a niektóre są przeznaczone do zastosowania przez programy użytkownika. Są też one najczęściej wykorzystywanym przez programistów elementem systemu. Wszystkie występujące w Atari przerwania można podzielić na dwie zasadnicze grupy: przerwania niemaskowalne (NMI - Non Maskable Interrupt) i maskowalne (IRQ - Interrupt ReQuest).

Zgłoszenie przerwania następuje na drodze sprzętowej poprzez przekazanie do CPU sygnału żądania przerwania odpowiednio po linii NMI lub IRQ szyny sterowania. Po otrzymaniu żądania przerwania NMI procesor odkłada na stos aktualną zawartość licznika programu i rejestru statusu oraz wykonuje skok pod adres zawarty w rejestrze NMIVEC ($FFFA). Przerwania NMI nie można zablokować i musi ono zostać wykonane.

Żądanie przerwania IRQ jest obsługiwane podobnie z dwiema istotnymi różnicami. Najpierw sprawdzany jest bit I rejestru statusu i gdy jest on ustawiony, to żądanie przerwania jest ignorowane i procesor kontynuuje wykonywanie programu. Gdy bit I jest skasowany, wykonywane są operacje jak przy NMI, a następnie skok pod adres zawarty w rejestrze IRQVEC ($FFFE). Dzięki zastosowaniu takiego sposobu postępowania przerwania IRQ mogą być zablokowane (zamaskowane) przez ustawienie bitu I w rejestrze statusu procesora.

Wypada dodać, że RESET jest także przerwaniem. Ma ono najwyższy priorytet i zawsze musi być wykonane. Wektor przerwania RESET (RESETVEC) jest umieszczony pod adresem $FFFC, między wektorami NMIVEC i IRQVEC.

3.1. Procedury przerwań niemaskowalnych

Mimo, iż przerwanie niemaskowalne nie może zostać zablokowane, to jednak istnieje możliwość jego zabronienia. Wynika to z budowy wewnętrznej komputera. Żądanie przerwania niemaskowalnego jest wysyłane do CPU przez procesor obrazowy ANTIC. Może on wysłać sygnał żądania przerwania tylko wtedy, gdy ustawiony jest bit 6 lub 7 w jego sprzętowym rejestrze NMIEN (NMI ENable - $D40E).

Ustawienie bitu 6 zezwala na żądanie przerwania wywołanego przez impuls synchronizacji pionowej obrazu (VBLKI - Vertical BLanK Interrupt). Ustawienie bitu 7 zezwala na żądanie przerwania wywołanego przez program ANTIC-a (DLI - Display List Interrupt). Jednocześnie z

żądaniem przerwania ANTIC ustawia odpowiedni bit w rejestrze NMIST (NMI STatus) informującym o źródle przerwania.

3.1.1. Procedura rozpoznania źródła przerwania

Po otrzymaniu sygnału żądania przerwania niemaskowalnego procesor pobiera z rejestru NMIVEC (NMI VECtor) adres procedury obsługi tego przerwania i przechodzi do jej wykonywania od pobranego adresu. Adres ten wskazuje na procedurę NMIFIRST ($C018).

```
0100 ;NMI FIRST
0110 ;
0120 DLIV = $0200
0130 NMIST = $D40F
0140 VVBLKI = $0222
0150 ;
0160     *=  $C018
0170 ;
0180     BIT NMIST
0190     BPL VBL
0200     JMP (DLIV)
0210 VBL CLD
0220     PHA
0230     TXA
0240     PHA
0250     TYA
0260     PHA
0270     STA NMIST
0280     JMP (VVBLKI)
```

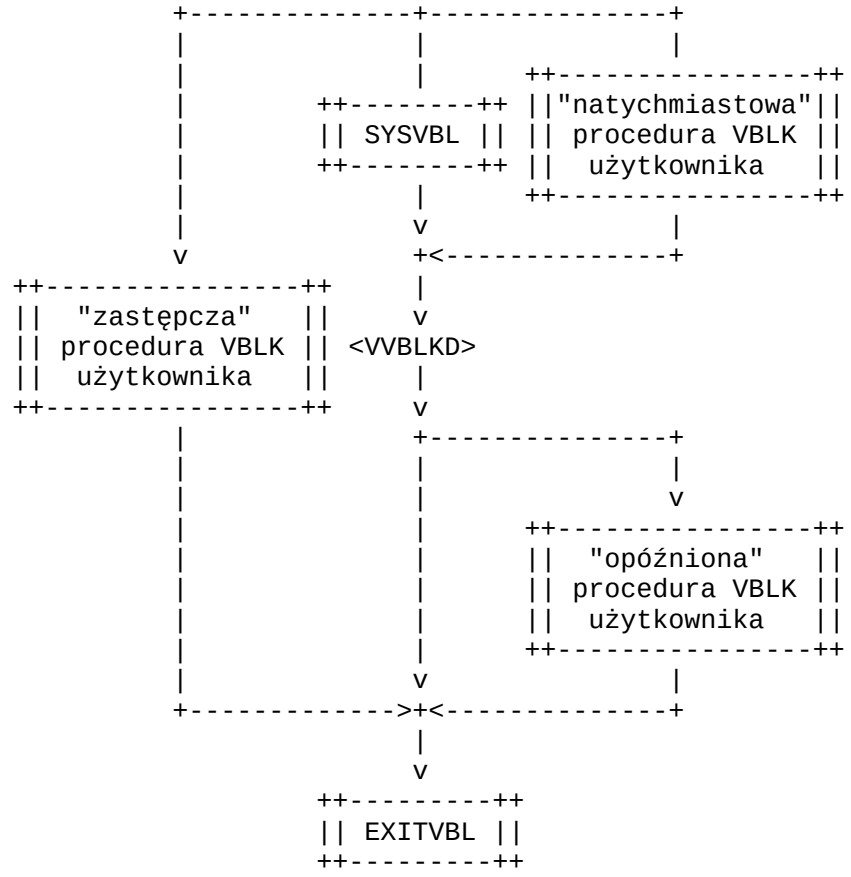
W pierwszym kroku procedury sprawdzany jest najstarszy bit rejestru NMIST. Jeżeli jest on ustawiony, to znaczy, że przerwanie zostało wywołane przez program ANTIC-a i wykonywany jest skok do adresu zawartego w rejestrze DLIV (DLI Vector). Wskazywana przez ten rejestr procedura nie występuje w systemie i musi być zaprogramowana przez użytkownika. Wektor DLIV wskazuje więc normalnie rozkaz RTI.

Gdy bit 7 w NMIST jest skasowany, to znaczy, że chodzi o przerwanie synchronizacji pionowej (Vertical BLanK Interrupt - VBLKI). W takim przypadku kasowany jest tryb dziesiętny procesora, zawartości jego rejestrów są odkładane na stos i rejestr NMIST jest zerowany, aby umożliwić zasygnalizowanie kolejnego przerwania. Następnie wykonywany jest skok do procedury wskazywanej przez wektor VVBLKI (Vector VBLK Immediate).

3.1.2. Struktura przerwania VBLK

Przerwanie synchronizacji pionowej podzielone jest na dwie części. Każda z nich jest wskazywana przez wektor umieszczony w pamięci RAM. Umożliwia to prostą ingerencję programisty w przebieg przerwania. Schemat procedury jest następujący:

```
<VVBLKI>
    |
    v
```



Z powyższego schematu widać, że użytkownik może wstawić swoją procedurę zarówno przed ("natychmiastowa"), jak i po ("opóźniona") procedurze systemowej. Możliwe jest także całkowite ominięcie procedury systemowej przez procedurę użytkownika ("zastępcza"). W tym celu wystarczy jedynie zmienić odpowiednie wektory. Istnieje tu jednak pewne niebezpieczeństwo. Gdy przerwanie zostanie wywołane podczas wykonywania tej zmiany, to system zawiesi się. Dla uniknięcia takiej sytuacji OS zawiera specjalną procedurę SETVBLV, która służy do zmian wektorów.

3.1.3. Systemowa procedura VBLKI

Podstawowym zadaniem procedury przerwania synchronizacji pionowej jest odliczanie czasu i obsługa zegarów systemowych. Przede wszystkim na początku procedury zwiększany jest stan zegara czasu rzeczywistego RTCLOCK (Real Time CLOCK). W tej fazie zwiększany jest także co 256 przerwań rejestr ATRACT i wykonywana jest obsługa tzw. trybu przyciągania uwagi.

Tryb przyciągania uwagi (Attract Mode) jest specyficznym pomysłem twórców Atari. Rejestr ATRACT jest zerowany przez każde naciśnięcie klawisza. Jeżeli jednak osiągnie on wartość $80 (trwa to 32768 cykli zegarowych, czyli 645,36 sekund), to włączany jest tryb Attract. Rejestr ATRACT otrzymuje wartość $FE, rejestr ATRMSK wartość $F6 (zamiast $FE), a rejestr COLRSH wartość drugiego bajtu zegara RTCLOCK. Teraz wartość koloru pobrana z rejestru-cienia jest

poddawana operacji EOR z zawartością COLRSH i AND z ATRMSK oraz zapisywana do rejestru koloru. Powoduje to cykliczne zmiany kolorów na ekranie.

Następnie z wartością zero w rejestrze X wywoływana jest procedura DECTIM w celu zmniejszenia stanu licznika systemowego TIMCNT1 (TIMer CouNTer 1). Powrót z tej procedury z ustawionym bitem Z w rejestrze statusu oznacza wyzerowanie licznika i powoduje wywołanie poprzez skok do JMPTIM1 procedury od adresu wskazanego przez wektor TIMVEC1 (TIMer VECtor 1).

Teraz sprawdzany jest znacznik CRITIC (CRITICal I/O). Jeśli jego wartość jest różna od zera, oznacza to, że wykonywana jest przez system operacja wejścia/wyjścia krytyczna czasowo. W takim przypadku procedura przerwania VBLK jest kończona skokiem bezpośrednio do procedury EXITVBL.

```
0100 ;SYStem Vertical
0110 ;BLank interrupt
0120 ;
0130 ATRACT = $4D
0140 ATRMSK = $4E
0150 CHACT = $02F3
0160 CHBAS = $02F4
0170 CHBASE = $D409
0180 CHRCTL = $D401
0190 COLPF1 = $D017
0200 COLPF1S = $02C5
0210 COLPM0 = $D012
0220 COLPM0S = $02C0
0230 COLRSH = $4F
0240 CONSOL = $D01F
0250 CRITIC = $42
0260 DECTIM = $C255
0270 DLPTR = $D402
0280 DLPTRS = $0230
0290 DMACTL = $D400
0300 DMACTLS = $022F
0310 EXITVBL = $C28A
0320 GINTLK = $03FA
0330 GTIACTL = $D01B
0340 GTICTLS = $026F
0350 JSTICK0 = $0278
0360 JSTICK1 = $0279

0640 TRIG1S = $0285
0650 TRIG2S = $0286
0660 TRIG3 = $D013
0670 TRIG3S = $0287
0680 VSCROL = $D405
0690 VSFLAG = $026C
0700 VVBLKD = $0224
0710 ;
0720      *=  $C0DF
0730 ;
0740 WAIT JMP WAIT
0370 JSTICK2 = $027A
0380 JSTICK3 = $027B
0390 KBCODE = $D209
0400 KBCODES = $02FC
0410 KEYDEL = $02F1
0420 KEYDIS = $026D
0430 KEYREP = $02DA
0440 LPENH = $D40C
0450 LPENHS = $0234
0460 LPENV = $D40D
0470 LPENVS = $0235
0480 PADDL0 = $0270
0490 PORTA = $D300
0500 POT0 =  $D200
0510 POTGO = $D20B
0520 PTRIG0 = $027C
0530 PTRIG1 = $027D
0540 RTCLOCK = $12
0550 SKSTAT = $D20F
0560 SRTIMER = $022B
0570 STACK = $0100
0580 TIMCNT1 = $0218
0590 TIMVEC1 = $0226
0600 TIMVEC2 = $0228
0610 TRIG0 = $D010
0620 TRIG0S = $0284
0630 TRIG1 = $D011

1200     LDA VSFLAG
1210     BEQ NSC
1220     DEC VSFLAG
1230     LDA #$08
1240     SEC
1250     SBC VSFLAG
1260     AND #$07
1270     STA VSCROL
1280 NSC LDX #$08
1290     STX CONSOL
1300 COL CLI
```

```
0750 SYSVBL INC RTCLOCK+2
0760     BNE ATT
0770     INC ATRACT
0780     INC RTCLOCK+1
0790     BNE ATT
0800     INC RTCLOCK
0810 ATT LDA #$FE
0820     LDX #$00
0830     LDY ATRACT
0840     BPL ATR
0850     STA ATRACT
0860     LDX RTCLOCK+1
0870     LDA #$F6
0880 ATR STA ATRMSK
0890     STX COLRSH
0900     LDA COLPF1S
0910     EOR COLRSH
0920     AND ATRMSK
0930     STA COLPF1
0940     LDX #$00
0950     JSR DECTIM
0960     BNE NTI1
0970     JSR JMPTIM1
0980 NTI1 LDA CRITIC
0990     BNE IOC
1000     TSX
1010     LDA STACK+4,X
1020     AND #$04
1030     BEQ PHASE2
1040 IOC JMP EXITVBL
1050 PHASE2 LDA TRIG3
1060     CMP GINTLK
1070     BNE WAIT
1080     LDA LPENV
1090     STA LPENVS
1100     LDA LPENH
1110     STA LPENHS
1120     LDA DLPTRS+1
1130     STA DLPTR+1
1140     LDA DLPTRS
1150     STA DLPTR
1160     LDA DMACTLS
1170     STA DMACTL
1180     LDA GTICTLS
1190     STA GTIACTL

1310     LDA COLPM0S,X
1320     EOR COLRSH
1330     AND ATRMSK
1340     STA COLPM0,X
1350     DEX
1360     BPL COL
1370     LDA CHBAS
1380     STA CHBASE
1390     LDA CHACT
1400     STA CHRCTL
1410     LDX #$02
1420     JSR DECTIM
1430     BNE NTI2
1440     JSR JMPTIM2
1450 NTI2 LDX #$02
1460 TIM INX
1470     INX
1480     LDA TIMCNT1,X
1490     ORA TIMCNT1+1,X
1500     BEQ NTI3
1510     JSR DECTIM
1520     STA TIMVEC1,X
1530 NTI3 CPX #$08
1540     BNE TIM
1550     LDA SKSTAT
1560     AND #$04
1570     BEQ NKY
1580     LDA KEYDEL
1590     BEQ NKY
1600     DEC KEYDEL
1610 NKY LDA SRTIMER
1620     BEQ JOY
1630     LDA SKSTAT
1640     AND #$04
1650     BNE NOKEY
1660     DEC SRTIMER
1670     BNE JOY
1680     LDA KEYDIS
1690     BNE JOY
1700     LDA KEYREP
1710     STA SRTIMER
1720     LDA KBCODE
1730     CMP #$9F
1740     BEQ JOY
1750     CMP #$83

1760     BEQ JOY
1770     CMP #$84
1780     BEQ JOY
1790     CMP #$94
1800     BEQ JOY
1810     AND #$3F
1820     CMP #$11
1830     BEQ JOY
1840     LDA KBCODE
1850     STA KBCODES
1860     JMP JOY
1870 NOKEY LDA #$00
1880     STA SRTIMER

2040     STA TRIG1S
2050     STA TRIG3S
2060     LDX #$03
2070 PAD LDA POT0,X
2080     STA PADDL0,X
2090     STA PADDL0+4,X
2100     DEX
2110     BPL PAD
2120     STA POTGO
2130     LDX #$02
2140     LDY #$01
2150 PDT LDA JSTICK0,Y
2160     LSR A
```

```
1890 JOY LDA PORTA            2170     LSR A
1900     LSR A                2180     LSR A
1910     LSR A                2190     STA PTRIG1,X
1920     LSR A                2200     STA PTRIG1+4,X
1930     LSR A                2210     LDA #$00
1940     STA JSTICK1          2220     ROL A
1950     STA JSTICK3          2230     STA PTRIG0,X
1960     LDA PORTA            2240     STA PTRIG0+4,X
1970     AND #$0F             2250     DEX
1980     STA JSTICK0          2260     DEX
1990     STA JSTICK2          2270     DEY
2000     LDA TRIG0            2280     BPL PDT
2010     STA TRIG0S           2290     JMP (VVBLKD)
2020     STA TRIG2S           2300 JMPTIM1 JMP (TIMVEC1)
2030     LDA TRIG1            2310 JMPTIM2 JMP (TIMVEC2)
```

Jeżeli nie ma ograniczenia czasowego, to wykonywana jest druga faza procedury przerwania VBLK. Najpierw porównywane są zawartości rejestrów TRIG3 i GINTLK. Gdy są one różne, to znaczy, że został wyjęty lub włożont cartridge. Powoduje to skok do procedury WAIT, której pełna nazwa - WAIT for RESET (Czekaj na RESET) - dobrze wyjaśnia działanie: system zawiesza się i oczekuje na naciśnięcie klawisza RESET.

W dalszej części procedury przerwania rejestry sprzętowe układów komputera uaktualniane są według rejestrów-cieni. Teraz następuje także zmniejszenie zawartości licznika przesuwu pionowego obrazu VSFLAG i przepisanie jej do VSCROL.

Kolejny etap obejmuje czterokrotne wywołanie procedury DECTIM w celu zmniejszenia stanu pozostałych liczników systemowych. W przypadku wyzerowania licznika TIMCNT2 poprzez JMPTIM2 wywoływana jest procedura użytkownika (normalnie nie wykorzystywana przez OS). Wyzerowanie pozostałych liczników jest jedynie sygnalizowane we wskaźnikach TIMFLG3-TIMFLG5 (TIMer FLaG).

Teraz wykonywana jest wstępna obsługa klawiatury. Jeśli rejestr SKSTAT (Serial/Keyboard STATus) sygnalizuje naciśnięcie klawisza, to zmniejszany jest licznik KEYDEL, który określa czas pomiędzy kolejnymi odczytami klawiatury. Naciśnięcie klawisza powoduje także zmniejszenie stanu licznika SRTIMER (wykorzystywany przez POKEY) i sprawdzanie kodu klawisza. Gdy nie jest to CTRL-1, HELP, CTRL-F1, CTRL-F2 ani CTRL-F4 (trzy ostatnie kody dotyczą modelu 1200XL), to zostaje przepisany do rejestru-cienia w pamięci RAM.

Ostatnią częścią procedury SYSVBL jest przepisanie informacji z portów joysticków obsługiwanych przez układy I/O do rejestrów RAM. Z układu PIA odczytywane jest położenie joysticków, z układu POKEY położenie potencjometrów, a z GTIA - stan przycisków w joystickach i potencjometrach. Stare modele 400/800 posiadały gniazda dla czterech joysticków lub ośmiu potencjometrów, a modele XL/XE mają ich o połowę mniej. Ponieważ niektóre starsze gry wymagają takiej ilości manipulatorów jak w 400/800, to wartości dotyczące joysticków 0 i 1 są

kopiowane do rejestrów joysticków 2 i 3. To samo dotyczy potencjometrów (wartości z 0-3 są kopiowane do 4-7).

Procedura SYSVBL kończy się skokiem pod adres wskazywany przez wektor VVBLKD (Vector VBLK Deferred). Normalnie wskazuje on procedurę EXITVBL, lecz może być zmieniony przez użytkownika (przy pomocy SETVBLV).

```
0100 ;EXIT Vertical BLank interrupt
0110 ;
0120     *=  $C28A
0130 ;
0140     PLA
0150     TAY
0160     PLA
0170     TAX
0180     PLA
0190     RTI
```

Zakończenie przerwania synchronizacji pionowej odbywa się poprzez procedurę EXITVBL. Jej jedynym zadaniem jest odtworzenie ze stosu umieszczonych tam na początku procedury obsługi przerwania rejestrów procesora. Po instrukcji RTI procesor odtwarza jeszcze rejestr statusu i licznik programu, co automatycznie powoduje kontynuowanie programu od miejsca, w którym wystąpiło żądanie przerwania.

3.1.4. Procedury uzupełniające VBLKI

Podczas przerwania synchronizacji pięciokrotnie wywoływana jest procedura DECTIM. Służy ona do zmniejszania liczników systemowych TIMCNT1-5. Przed jej wywołaniem w rejestrze X musi zostać umieszczony numer licznika zmniejszony o 1 i pomnożony przez 2 ((licznik-1)*2), czyli indeks licznika liczony od TIMCNT1.

```
0100 ;DECrement TIMer
0110 ;
0120 TIMCNT = $0218
0130 ;
0140     *=  $C255
0150 ;
0160     LDY TIMCNT,X
0170     BNE DCL
0180     LDY TIMCNT+1,X
0190     BEQ NOT0
0200     DEC TIMCNT+1,X
0210 DCL DEC TIMCNT,X
0220     BNE NOT0
0230     LDY TIMCNT+1,X
0240     BNE NOT0
0250     LDA #$00
0260     RTS
0270 NOT0 LDA #$FF
0280     RTS
```

Jeżeli zmniejszenie stanu licznika spowoduje jego wyzerowanie, to przed opuszczeniem procedury do akumulatora wpisywana jest wartość 0. W przeciwnym wypadku zawartością akumulatora jest #$FF.

Wyzerowanie licznika TIMCNT1 lub TIMCNT2 powoduje wywołanie procedury, której adres wskazuje wektor TIMVEC1 lub TIMVEC2. Normalnie procedura licznika 2 jest niewykorzystana, natomiast TIMEVEC1 wskazuje procedurę TIM1INT.

```
0100 ;TIMer 1 INTerrupt
0110 ;
0120 TIMFLG = $0317
0130 ;
0140     *=  $EC11
0150 ;
0160     LDA #$00
0170     STA TIMFLG
0180     RTS
```

Procedura ta zeruje wskaźnik Timeout (czasu oczekiwania na odpowiedź), co jest wykorzystywane podczas operacji wejścia/wyjścia.

3.1.5. Ingerencja w procedurę VBLKI

Jak wspomniano wcześniej zmiana procedury synchronizacji pionowej jest bardzo prosta. Własną procedurę użytkownik może umieścić w dowolnym miejscu procedury systemowej lub też zamiast niej. Dostęp systemu do procedury użytkownika uzyskuje się przez zmianę wektorów VVBLKI i/lub VVBLKD. Służy do tego procedura SETVBLV.

Wymaga ona przed wywołaniem umieszczenia w rejestrze X starszego bajtu wektora, w rejestrze Y młodszego bajtu wektora i w akumulatorze numeru zmienianego wektora. Numery wektorów są następujące:

```
0 - VIMIRQ
1 - TIMCNT1
2 - TIMCNT2
3 - TIMCNT3
4 - TIMCNT4
5 - TIMCNT5
6 - VVBLKI
7 - VVBLKD
8 - TIMVEC1
9 - TIMVEC2
```

Wymienione powyżej rejestry TIMCNT są licznikami zliczającymi wstecz (do zera) i zamiast wektora należy podać dla nich wartość początkową.

```
0100 ;SET Vertical BLank
0110 ;interrupt Vectors
0120 ;
0130 INTEMP = $022D
0140 VIMIRQ = $0216
```

```
0150 WSYNC = $D40A
0160 ;
0170     *=  $C272
0180 ;
0190     ASL A
0200     STA INTEMP
0210     TXA
0220     LDX #$05
0230     STA WSYNC
0240 LOOP DEX
0250     BNE LOOP
0260     LDX INTEMP
0270     STA VIMIRQ+1,X
0280     TYA
0290     STA VIMIRQ,X
0300     RTS
```

A oto opis działania SETVBLV. Najpierw numer wektora mnożony jest przez 2 (adresy są dwubajtowe) i odkładany tymczasowo do INTEMP, a do akumulatora przepisywana jest zawartość rejestru X. Po wpisaniu dowolnej wartości do rejestru WSYNC (Wait for SYNChronisation) następuje zatrzymanie procesora, aż do impulsu synchronizacji pionowej. Następnie, po odliczeniu czasu potrzebnego na wykonanie przez ANTIC operacji tworzenia linii obrazu, do rejestru określonego wartością pobraną z INTEMP (indeks od VIMIRQ) wpisywany jest starszy bajt z akumulatora i młodszy z rejestru Y. Procedura kończy się zwykłym rozkazem RTS.

3.1.6. Procedura przerwania DL

Mimo, iż system operacyjny nie korzysta z przerwań wywoływanych przez program ANTIC-a (Display List Interrupt), to w pamięci ROM znajduje się procedura tego przerwania. Służy ona do obsługi delikatnego przesuwu obrazu. Przesuw delikatny polega na pionowym przemieszczaniu obrazu o jedną linię ekranową. Dla przesunięcia o jeden wiersz (linię obrazu) w trybie GRAPHICS 0 trzeba wykonać osiem przesunięć o linię ekranu.

```
0100 ;Fine Scroll Display List
0110 ;
0120 ATRMSK = $4E
0130 COLPF1 = $D017
0140 COLPF2S = $02C6
0150 COLRSH = $4F
0160 WSYNC = $D40A
0170 ;
0180     *=  $FCC4
0190 ;
0200     PHA
0210     LDA COLPF2S
0220     EOR COLRSH
0230     AND ATRMSK
0240     STA WSYNC
0250     STA COLPF1
0260     PLA
0270     RTI
```

Procedura ta jest wywoływana przed wyświetleniem ostatniej linii obrazu i powoduje zmianę koloru tej linii tak, aby nie była widoczna jej zawartość. Ma to na celu ukrycie dodatkowej, ostatniej linii podczas przesuwania obrazu.

3.2. Procedury przerwań maskowalnych

Żądanie przerwania maskowalnego IRQ powoduje (po zapisaniu na stos licznika programu i rejestru statusu) skok pod adres wskazany przez wektor IRQVEC ($FFFE). Wektor ten zawiera adres procedury JMPIRQV ($C02C).

```
0100 ;JuMP IRQ Vector
0110 ;
0120 VIMIRQ = $0216
0130 ;
0140     *=  $C02C
0150 ;
0160     CLD
0170     JMP (VIMIRQ)
```

Ta króciutka procedura została dodana w modelach XL/XE, ponieważ wcześniejsze modele 400/800 robiły "dziwne" rzeczy podczas przerwań. Brakowało tam po prostu instrukcji CLD, która wyłącza dziesiętny tryb pracy procesora. Teraz po skasowaniu trybu dziesiętnego następuje skok do głównej procedury przerwania IRQ wskazywanej przez wektor VIMIRQ (Vector IMmediate IRQ).

3.2.1. Procedura rozpoznania źródła przerwania

Do rozpoznania źródła sygnału żądania przerwania wykorzystywany jest przede wszystkim rejestr IRQST (IRQ STatus), którego każdy bit odpowiada jednemu z możliwych źródeł przerwania. Normalnie wszystkie bity tego rejestru są ustawione. Sygnał żądania przerwania powoduje (na drodze sprzętowej) skasowanie bitu odpowiadającego źródłu tego sygnału. Przyporządkowanie bitów rejestru IRQST jest następujące:

```
0 - przerwanie zegara TIMER1
1 - przerwanie zegara TIMER2
2 - przerwanie zegara TIMER4
3 - przerwanie końca transmisji
4 - przerwanie zapisu szeregowego
5 - przerwanie odczytu szeregowego
6 - przerwanie klawiatury
7 - przerwanie klawisza BREAK
```

Analogiczne przyporządkowanie mają bity rejestru IRQEN (IRQ ENable). Skasowanie bitu w tym rejestrze powoduje zabronienie odpowiadającego mu przerwania IRQ.

Rozpoznanie źródła przerwania IRQ wykonuje procedura SINRDYI. Wiele z jej odgałęzień jest aktualnie nie wykorzystane. Są one przewidziane dla przyszłych rozszerzeń systemu lub zastosowania przez użytkownika.

W procedurze SINRDYI można zauważyć jedną z niewielu niekonsekwencji systemu. W jej środku jest wstawiona procedura BREAKIRQ, która nie ma żadnego związku z SINRDYI i musi być ominięta.

```
0100 ;Serial INput ReaDY Irq
0110 ;
0120 DLIV =  $0200
0130 IRQENS = $10
0140 IRQST = $D20E
0150 KEYDIS = $026D
0160 NEWIOP = $028C
0170 PACTL = $D302
0180 PBCTL = $D303
0190 PDVREG = $D1FF
0200 PINTMSK = $0249
0210 PORTA = $D300
0220 PORTB = $D301
0230 VBREAK = $0206
0240 VINTER = $0204
0250 VPIRQ = $0238
0260 VPRCED = $0202
0270 VSERIN = $020A
0280 ;
0290     *=  $C030
0300 ;
0310     PHA
0320     LDA IRQST
0330     AND #$20
0340     BNE PDV
0350     LDA #$DF
0360     STA IRQST
0370     LDA IRQENS
0380     STA IRQST
0390     JMP (VSERIN)
0400 PDV TXA
0410     PHA
0420     LDA PDVREG
0430     AND PINTMSK
0440     BEQ MSK
0450     JMP (VPIRQ)
0460 MSK LDX #$06
0470 LOOP LDA MASKTAB,X
0480     CPX #$05
0490     BNE BPS1
0500     AND IRQENS
0510     BEQ BPS2
0520 BPS1 BIT IRQST
0530     BEQ FIN
0540 BPS2 DEX
0550     BPL LOOP
0560     JMP PRC
0570 FIN EOR #$FF
0580     STA IRQST
0590     LDA IRQENS
0600     STA IRQST
0610     CPX #$00
0620     BNE NEW
0630     LDA KEYDIS
```

```
0640     BNE PRC
0650 NEW LDA VECTAB,X
0660     TAX
0670     LDA DLIV,X
0680     STA NEWIOP
0690     LDA DLIV+1,X
0700     STA NEWIOP+1
0710     PLA
0720     TAX
0730     JMP (NEWIOP)
0740 ;
0750     *=  $C0A0
0760 ;
0770 PRC PLA
0780     TAX
0790     BIT PACTL
0800     BPL INT
0810     LDA PORTA
0820     JMP (VPRCED)
0830 INT BIT PBCTL
0840     BPL BRK
0850     LDA PORTB
0860     JMP (VINTER)
0870 BRK PLA
0880     STA NEWIOP
0890     PLA
0900     PHA
0910     AND #$10
0920     BEQ EXIT
0930     LDA NEWIOP
0940     PHA
0950     JMP (VBREAK)
0960 EXIT LDA NEWIOP
0970     PHA
0980 PLARTI PLA
0990 RTI RTI
1000 ;
1010 ;MASK TABle
1020 ;
1030 MASKTAB .BYTE $80,$40,$04,$02
1040     .BYTE $01,$08,$10,$20
1050 ;
1060 ;VECtor TABle
1070 ;
1080 VECTAB .BYTE $36,$08,$14,$12
1090     .BYTE $10,$0E,$0C,$0A
```

W pierwszej kolejności sprawdzany jest bit 5 IRQST, który sygnalizuje zapełnienie rejestru wejściowego i konieczność odczytania z niego danych. Gdy bit 5 jest skasowany, sterowanie zostaje przekazane do procedury ISRSIR.

Następnie przez porównywanie poszczególnych bitów w rejestrach PDVREG i PINTMSK (Parallel INTerrupt MaSK) sprawdzane jest, czy źródłem żądania przerwania jest nowe urządzenie dołączone do szyny równoległej. Jeśli tak, to wykonywany jest skok do procedury obsługi tego przerwania wskazanej wektorem VPIRQ (Vector Parallel IRQ).

Teraz sprawdzane są pozostałe bity rejestru IRQST w kolejności określonej przez maski bitowe z tabeli MASKTAB. Kolejność ta wyznacza więc priorytet przerwań maskowalnych. Po znalezieniu skasowanego bitu pozostałe są ustawiane, aby zablokować przerwania o niższym priorytecie. Gdy przerwanie zostało wywołane naciśnięciem klawisza (oprócz BREAK), to sprawdzane jest jeszcze, czy rejestr KEYDIS (KEYboard DISable) nie wskazuje zablokowania klawiatury.

Następnie według znalezionego źródła przerwania odszukiwany jest w tabeli VECTAB indeks jego wektora. Wektor ten jest umieszczany w rejestrze NEWIOP i wykonywany jest skok pod zawarty tam adres. W ten sposób wywoływane są procedury ISRODN, ISRXD, CPUIRQ i BREAKIRQ. Pozostałe trzy wektory procedur obsługi przerwań wywołanych przez liczniki POKEY-a nie są normalnie wykorzystywane przez OS. Wskazują one na sekwencję rozkazów PLA, RTI i powodują powrót z przerwania.

Jeżeli do tej pory nie zostało znalezione źródło przerwania, to sprawdzane są jeszcze porty układu PIA. Stwierdzenie żądania przerwania przez port A powoduje skok pod adres wskazany wektorem VPRCED (Vector PRoCEeD), a przez port B - wektorem VINTER (Vector INTERrupt). Także te procedury nie są używane przez system i przerwania kończy się sekwencją PLA, RTI.

Na końcu kontrolowany jest bit B rejestru statusu procedora sprzed przerwania (trzeba go więc pobrać ze stosu). Jeśli bit B jest ustawiony, to znaczy, że przerwanie zostało wywołane rozkazem BRK i następuje skok pod adres umieszczony w rejestrze VBREAK (normalnie wskazuje on PLA, RTI - koniec przerwania).

Jeżeli nie zostało odnalezione żadne źródło przerwania, to system operacyjny zakłada, że wystąpił jakiś błąd i kończy procedurę przerwania.

3.2.2. Przerwanie odczytu z szyny szeregowej

Kontrolujący komunikację z urządzeniami zewnętrznymi poprzez szynę szeregową układ POKEY jest głównym źródłem przerwań maskowalnych. Jednocześnie z sygnałem żądania przerwania kasuje on odpowiedni bit rejestru IRQST, aby wskazać systemowi operacyjnemu konkretną przyczynę przerwania. Najwyższy priorytet ma przerwanie wywoływane przez POKEY, gdy w rejestrze SERIN (SERial INput) pojawi się bajt danych z urządzenia zewnętrznego. Kasowany jest wtedy bit 5 rejestru IRQST i procedura SINRDYI po rozpoznaniu źródła przerwania wywołuje procedurę ISRSIR

```
0100 ;Interrupt Service Routine
0110 ;at Serial Input Ready
0120 ;
0130 BUFEN = $34
0140 BUFR =  $32
0150 BUFRFL = $38
0160 CHKSUM = $31
```

```
0170 NOCKSM = $3C
0180 RECVND = $39
0190 SERIN = $D20D
0200 SKSTAT = $D20F
0210 SKSTRES = $D20A
0220 STATUS = $30
0230 ;
0240     *=  $EB2E
0250 ;
0260     LDA SKSTAT
0270     STA SKSTRES
0280     BMI NER1
0290     LDY #$8C
0300     STY STATUS
0310 NER1 AND #$20
0320     BNE NER2
0330     LDY #$8E
0340     STY STATUS
0350 NER2 LDA BUFRFL
0360     BEQ EMPTY
0370     LDA SERIN
0380     CMP CHKSUM
0390     BEQ END
0400     LDY #$8F
0410     STY STATUS
0420 END LDA #$FF
0430     STA RECVND
0440 EXIT PLA
0450     TAY
0460     PLA
0470     RTI
0480 EMPTY LDA SERIN
0490     LDY #$00
0500     STA (BUFR),Y
0510     CLC
0520     ADC CHKSUM
0530     ADC #$00
0540     STA CHKSUM
0550     INC BUFR
0560     BNE BPS
0570     INC BUFR+1
0580 BPS LDA BUFR
0590     CMP BUFEN
0600     LDA BUFR+1
0610     SBC BUFEN+1
0620     BCC EXIT
0630     LDA NOCKSM
0640     BEQ CKS
0650     LDA #$00
0660     STA NOCKSM
0670     BEQ END
0680 CKS LDA #$FF
0690     STA BUFRFL
0700     BNE EXIT
```

Procedura ISRSIR najpierw sprawdza rejestr statusu złącza szeregowego i klawiatury SKSTAT. Poszczególne bity tego rejestru sygnalizują prawidłowość operacji odczytu. Gdy ustawiony jest bit

7, to znaczy, że odczytano zbyt mało lub zbyt dużo bitów. Powoduje to wpisanie do rejestru STATUS kodu błędu $8C (tzw. Framing Error). Ustawienie bitu 5 oznacza przepełnienie bufora i powoduje sygnalizowanie błędu o kodzie $8E (SIO Overrun).

Następnie sprawdzany jest znacznik zapełnienia bufora wejściowego, w którym wartość różna od zera oznacza pełny bufor. W takim przypadku odebrany bajt danych traktowany jest jako suma kontrolna i porównywany z zawartością rejestru CHKSUM. Gdy są one różne, to sygnalizowany jest błąd sumy kontrolnej (kod $8F). Przed końcem procedury wskaźnik zakończenia transmisji RECVND (RECeiVe eND) jest jeszcze ustawiany na wartość $FF.

Jeśli bufor wejściowy nie został jeszcze zapełniony, to bajt z rejestru wejściowego SERIN zostaje umieszczony w buforze i adres bufora (BUFR - BUFfeR) zwiększa się o jeden. Teraz następuje porównanie tego adresu z adresem końca bufora (BUFEN - BUFfer ENd). Gdy są one różne, procedura się kończy. W przeciwnym razie sprawdzany jest znacznik NOCKSM (NO ChecK SuM). Jego zawartość równa zero oznacza, że po danych nastąpi suma kontrolna. Wskaźnik zapełnienia bufora ustawiany jest więc na wartość $FF i procedura się kończy. Jeżeli wskaźnik NOCKSM ma wartość różną od zera, to znaczy, że nie będzie sumy kontrolnej. W tym przypadku przed zakończeniem przerwania wskaźnik RECVND otrzymuje wartość $FF, a NOCKSM - wartość zero.

3.2.3. Przerwanie zapisu na szynę szeregową.

Stwierdzenie przez procedurę SINDRYI skasowania bitu 4 rejestru IRQST oznacza, że rejestr wyjściowy SEROUT został opróżniony i konieczne jest umieszczenie w nim następnego bajtu do zapisu na urządzeniu zewnętrznym. Wywoływana jest w tym celu procedura ISRODN.

```
0100 ;Interrupt Service Routine
0110 ;if Output Data Needed
0120 ;
0130 BUFEN = $34
0140 BUFR = $32
0150 CHKSNT = $3B
0160 CHKSUM = $31
0170 IRQEN = $D20E
0180 IRQEND = $10
0190 SEROUT = $D20D
0200 ;
0210      *=  EAAD
0220 ;
0230      TYA
0240      PHA
0250      INC BUFR
0251      TYA
0252      PHA

0260      BNE BPS
0270      INC BUFR+1
0280 BPS  LDA BUFR
0290      CMP BUFEN
0300      LDA BUFR+1
```

```
0310     SBC BUFEN+1
0320     BCC CONT
0330     LDA CHKSNT
0340     BNE SEND
0350     LDA CHKSUM
0360     STA SEROUT
0370     LDA #$FF
0380     STA CHKSNT
0390     BNE EXIT
0400 SEND LDA IRQENS
0410     ORA #$08
0420     STA IRQENS
0430     STA IRQEN
0440 EXIT PLA
0450     TAY
0460     PLA
0470     RTI
0480 CONT LDY #$00
0490     LDA (BUFR),Y
0500     STA SEROUT
0510     CLC
0520     ADC CHKSUM
0530     ADC #$00
0540     STA CHKSUM
0550     JMP EXIT
```

Na początku procedury adres bufora wyjściowego jest zwiększany o jeden i porównywany z adresem końca bufora. Jeżeli są one różne, to bajt danych z bufora jest przesyłany do rejestru SEROUT (SERial OUTput) oraz dodawany do zawartości rejestru CHKSUM.

Gdy cały bufor został opróżniony, to sprawdzany jest wskaźnik CHKSNT (CHecK sum SeNT). Jeśli jest równy zero, to do rejestru SEROUT przepisywana jest suma kontrolna z rejestru CHKSUM i znacznik CHKSNT otrzymuje wartość $FF.

Jeżeli CHKSNT wa wartość $FF, to znaczy, że suma kontrolna nie będzie wysyłana i bit 3 rejestru IRQEN jest ustawiany, co zezwala na wywołanie przerwania przez koniec transmisji.

3.2.4. Przerwanie końca transmisji szeregowej.

```
0100 ;Interrupt Service Routine
0110 ;if eXmitend Data
0120 ;
0130 CHKSNT = $3B
0140 IRQEN = $D20E
0150 IRQENS = $10
0160 XMTDON = $3A
0170 ;
0180     *=  $EAEC
0190 ;
0200     LDA CHKSNT
0210     BEQ EXIT
0220     STA XMTDON
0230     LDA IRQENS
0240     AND #$F7
```

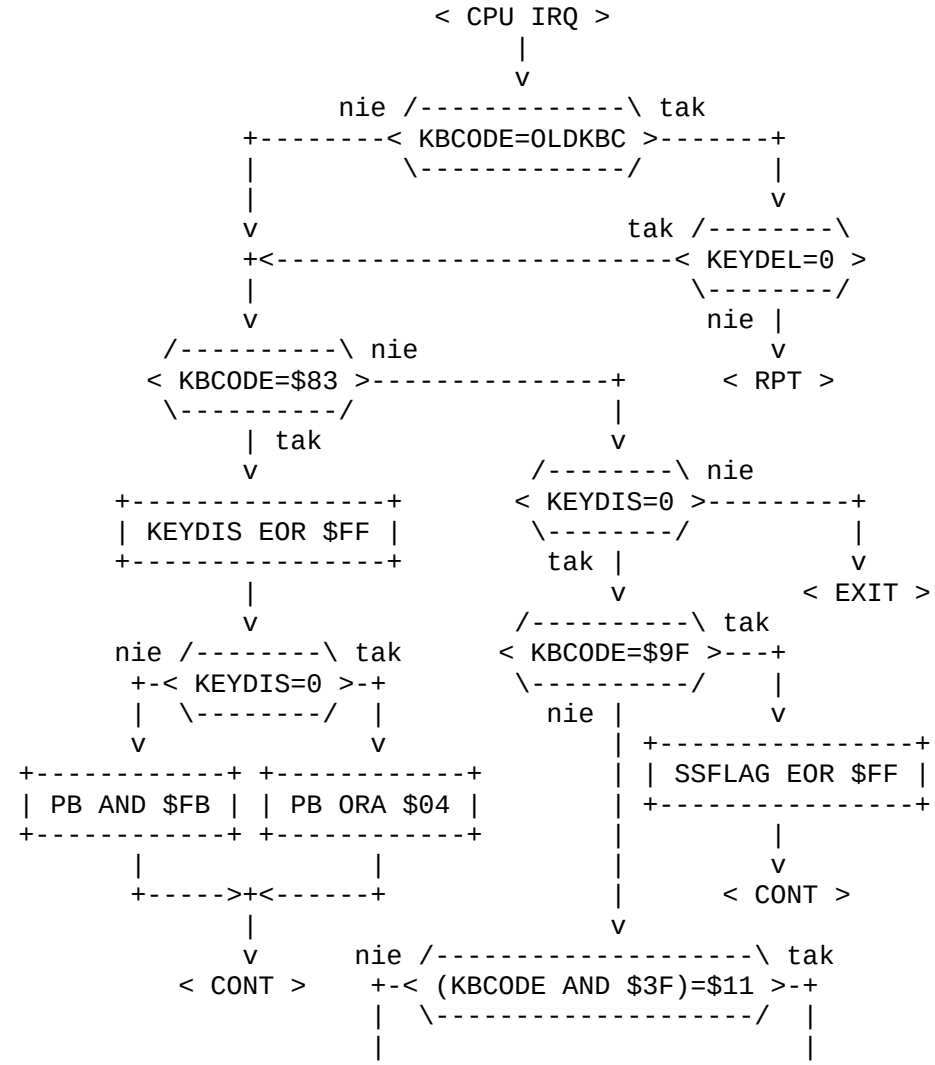
```
0250     STA IRQENS
0260     STA IRQEN
0270 EXIT PLA
0280     RTI
```

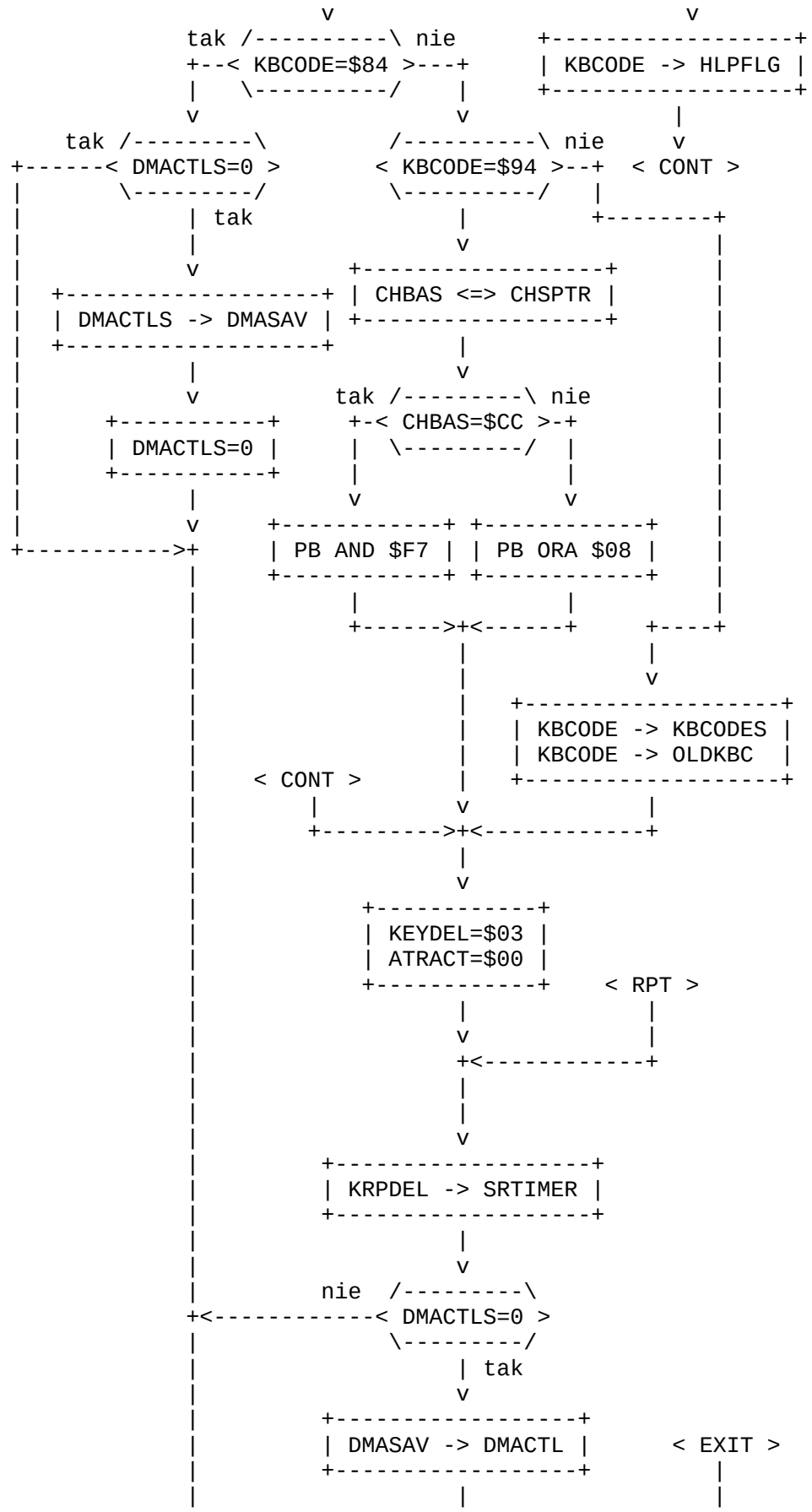
Skasowanie bitu 3 w rejestrze IRQST sygnalizuje zakończenie transmisji szeregowej i powoduje wywołanie z SINDRYI procedury ISRXD.

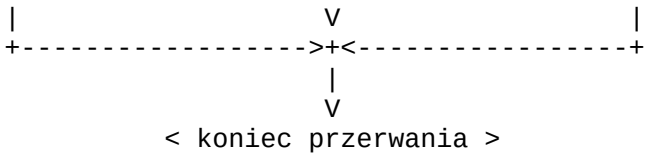
Najpierw badany jest wskaźnik CHKSNT. Jeśli jest równy 0, to suma kontrolna nie była wysłana i procedura się kończy. W przeciwnym razie CHKSNT jest przepisywany do wskaźnika XMTDON (eXMiTe DONe) sygnalizującego zakończenie transmisji i w rejestrze IRQEN kasowany jest bit zezwolenia na przerwanie końca transmisji (bit 3).

3.2.5. Przerwanie klawiatury

Naciśnięcie dowolnego klawisza (oprócz BREAK) powoduje skasowanie bitu 6 rejestru IRQST. Procedura SINDRYI po rozpoznaniu źródła przerwania wywołuje wtedy procedurę CPUIRQ, której zadaniem jest odczyt naciśniętego klawisza.




tak
KBCODE=$84
nie
KBCODE -> HLPFLG
tak
DMACTLS=0
KBCODE=$94
nie
CONT
tak
CHBAS <=> CHSPTR
DMACTLS -> DMASAV
DMACTLS=0
tak
CHBAS=$CC
nie
PB AND $F7
PB ORA $08
KBCODE -> KBCODES
KBCODE -> OLDKBC
CONT
KEYDEL=$03
ATRACT=$00
RPT
KRPDEL -> SRTIMER
nie
DMACTLS=0
tak
DMASAV -> DMACTL
EXIT



Należy zwrócić uwagę, że znaczna część wykonywanych w tej procedurze czynności dotyczy wyłącznie modelu 1200XL posiadającego klawisze funkcyjne od F1 do F4. Poprzez przedefiniowanie klawiszy można jednak zaimplementować te funkcje w pozostałych modelach serii XL i XE.

Procedura CPUIRQ rozpoczyna się od porównania kodu naciśniętego klawisza pobranego z rejestru KBCODE (KeyBoard CODE) z kodem zapisanym w rejestrze OLDKBC (OLD KeyBoard Code) podczas poprzedniego przerwania. Jeżeli są takie same, to znaczy, że naciśnięty jest ten sam klawisz. Wtedy sprawdzany jest rejestr KEYDEL (KEYboard DELay) określający czas między dwoma naciśnięciami klawiszy. Gdy jest on różny od zera, to następuje skok do końcowej fazy procedury.

Jeśli został naciśnięty inny klawisz niż poprzednio lub upłynął czas między dwoma odczytami, to kolejno sprawdzane są specjalne kombinacje klawiszy. Odczytany kod klawisza równy $83 oznacza naciśnięcie CONTROL-F1. Kombinacja ta służy do włączania i wyłączania klawiatury (w 1200XL), więc na zawartości rejestru KEYDIS dokonywana jest operacja EOR $FF, co powoduje przełączenie klawiatury (KEYDIS=0 - klawiatura włączona, KEYDIS=$FF - wyłączona). Następnie jest ustawiany lub kasowany bit 3 rejestru PORTB, który w modelu 1200XL steruje diodą LED 1 sygnalizującą zablokowanie klawiatury. Po zablokowaniu klawiatury procedura się kończy.

Jako druga sprawdzana jest kombinacja klawiszy CONTROL-1, która zatrzymuje i wznawia przesuw pionowy obrazu (scrolling). Rozpoznanie tej kombinacji (kod $9F) powoduje przełączenie - poprzez EOR $FF - zawartości rejestru SSFLAG (Start/Stop FLAG). Normalnie w rejestrze tym znajduje się wartość 0. Naciśnięcie CONTROL-1 zmienia ją na $FF i powoduje zatrzymanie wyprowadzania danych na ekran. Kolejne naciśnięcie CONTROL-1 przywraca poprzednią sytuację.

```
0100 ;CPU Interrupt ReQuest
0110 ;
0120 ATRACT = $4D
0130 CHBAS = $02F4
0140 CHSPTR = $026B
0150 DMACTLS = $022F
0160 DMASAV = $02DD
0170 HLPFLG = $02DC
0180 KBCODE = $D209
0190 KBCODES = $02FC
0200 KEYDEL = $02F1
0210 KEYDIS = $026D
0220 KRPDEL = $02D9
```

```
0230 OLDKBC = $02F2
0240 PORTB = $D301
0250 SRTIMER = $022B
0260 SSFLAG = $02FF
0270 ;
0280     *=  $FC19
0290 ;
0300     TXA
0310     PHA
0320     TYA
0330     PHA
0340     LDY PORTB
0350     LDA KBCODE
0360     CMP OLDKBC
0370     BNE NKY
0380     LDX KEYDEL
0390     BNE RPT
0400 NKY LDX KEYDIS
0410     CMP #$83
0420     BNE NKB
0430     TXA
0440     EOR #$FF
0450     STA KEYDIS
0460     BNE KBEN
0470     TYA
0480     ORA #$04
0490     BNE KEND
0500 KBEN TYA
0510     AND #$FB
0520 KEND TAY
0530     BCS CONT
0540 NKB TXA
0550     BNE EXIT
0560     LDA KBCODE
0570     TAX
0580     CMP #$9F
0590     BNE NC1
0600     LDA SSFLAG
0610     EOR #$FF
0620     STA SSFLAG
0630     BCS CONT
0640 NC1 AND #$3F
0650     CMP #$11
0660     BNE FKY
0670     STX HLPFLG
0680     BEQ CONT
0690 STORE STX KBCODES
0700     STX OLDKBC
0710 CONT LDA #$03
0720     STA KEYDEL
0730     LDA #$00
0740     STA ATRACT
0750 RPT LDA KRPDEL
0760     STA SRTIMER
0770     LDA DMACTLS
0780     BNE EXIT
0790     LDA DMASAV
0800     STA DMACTLS
0810 EXIT STY PORTB
```

```
0820     PLA
0830     TAY
0840     PLA
0850     TAX
0860     PLA
0870     RTI
0880 FKY CPX #$84
0890     BEQ DISP
0900     CPX #$94
0910     BNE STORE
0920     LDA CHBAS
0930     LDX CHSPTR
0940     STA CHSPTR
0950     STX CHBAS
0960     CPX #$CC
0970     BEQ SET2
0980     TYA
0990     ORA $08
1000     TAY
1010     BNE CONT
1020 SET2 TYA
1030     AND #$F7
1040     TAY
1050     JMP CONT
1060 DISP LDA DMACTLS
1070     BEQ EXIT
1080     STA DMASAV
1090     LDA #$00
1100     STA DMACTLS
1110     BEQ EXIT
```

Następnie badane jest, czy sześć młodszych bitów rejestru KBCODE ma wartość $11, co oznacza naciśnięcie klawisza HELP (w dowolnej kombinacji z SHIFT i/lub CONTROL). Jeśli tak, to zawartość KBCODE jest przepisywana do HLPFLG (HeLP FLaG).

Jako ostatnie sprawdzane są dwie kombinacje klawiszy charakterystyczne dla 1200XL. Kod $84 oznacza kombinację CONTROL-F2, a kod $94 kombinację CONTROL-F4. Pierwsza z nich włącza i wyłącza dostęp ANTIC-a do pamięci, a więc włącza i wyłącza obraz. Na czas wyłączenia obrazu zawartość rejestru DMACTLS (DMA ConTroL Shadow register) jest przechowywana w DMASAV (DMA SAVe).

Kombinacja CONTROL-F4 przełącza wbudowane zestawy znaków: standardowy i międzynarodowy. W tym przypadku zamieniane są miejscami zawartości wektorów CHBAS (CHaracter set BASe) i CHSPTR (CHaracter Set PoinTeR) oraz ustawiany lub kasowany jest bit 4 PORTB, który w 1200XL steruje diodą LED 2 (świeci się przy korzystaniu z zestawu międzynarodowego).

Gdy nie została rozpoznana żadna z wymienionych kombinacji klawiszy, to kod naciśniętego klawisza jest zapisywany do rejestrów KBCODES i OLDKBC, rejestr KEYDEL otrzymuje wartość

3, a ATRACT jest zerowany. Następnie zawartość KRPDEL (Key RePeat DELay) jest przepisywana do licznika SRTIMER i procedura się kończy.

3.2.6. Przerwanie klawisza BREAK

Klawisz BREAK posiada szczególne znaczenie i jego naciśnięcie jest oddzielnie sygnalizowane w bicie 7 rejestru IRQST. Przerwanie wywołane naciśnięciem tego klawisza jest obsługiwane przez procedurę BREAKIRQ.

```
0100 ;BREAK key IRQ
0110 ;
0120 ATRACT = $4D
0130 CRSINH = $02F0
0140 IRQSTAT = $11
0150 SSFLAG = $02FF
0160 ;
0170     *=  $C092
0180 ;
0190     LDA #$00
0200     STA IRQSTAT
0210     STA SSFLAG
0220     STA CRSINH
0230     STA ATRACT
0240     PLA
0250     RTI
```

Jedynym zadaniem tej procedury jest wyzerowanie rejestrów IRQSTAT, SSFLAG, ATRACT i CRSINH. Znaczenie trzech pierwszych było już omawiane. Rejestr CRSINH (CuRSor INHibition) określa widoczność kursora. Jeżeli zawiera on wartość różną od zera, to kursor jest niewidoczny na ekranie.

3.2.7. Przerwanie nowego urządzenia

Jako ostatnie zostanie omówione przerwanie, które ma drugi priorytet - ustępuje jedynie przerwaniu wywołanemu przez odczyt z szyny szeregowej. Jest to przerwanie wywoływane przez nowe urządzenie przyłączone do szyny równoległej. Znaczenie określenia "nowe urządzenie" zostało wyjaśnione wcześniej.

Żądanie przerwania jest sygnalizowane przez nowe urządzenie w rejestrze PINTMSK i procedura SINDRYI po sprawdzeniu istnienia urządzenia (w rejestrze PDVREG) wywołuje procedurę NEWIOREQ.

```
0100 ;NEW I/O REQest
0110 ;
0120 BITMASK = $CA21
0130 DEVINT = $D808
0140 PDVREG = $D1FF
0150 PDVRS = $0248
0160 ;
0170     *=  $C96E
0180 ;
```

```
0190     LDX #$08
0200 NEXT ROR A
0210     BCS FIND
0220     DEX
0230     BNE NEXT
0240 FIND LDA PDVRS
0250     PHA
0260     LDA BITMASK-1,X
0270     STA PDVRS
0280     STA PDVREG
0290     JSR DEVINT
0300     PLA
0310     STA PDVRS
0320     STA PDVREG
0330     PLA
0340     TAX
0350     PLA
0360     RTI
```

Przed wywołaniem tej procedury w akumulatorze ustawione są bity odpowiadające urządzeniom żądającym przerwania. Po znalezieniu urządzenia, które wysłało sygnał żądania przerwania, aktualna zawartość rejestru PDVRS jest odkładana na stos, a w PDVRS jest umieszczana odpowiednia maska bitowa pobrana z tabeli BITMASK.

```
1000 ;BIT MASK
1010 ;
1020     *=  $CA21
1030 ;
1040     .BYTE $80,$40,$20,$10
1050     .BYTE $08,$04,$02,$01
```

Następnie wywoływana jest znajdująca się w pamięci ROM nowego urządzenia procedura obsługi przerwania DEVINT (DEVice INTerrupt). Po jej zakończeniu odtwarzana jest ze stosu zawartość rejestrów PDVRS i PDVREG i na tym przerwanie jest zakończone.

Rozdział 4

PROCEDURY ZMIENNOPRZECINKOWE

W systemie operacyjnym komputerów Atari znajduje się wydzielony blok 2 KB pamięci ROM, w którym znajdują się procedury operacji matematycznych na liczbach rzeczywistych, zwanych także liczbami zmiennoprzecinkowymi (floating point). Są one niezbędne do wykonywania obliczeń na liczbach spoza zakresu obejmowanego przez dwubajtowe liczby całkowite, które stanowią podstawową reprezentację liczb w komputerach.

Trzeba od razu zaznaczyć, że jest to najgorzej opracowana część systemu. Programy i języki wyższego poziomu korzystające z tych procedur są bardzo wolne, czego najlepszym przykładem jest wbudowany Atari Basic. Wiele z tych procedur można jednak wykorzystać w całości lub w części we własnych programach.

4.1. Format liczb zmiennoprzecinkowych

Liczby rzeczywiste są zapisywane w Atari w specjalny sposób, pozwalający na umieszczenie w stosunkowo niewielkiej przestrzeni dużego zakresu wartości liczbowych. Arytmetyka zmiennorzecinkowa Atari może dzięki temu operować na liczbach z zakresu od 10^{-98} do 10^{+98}. Dla przykładu zakres liczb dostępnych w Commodore 64 jest znacznie mniejszy: 10^{-38}-10^{+38}.

Liczby zmiennoprzecinkowe (Floating Point - FP) zajmują zawsze sześć bajtów. W pierwszym bajcie zapisany jest znak liczby oraz jej wykładnik (eksponent). Znak liczby jest określony przez najstarszy, siódmy bit. Gdy jest on ustawiony, to liczba jest ujemna. Pozostałe siedem bitów jest właściwym wykładnikiem potęgi liczby 100 zwiększonym o $40 (64).

Reszta, czyli pięć bajtów, zawiera mantysę liczby zapisaną w kodzie BCD. Kod ten pozwala na umieszczenie w jednym bajcie dwóch cyfr dziesiętnych, więc cała mantysa ma dziesięć cyfr znaczących. Mantysa jest zapisywana zawsze w ten sposób, aby jej pierwszy bajt był różny od zera. System zakłada, że za pierwszym bajtem znajduje się punkt dziesiętny (w zapisie anglosaskim stosowanym w komputerach jest to kropka, w polskim - przecinek).

Zero jest w tej konwencji reprezentowane przez zerowy wykładnik i zerową mantysę (sześć bajtów równych zero). Przykłady konwersji liczb na zapis stosowany w Atari można znaleźć m. in. w książkach "Asembler 6502", "De Re Atari" i "PEEK-POKE 2".

4.2. Procedury operacji na liczbach FP

Pakiet procedur FP znajduje się w pamięci ROM w obszarze $D800-$DFFF. Należy przy tym zwrócić uwagę, że ten sam obszar jest wykorzystywany przez nowe urządzenia. Procedury i

programy obsługi nowych urządzeń nie mogą więc korzystać z procedur FP.

Dla potrzeb operacji FP wykorzystywane są także obszary pamięci RAM od $D4 do $FF oraz od $057E do $05FF. W pierwszym z wymienionych obszarów znajdują się podstawowe rejestry liczb FP oraz rejestry pomocnicze operacji FP. Drugi obszar zajmuje bufor danych dla operacji FP.

Liczby zmiennoprzecinkowe przechowywane są w czterech specjalnych sześciobajtowych rejestrach: FR0 (Floating point Register 0 - $D4-$D9), FR1 ($E0-$E5), FR2 ($E6-$EB) oraz FRE (FP Register Extra - $DA-$DF). Do wyprowadzania liczb FP w innych formatach służy bufor LBUFF (Line BUFfer), zaś bufor wejściowy jest wskazywany wektorem INBUFP (INput BUFfer Pointer). Liczby w formacie FP umieszczone poza rejestrami FR są wskazywane wektorami FLPTR (FLoating PoinTeR) i FPTR2 (Floating PoinTeR 2). Poza tym podczas obliczeń wykorzystywane są rejestry EEXP (E EXPonent), NSIGN (Number SIGN - znak liczby) i ESIGN (Exponent SIGN - znak wykładnika) oraz rejestry przejściowe ZTEMP1-3 (Zeropage TEMPorary register).

Spośród procedur składających się na pakiet arytmetyki zmiennoprzecinkowej można wyodrębnić trzy najważniejsze grupy: procedury przekształceń z i na format FP, procedury obliczeń na liczbach FP oraz procedury przemieszczeń liczb FP.

4.2.1. Procedury przekształceń liczb FP

Ponieważ komputer normalnie nie operuje na liczbach zmiennoprzecinkowych, muszą one być wprowadzane w innej postaci i zamieniane na format FP. Odwrotną operację należy wykonać w celu wyprowadzenia otrzymanych wyników. W komputerze liczby mogą być przedstawiane w postaci dwubajtowych liczb całkowitych albo w postaci ciągu znaków kodu ASCII. Pakiet procedur zmiennoprzecinkowych zawiera dwie pary procedur do przekształceń liczb z tych formatów na sześciobajtowy format FP i odwrotnie.

Pierwsza para procedur umożliwia konwersję z i na dwubajtowe liczby całkowite. Zarówno liczba wejściowa, jak i wyjściowa umieszczane są w rejestrze FR0, przy czym liczba całkowita zajmuje w nim tylko dwa pierwsze bajty.

Procedura IFP wykonuje przekształcenie liczby całkowitej umieszczonej w dwóch pierwszych bajtach FR0 na liczbę FP, którą również umieszcza w FR0. W tym celu najpierw przenosi liczbę całkowitą do rejestru ZTEMP2 w odwróconej kolejności bajtów (starszy, młodszy) i zeruje rejestr FR0 przez wywołanie procedury ZFR0.

Następnie w trybie dziesiętnym procesora wykonuje szesnaście razy mnożenie liczby całkowitej przez 2 i dodawanie z użyciem bitu przeniesienia (Carry) trzech pierwszych bajtów mantysy liczby zmiennoprzecinkowej. W ten sposób dokonana zostaje konwersja cyfr. Ponieważ maksymalną

wartością liczby całkowitej jest 65535, to zakłada się wykładnik potęgi sto równy 2 (po zwiększeniu $42).

```
0100 ;Integer to FP conversion
0110 ;
0120 FR0 =   $D4
0130 NFR0 =  $DC00
0140 ZFR0 =  $DA44
0150 ZTEMP2 = $F7
0160 ;
0170     *=  $D9AA
0180 ;
0190     LDA FR0
0200     STA ZTEMP2+1
0210     LDA FR0+1
0220     STA ZTEMP2
0230     JSR ZFR0
0240     SED
0250     LDY #$10
0260 NX1 ASL ZTEMP2+1
0270     ROL ZTEMP2
0280     LDX #$03
0290 NX2 LDA FR0,X
0300     ADC FR0,X
0310     STA FR0,X
0320     DEX
0330     BNE NX2
0340     DEY
0350     BNE NX1
0360     CLD
0370     LDA #$42
0380     STA FR0
0390     JMP NFR0
```

Otrzymujemy więc liczbę postaci NN.NNNN*1002, gdzie N jest czterobitową cyfrą w kodzie BCD. Jeżeli liczba całkowita jest mniejsza niż 10000, to dwie pierwsze cyfry liczby FP są równe 0. Sytuacja taka jest nieprawidłowa, więc na końcu wywoływana jest procedura NFR0, która doprowadza liczbę zawartą w rejestrze FR0 do wymaganej postaci.

```
0100 ;Floating Point to Integer
0110 ;
0120 FR0 =   $D2
0130 IDEX = $DCB9
0140 ROLZ2 = $DA5A
0150 ZTEMP1 = $F5
0160 ZTEMP2 = $F7
0170 ZTEMP3 = $F9
0180 ;
0190     *=  $D9D2
0200 ;
0210     LDA #$00
0220     STA ZTEMP2
0230     STA ZTEMP2+1
0240     LDA FR0
0250     BMI EXIT
0260     CMP #$43
0270     BCS EXIT
```

```
0280     SEC
0290     SBC #$40
0300     BCC MIN
0310     ADC #$00
0320     ASL A
0330     STA ZTEMP1
0340 NXT JSR ROLZ2
0350     BCS EXIT
0360     LDA ZTEMP2
0370     STA ZTEMP3
0380     LDA ZTEMP2+1
0390     STA ZTEMP3+1
0400     JSR ROLZ2
0410     BCS EXIT
0420     JSR ROLZ2
0430     BCS EXIT
0440     CLC
0450     LDA ZTEMP2+1
0460     ADC ZTEMP3+1
0470     STA ZTEMP2+1
0480     LDA ZTEMP2
0490     ADC ZTEMP3
0500     STA ZTEMP2
0510     BCS EXIT
0520     JSR IDEX
0530     CLC
0540     ADC ZTEMP2+1
0550     STA ZTEMP2+1
0560     LDA ZTEMP2
0570     ADC #$00
0580     BCS EXIT
0590     STA ZTEMP2
0600     DEC ZTEMP1
0610     BNE NXT
0620 MIN JSR IDEX
0630     CMP #$05
0640     BCC BPS
0650     CLC
0660     LDA ZTEMP2+1
0670     ADC #$01
0680     STA ZTEMP2+1
0690     LDA ZTEMP2
0700     ADC #$00
0710     STA ZTEMP2
0720 BPS LDA ZTEMP2+1
0730     STA FR0
0740     LDA ZTEMP2
0750     STA FR0+1
0760     CLC
0770     RTS
0780 EXIT SEC
0790     RTS
```

Dokonująca odwrotnej zamiany procedura FPI rozpoczyna się od wyzerowania rejestru ZTEMP2 i sprawdzenia znaku liczby FP. Jeżeli liczba jest ujemna i nie może być przedstawiona jako całkowita, to ustawiany jest bit Carry i procedura się kończy. To samo następuje w przypadku wykładnika większego od 2, co oznacza liczbę większą od 65535 (a dokładniej: większą od

1000000).

Następnie jest obliczany wykładnik przez odjęcie $40 od pierwszego bajtu liczby FP. Wynik ujemny (wykładnik mniejszy od zera, czyli liczba mniejsza od 100) powoduje przeskoczenie do końcowej fazy procedury.

Teraz przy pomocy procedur IDEX i ROLZ2 są kolejno obliczane bajty liczby całkowitej. W przypadku wystąpienia błędu w dowolnej fazie tego przeliczania następuje opuszczenie procedury FPI z ustawionym bitem Carry. Jeżeli przebieg konwersji był poprawny, to liczba całkowita jest przenoszona z rejestru ZTEMP2 do dwóch pierwszych bajtów FR0 i po skasowaniu bitu Carry procedura się kończy.

```
0100 ;ROtate Left ZTEMP2
0110 ;
0120 ZTEMP2 = $F7
0130 ;
0140     *=  $DA5A
0150 ;
0160     CLC
0170     ROL ZTEMP2+1
0180     ROL ZTEMP2
0190     RTS
```

Procedura ROLZ2 wykonuje jedynie przesunięcie w lewo obu bajtów liczby całkowitej przechowywanej w rejestrze ZTEMP2.

```
0100 ;Integer Digit EXtract
0110 ;
0120 FRX =   $EC
0130 ROLFR0 = $DBEB
0140 ;
0150     *=  $DCB9
0160 ;
0170     JSR ROLFR0
0180     LDA FRX
0190     AND #$0F
0200     RTS
```

Pomocnicza procedura IDEX wywołuje najpierw procedurę ROLFR0, a następnie wydziela cyfrę (bity 0-3) z uzyskanego bajtu umieszczonego w dodatkowym rejestrze FRX.

```
0100 ;ROtate Left FP Registers
0110 ;
0120 FR0 =   $D4
0130 FR2 =   $E6
0140 FRX =   $EC
0150 ;
0160     *=  DBE7
0170 ;
0180 ROLFR2 LDX #FR2+1
0190     BNE CONT
0200 ROLFR0 LDA #FR0+1
0210 CONT LDY #$04
```

```
0220 NXT CLC
0230     ROL $04,X
0240     ROL $03,X
0250     ROL $02,X
0260     ROL $01,X
0270     ROL $00,X
0280     ROL FRX
0290     DEY
0300     BNE NXT
0310     RTS
```

Procedury ROLFR0 i ROLFR2 dokonują czterokrotnego przesunięcia w lewo mantysy liczby FP zawartej odpowiednio w rejestrze FR0 lub FR2. Przesunięciu podlega także zawartość rejestru FRX, dzięki czemu po zakończeniu procedury aktualnie najbardziej znacząca cyfra mantysy znajduje się w tym rejestrze (w bitach 0-3).

Do zamiany ciągu znaków ASCII na sześciobajtową liczbę FP służy procedura AFP. Przed jej rozpoczęciem adres ciągu znaków do zamiany musi być umieszczony w rejestrze INBUFP (INput BUFfer Pointer), a po dokonaniu konwersji otrzymana liczba zmiennoprzecinkowa znajduje się w rejestrze FR0.

```
0100 ;ASCII ro FP conversion
0110 ;
0120 AF1 =   $DA48
0130 ASCSS =  $DBBB
0140 CIX =   $F2
0150 DIGRT = $F1
0160 EEXP =  $ED
0170 ESIGN = $EF
0180 FCHRFLG = $F0
0190 FR0   = $D4
0200 FRX   = $EC
0210 INBCN = $DB94
0220 INBSS = $DBA1
0230 INCIX = $DB9D
0240 NFR0 =  $DC00
0250 NSIGN = $EE
0260 ROLFR0 = $DBEB
0270 ZFR0 =  $DA44
0280 ;
0290     *=  $D800
0300 ;
0310     JSR INBSS
0320     JSR ASCSS
0330     BCS EX1
0340     LDX #EEXP
0350     LDY #$04
0360     JSR AF1
0370     LDX #$FF
0380     STX DIGRT

0680     BNE PRV
0690     CMP #'+
0700     BEQ NXT
0710     CMP #'-
0390     JSR ZFR0
0400     BEQ BPS
0410 NXT LDA #$FF
0420     STA FCHRFLG
0430 BPS JSR INBCN
0440     BCS NDT
0450     PHA
0460     LDX FR0+1
0470     BNE OVF
0480     JSR ROLFR0
0490     PLA
0500     ORA FR0+5
0510     STA FR0+5
0520     LDX DIGRT
0530     BMI NXT
0540     INX
0550     STX DIGRT
0560     BNE NXT
0570 OVF PLA
0580     LDX DIGRT
0590     BPL NRM
0600     INC EEXP
0610 NRM JMP NXT
0620 EX1  RTS
0630 NDT CMP #'.
0640     BEQ DPT
0650     CMP #'E
0660     BEQ ESG
0670     LDX FCHRFLG

1100     STA EEXP
1110 ES0 PLA
1120     CLC
1130     ADC EEXP
```

```
0720     BEQ NEG
0730 NEG STA NSIGN
0740     BEQ NXT
0750 DPT LDX DIGRT
0760     BPL PRV
0770     INX
0780     STX DIGRT
0790     BEQ NXT
0800 ESG LDA CIX
0810     STA FRX
0820     JSR INBCN
0830     BCS NDG
0840 NX  TAX
0850     LDA EEXP
0860     PHA
0870     STX EEXP
0880     JSR INBCN
0890     BCS EVE
0900     PHA
0910     LDA EEXP
0920     ASL A
0930     STA EEXP
0940     ASL A
0950     ASL A
0960     ADC EEXP
0970     STA EEXP
0980     PLA
0990     CLC
1000     ADC EEXP
1010     STA EEXP
1020     LDY CIX
1030     JSR INCIX
1040 EVE LDA ESIGN
1050     BEQ ES0
1060     LDA EEXP
1070     EOR #$FF
1080     CLC
1090     ADC #$01
1140     STA EEXP
1150     BNE PRV
1160 NDG CMP #'+
1170     BEQ PLS
1180     CMP #'-
1190     BNE CNT
1200     STA ESIGN
1210 PLS JSR INBCN
1220     BCC NX
1230 CNT LDA FRX
1240     STA CIX
1250 PRV DEC CIX
1260     LDA EEXP
1270     LDX DIGRT
1280     BMI NSC
1290     BEQ NSC
1300     SEC
1310     SBC DIGRT
1320 NSC PHA
1330     ROL A
1340     PLA
1350     ROR A
1360     STA EEXP
1370     BCC POS
1380     JSR ROLFR0
1390 POS LDA EEXP
1400     CLC
1410     ADC #$44
1420     STA FR0
1430     JSR NFR0
1440     BCS EX2
1450     LDX NSIGN
1460     BEQ EXC
1470     LDA FR0
1480     ORA #$80
1490     STA FR0
1500 EXC CLC
1510 EX2 RTS
```

Na początku procedury AFP wyszukiwany jest (przez procedurę INBSS) pierwszy znak ciągu ASCII do konwersji. Następnie procedura ASCSS sprawdza poprawność tego znaku. Jeżeli znak jest nieprawidłowy, to procedura jest opuszczana z ustawionym bitem Carry.

Gdy ciąg może być poddany przekształceniu, to zerowane są rejestry EEXP, NSIGN, ESIGN, FCHRFLG (First CHaRacter FLaG) oraz FR0, a rejestr DIGRT (DIGits to Right of decimal) otrzymuje wartość $FF.

Teraz cyfry z ciągu ASCII są kolejno odczytywane i po zamianie na postać BCD, umieszczane w rejestrze FR0. Jeżeli odczytany znak nie jest cyfrą, to sprawdzane są inne dozwolone możliwości: punkt dziesiętny (.), znak wykładnika (E), znak plus (+) i znak minus (-).

Odczytanie punktu dziesiętnego powoduje ustalenie wartości wykładnika liczby FP. Po

rozpoznaniu "E" kolejno odczytywane są cyfry wykładnika. W ten sposób odczytywanie kolejnych znaków ciągu ASCII jest kontynuowane, aż do rozpoznania niedozwolonego znaku, co kończy procedurę AFP. Jeżeli w chwili zakończenia procedury liczba FP nie ma żadnej wartości (nie został prawidłowo odczytany żaden znak ciągu ASCII), to wskazywany jest błąd przez ustawienie bitu Carry.

Podczas przebiegu procedury AFP wywoływane jest kilka procedur pomocniczych, które mogą być zastosowane także w programach użytkownika. Niektóre z nich są także wywoływane podczas obliczeń na liczbach FP.

```
0100 ;INput Buffer Search for Space
0110 ;
0120 CIX =   $F2
0130 INBUFP = $F3
0140 ;
0150     *=  $DBA1
0160 ;
0170     LDY CIX
0180     LDA #$20
0190 NXT CMP (INBUFP),Y
0200     BNE EXIT
0210     INY
0220     BNE NXT
0230 EXIT STY CIX
0240     RTS
```

Zadaniem procedury INBSS jest wstępne rozpoznanie kolejnego znaku ciągu ASCII. Jeżeli jest to znak spacji, odczytywany jest następny znak. Gdy znak jest różny od spacji, to jego położenie w buforze jest zapisywane w rejestrze CIX (Current IndeX) i procedura się kończy.

```
0100 ;ASCII String Search
0110 ;
0120 CIX =   $F2
0130 INBCN = $DB94
0140 ;
0150     *=  $DBBB
0160 ;
0170     LDA CIX
0180     PHA
0190     JSR INBCN
0200     BCC EXC
0210     CMP #'.
0220     BEQ DPT
0230     CMP #'+
0240     BEQ SGN
0250     CMP #'-
0260     BEQ SGN
0270 EXS PLA
0280     SEC
0290     RTS
0300 SGN JSR INBCN
0310     BCC EXC
0320     CMP #'.
0330     BNE EXC
```

```
0340 DPT JSR INBCN
0350     BCC EXC
0360     BCS EXS
0370 EXC PLA
0380     STA CIX
0390     CLC
0400     RTS
```

Procedura ASCSS sprawdza poprawność odczytanego znaku. Gdy znak jest nieprawidłowy, to przed opuszczeniem procedury ustawiany jest bit Carry. Poprawnymi znakami są cyfry oraz punkt dziesiętny lub znaki "+" i "-", jeśli następuje po nich cyfra. W celu odczytania znaku ASCII i jego zamiany na bajt BCD wywoływana jest procedura INBCN.

```
0100 ;INput Buffer CoNvert
0110 ;
0120 ADBT =  $DBAF
0130 CIX =   $F2
0140 INBUFP = $F3
0150 ;
0160     *=  $DB94
0170 ;
0180     JSR ADBT
0190     LDY CIX
0200     BCC INCIX
0210     LDA (INBUFP),Y
0220 INCIX INY
0230     STY CIX
0240     RTS
```

Procedura INBCN najpierw wywołuje procedurę ADBT, która odczytuje cyfre z ciągu znaków ASCII, a następnie zwiększa o jeden zawartość rejestru CIX.

```
0100 ;ASCII Digit to ByTe
0110 ;
0120 CIX =   $F2
0130 INBUFP = $F3
0140 ;
0150     *=  $DBAF
0160 ;
0170     LDY CIX
0180     LDA (INBUFP),Y
0190     SEC
0200     SBC #$30
0210     BCC EXIT
0220     CMP #$0A
0230     RTS
0240 ;
0250     *=  $DBD0
0260 ;
0270 EXIT SEC
0280     RTS
```

Właściwej zamiany cyfry w kodzie ASCII na cztery bity kodu BCD dokonuje procedura ADBT. Odczytuje ona najpierw z bufora wejściowego wskazywanego wektorem INBUFP kolejny znak wyznaczony indeksem CIX i odejmuje od niego $30 (kod ASCII cyfry 0). Jeżeli uzyskany wynik

jest mniejszy od zera lub większy od 9 (a więc nie jest to cyfra), to przed opuszczeniem procedury ustawiany jest bit Carry.

Należy zwrócić uwagę, że sekwencja rozkazów SEC, RTS (oznaczona etykietą EXIT) znajduje się w procedurze ASCSS. Zaoszczędzono w ten sposób dwa bajty w pamięci.

Ostatnią z procedur przekształcających liczby FP jest procedura FASC, której zadaniem jest konwersja liczby FP na ciąg znaków ASCII. Liczba do konwersji musi być umieszczona w rejestrze FR0, a wynikowy ciąg zapisywany jest w buforze LBUFF.

```
0100 ;FP to ASCII conversion
0110 ;
0120 BTAD =  $DC9D
0130 CIX =   $F2
0140 DECIBP = $DCC1
0150 EEXP =  $ED
0160 FR0 =   $D4
0170 INBUFP = $F3
0180 LBPR2 = $057F
0190 LBSR =  $DCA4
0200 LBUFF = $0580
0210 STALB = $DC70
0220 STBV =  $DA51
0230 STLB =  $DC9F
0240 ;
0250     *=  $D8E6
0260 ;
0270     JSR STBV
0280     LDA #$30
0290     STA LBPR2
0300     LDA FR0
0310     BEQ ZERO
0320     AND #$7F
0330     CMP #$3F
0340     BCC CPL
0350     CMP #$45
0360     BCS CPL
0370     SEC
0380     SBC #$3F
0390     JSR STALB
0400     JSR LBSR
0410     ORA #$80
0420     STA LBUFF,X
0430     LDA LBUFF
0440     CMP #'.
0450     BEQ DIB
0460     JMP ADB

0840     STA EEXP
0850     LDA #'-
0860     BNE STR
0870 PLS LDA #'+
0880 STR JSR STLB
0890     LDX #$00
0900     LDA EEXP
0470 DIB JSR DECIBP
0480     JMP SGN
0490 ZERO LDA #'0+$80
0500     STA LBUFF
0510     RTS
0520 CPL LDA #$01
0530     JSR STALB
0540     JSR LBSR
0550     INX
0560     STX CIX
0570     LDA FR0
0580     ASL A
0590     SEC
0600     SBC #$80
0610     LDX LBUFF
0620     CPX #$30
0630     BEQ EXP
0640     LDX LBUFF+1
0650     LDY LBUFF+2
0660     STX LBUFF+2
0670     STY LBUFF+1
0680     LDX CIX
0690     CPX #$02
0700     BNE NIN
0710     INC CIX
0720 NIN CLC
0730     ADC #$01
0740 EXP STA EEXP
0750     LDA #'E
0760     LDY CIX
0770     JSR STLB
0780     STY CIX
0790     LDA EEXP
0800     BPL PLS
0810     LDA #$00
0820     SEC
0830     SBC EEXP

1030     JSR BTAD
1040 ADB LDA LBUFF
1050     CMP #$30
1060     BNE SGN
1070     CLC
1080     LDA INBUFP
1090     ADC #$01
```

```
0910 NXT SEC                    1100     STA INBUFP
0920     SBC #$0A               1110     LDA INBUFP+1
0930     BCC FIN                1120     ADC #$00
0940     INX                    1130     STA INBUFP+1
0950     BNE NXT                1140 SGN LDA FR0
0960 FIN CLC                    1150     BPL EXIT
0970     ADC #$0A               1160     JSR DECIBP
0980     PHA                    1170     LDY #$00
0990     TXA                    1180     LDA #'-
1000     JSR BTAD               1190     STA (INBUFP),Y
1010     PLA                    1200 EXIT RTS
1020     ORA #$80
```

Na początku poprzez wywołanie procedury STBV ustalany jest adres bufora, w którym zostanie zapisany wynikowy ciąg znaków ASCII i do poprzedzającego bufor rejestru LBPR2 (Line Buffer PRefix 2) wpisywane jest zero.

Następnie odczytywany jest bajt wykładnika liczby FP. Gdy jest on równy 0, to do bufora wpisywany jest znak zera zwiększony o $80 (w inverse video) i procedura się kończy. W przeciwnym razie jest sprawdzane, czy wykładnik mieści się w zakresie od 0 do 4.

Jeżeli tak, to liczba FP jest bezpośrednio zamieniana na ciąg ASCII. Potem ustalane jest jeszcze tylko położenie punktu dziesiętnego i znak liczby.

Gdy wykładnik jest mniejszy od zera (liczba mniejsza od 0) lub większy od 4 (liczba ma więcej niż osiem cyfr przed punktem dziesiętnym), to ciąg wynikowy będzie zawierał liczbę w notacji naukowej (tzn. mantysa i wykładnik potęgi 10). W takim przypadku najpierw odtwarzane są cyfry znaczące, a następnie wartość wykładnika potęgi 100 jest przeliczana na wartość wykładnika potęgi 10.

Na końcu (niezależnie od wielkości przekształcanej liczby) otrzymany ciąg jest przeszukiwany w celu odnalezienia pierwszej cyfry różnej od zera i na tą cyfrę ustawiany jest wektor bufora. Teraz sprawdzany jest znak liczby i gdy liczba jest ujemna, to wektor bufora jest zmniejszany o jeden i przed pierwszą cyfrą wpisywany jest znak "-".

Poniżej przedstawione są pomocnicze procedury wykorzystywane przez FASC i inne procedury FP.

```
0100 ;STore Buffer Vector
0110 ;
0120 INBUFP = $F3
0130 LBUFF = $0580
0140 ;
0150     *=  $DA51
0160 ;
0170     LDA # >LBUFF
0180     STA INBUFP+1
0190     LDA # <LBUFF
```

```
0200     STA INBUFP
0210     RTS
```

Jedynym zadaniem procedury STBV jest wpisanie do wektora INBUFP (INput BUFfer Pointer) adresu bufora wyjściowego LBUFF (Line BUFFer). Zapewnia to umieszczenie wynikowego ciągu ASCII w buforze LBUFF.

```
0100 ;DECrement Input Buffer Pointer
0110 ;
0120 INBUFP = $F3
0130 ;
0140     *=  $DCC1
0150 ;
0160     SEC
0170     LDA INBUFP
0180     SBC #$01
0190     STA INBUFP
0200     LDA INBUFP+1
0210     SBC #$00
0220     STA INBUFP+1
0230     RTS
```

Procedura DECIBP zmniejsza o jeden wektor INBUFP. Wektor ten jest aktualizowany po wpisaniu do bufora każdego znaku i wskazuje zawsze miejsce wpisania następnego znaku. Po wywołaniu DECIBP wektor wskazuje więc na ostatni znak ciągu wynikowego.

```
0100 ;STore ASCII to LBUFF
0110 ;
0120 BTAD =  $DC9D
0130 FR0 =   $D4
0140 STLB =  $DC9F
0150 ZTEMP2 = $F7
0160 ;
0170     *=  $DC70
0180 ;
0190     STA ZTEMP2
0200     LDX #$00
0210     LDY #$00
0220 NXT JSR DP
0230     SEC
0240     SBC #$01
0250     STA ZTEMP2
0260     LDA FR0+1,X
0270     LSR A
0280     LSR A
0290     LSR A
0300     LSR A
0310     JSR BTAD
0320     LDA FR0+1,X
0330     AND #$0F
0340     JSR BTAD
0350     INX
0360     CPX #$05
0370     BCC NXT
0380 DP  LDA ZTEMP2
0390     BNE EXIT
```

```
0400     LDA #'.
0410     JSR STLB
0420 EXIT RTS
```

Przed wywołaniem procedury STALB w akumulatorze zostaje umieszczona liczba określająca ilość cyfr przed punktem dziesiętnym podzielona przez dwa. Po wywołaniu liczba ta zostaje przepisana do rejestru ZTEMP2. Teraz kolejno odczytywane są bajty mantysy z rejestru FR0. Z każdego z nich wydzielane są najpierw cztery starsze, a potem cztery młodsze bity. Zawarte w nich cyfry kodu BCD są zamieniane przez procedurę BTAD na znaki ASCII i umieszczane w buforze wyjściowym.

Przed odczytaniem każdego bajtu liczby FP zawartość rejestru ZTEMP2 jest zmniejszana o 1 i gdy osiągnie wartość 0, to do bufora wpisywany jest punkt dziesiętny (kropka - ".").

```
0100 ;ByTe to ASCII Digit
0110 ;
0120 LBUFF = $0580
0130 ;
0140     *=  $DC9D
0150 ;
0160     ORA #$30
0170 STLB STA LBUFF,Y
0180     INY
0190     RTS
```

Wspomniana już wcześniej procedura BTAD dodaje $30 do wartości znajdującej się w akumulatorze cyfry, przez co uzyskujemy kod ASCII tej cyfry. Następnie wpisuje zawartość akumulatora do buforu LBUFF w miejsce określone przez zawartość rejestru Y. Ta operacja jest też wykorzystywana oddzielnie przez wywołanie procedury od etykiety STLB ($DC9F) zamiast od początku.

```
0100 ;Line Buffer SeaRch
0110 ;
0120 LBUFF = $0580
0130 ;
0140     *=  $DCA4
0150 ;
0160     LDX #$0A
0170 NXT LDA LBUFF,X
0180     CMP #'.
0190     BEQ FIN
0200     CMP #'0
0210     BNE EXIT
0220     DEX
0230     BNE NXT
0240 FIN DEX
0250     LDA LBUFF,X
0260 EXIT RTS
```

Procedura LBSR przeszukuje wstecz od dziesiątego znaku bufor wyjściowy i zwraca w akumulatorze pierwszy napotkany znak różny od punktu dziesiętnego i od zera. Rejestr X zawiera

wtedy indeks tego znaku od początku bufora.

Po przekształceniu liczby na format FP może się zdarzyć, że uzyskany wynik nie odpowiada w pełni wymaganemu formatowi. Sześciobajtowa liczba zmiennoprzecinkowa musi mieć pierwszy bajt mantysy różny od zera. Niektóre z procedur konwersji nie spełniają tego wymagania i dlatego wywołują procedurę NFR0, która poprawia format liczby FP.

```
0100 ;Normalize FR0
0110 ;
0120 FR0 =   $D4
0130 FRE =   $DA
0140 ZFR0 =  $DA44
0150 ;
0160     *=  $DC00
0170 ;
0180     LDX #$00
0190     STX FRE
0200 NFR0A LDX #$04
0210     LDA FR0
0220     BEQ EXIT
0230 NX1 LDA FR0+1
0240     BNE BPS
0250     LDY #$00
0260 NX2 LDA FR0+2,Y
0270     STA FR0+1,Y
0280     INY
0290     CPY #$05
0300     BCC NX2
0310     DEC FR0
0320     DEX
0330     BNE NX1
0340     LDA FR0+1
0350     BNE BPS
0360     STA FR0
0370     CLC
0380     RTS
0390 BPS LDA FR0
0400     AND #$7F
0410     CMP #$71
0420     BCC NEX
0430     RTS
0440 NEX CMP #$0F
0450     BCS EXIT
0460     JSR ZFR0
0470 EXIT CLC
0480     RTS
```

Na początku sprawdzany jest wykładnik liczby FP i gdy jest on równy zero, to cała liczba jest równa zero i następuje opuszczenie procedury. Jeżeli liczba jest różna od zera to rozpoczyna się pętla normalizowania formatu mantysy.

Gdy pierwszy bajt mantysy jest równy zero, to cała mantysa jest przesuwana o jeden bajt, a wykładnik zmniejsza się o 1. Operacja ta powtarzana jest czterokrotnie, chyba że w kolejnym

przejściu zostanie stwierdzona niezerowa wartość pierwszego bajtu mantysy. Jeśli po tym mantysa nadal jest równa zero, to cała liczba otrzymuje przez wyzerowanie wykładnika wartość zero i procedura się kończy.

Gdy wartość pierwszego bajtu mantysy jest różna od zera, następuje sprawdzenie wartości wykładnika. Wykładnik większy od 49 oznacza przekroczenie dopuszczalnego zakresu wartości i procedura kończy się z ustawionym bitem Carry. Wykładnik mniejszy od -49 także oznacza przekroczenie dozwolonego zakresu wartości, lecz nie wywołuje błędu, ale powoduje wyzerowanie całego rejestru FR0. Zakończenie procedury ze skasowanym bitem Carry oznacza, że liczba FP jest równa zero lub mieści się w dopuszczalnym zakresie.

Ostatnią z procedur pomocniczych jest ZFR0, która służy do zerowania całego rejestru FR0.

```
0100 ;set Zero to FR0
0110 ;
0120 FR0 =   $D4
0130 ;
0140     *=  $DA44
0150 ;
0160     LDX #FR0
0170     LDY #$06
0180 AF1 LDA #$00
0190 NXT STA $00,X
0200     INX
0210     DEY
0220     BNE NXT
0230     RTS
```

Jej dodatkowym zastosowaniem jest zerowanie dowolnego obszaru zerowej strony pamięci od adresu podanego w rejestrze X i o długości określonej przez rejestr Y. Wymaga to uprzedniego umieszczenia odpowiednich wartości w rejestrach X i Y oraz wywołanie procedury od etykiety AF1 ($DA48). Niektóre źródła podają wartość $DA46 jako adres AF1, lecz takie wywołanie nie występuje w żadnym miejscu pakietu procedur zmiennoprzecinkowych.

4.2.2. Procedury przemieszczeń liczb FP

Druga ważna grupa procedur zmiennoprzecinkowych to procedury przemieszczeń liczb w formacie FP. Mamy w tej grupie procedury zapisu, odczytu i przenoszenia liczb FP. Można je łatwo wykorzystać we własnych programach, niekoniecznie w całości.

```
0100 ;FP number LoaD to FR0
0110 ;using X,Y Registers
0120 ;
0130 FLPTR = $FC
0140 FR0 =   $D4
0150 ;
0160     *=  $DD89
0170 ;
0180     STX FLPTR
0190     STY FLPTR+1
```

```
0200 FLD0P LDY #$05
0210 NXT LDA (FLPTR),Y
0220     STA FR0,Y
0230     DEY
0240     BPL NXT
0250     RTS
```

Pierwsza procedura odczytuje sześciobajtową liczbę FP i umieszcza ją w rejestrze FR0. Liczba jest pobierana z miejsca, którego starszy bajt zawarty jest w rejestrze Y, a młodszy w rejestrze X. Wywołanie tej procedury od etykiety FLD0P (FP number LoaD into FR0 using Pointer - $DD8D) spowoduje przeniesienie do FR0 liczby, której adres jest wskazany wektorem FLPTR (FLoating PoinTeR).

```
0100 ;FP number LoaD to FR1
0110 ;using X,Y Registers
0120 ;
0130 FLPTR = $FC
0140 FR1 =   $E0
0150 ;
0160     *=  $DD98
0170 ;
0180     STX FLPTR
0190     STY FLPTR+1
0200 FLD1P LDY #$05
0210 NXT LDA (FLPTR),Y
0220     STA FR1,Y
0230     DEY
0240     BPL NXT
0250     RTS
```

Działanie procedury FLD1R jest analogiczne, jedynie liczba FP jest umieszczana w rejestrze FR1. Również tutaj istnieje drugie wejście do procedury oznaczone etykietą FLD1P (FP number LoaD into FR1 using Pointer - $DD9C).

```
0100 ;FP number STore from FR0
0110 ;using X,Y Registers
0120 ;
0130 FLPTR = $FC
0140 FR0 =   $D4
0150 ;
0160     *=  $DDA7
0170 ;
0180     STX FLPTR
0190     STY FLPTR+1
0200 FST0P LDY #$05
0210 NXT LDA FR0,Y
0220     STA (FLPTR),Y
0230     DEY
0240     BPL NXT
0250     RTS
```

Funkcję odwrotną w stosunku do FLD0R pełni procedura FST0R, która zapisuje we wskazanym miejscu pamięci zawartość rejestru FR0. Przy wywołaniu od etykiety FST0R jako wektor wykorzystywana jest zawartość rejestrów X i Y, a przy wywołaniu od FST0P (FP number STore

using Pointer - $DDAB) zawartość rejestru FLPTR.

Kolejne trzy procedury służą do przemieszczania liczb FP pomiędzy poszczególnymi rejestrami FR.

```
0100 ;FP number MOVe
0110 ;from FR0 to FR1
0120 ;
0130 FR0 =   $D4
0140 FR1 =   $E0
0150 ;
0160     *=  $DDB6
0170 ;
0180     LDX #$05
0190 NXT LDA FR0,X
0200     STA FR1,X
0210     DEX
0220     BPL NXT
0230     RTS
```

Procedura FPMOV01 przepisuje zawartość rejestru FR0 do rejestru FR1.

```
0100 ;FP number MOVe
0110 ;from FR0 to FRE
0120 ;
0130 FR0 =   $D4
0140 FRE =   $DA
0150 ;
0160     *=  $DD34
0170 ;
0180     LDY #$05
0190 NXT LDA FR0,Y
0200     STA FRE,Y
0210     DEY
0220     BPL NXT
0230     RTS
```

Procedura FPMOV0E przepisuje liczbę w formacie FP z rejestru FR0 do rejestru FRE.

```
0100 ;FP number MOVe
0110 ;from FR1 to FR2
0120 ;
0130 FR1 =   $E0
0140 FR2 =   $E6
0150 ;
0160     *=  $DD28
0170 ;
0180     LDY #$05
0190 NXT LDA FR1,Y
0200     STA FR2,Y
0210     DEY
0220     BPL NXT
0230     RTS
```

Procedura FPMOV12 przepisuje sześć bajtów z rejestru FR1 do rejestru FR2.

4.2.3. Procedury obliczeń zmiennoprzecinkowych

Najważniejszą częścią pakietu arytmetyki zmienno- przecinkowej są procedury wykonujące obliczenia na liczbach FP. Pakiet zawiera procedury czterech działań podstawowych oraz potęgowania i logarytmowania.

Procedury dodawania (FADD) i odejmowania (FSUB) wymagają, aby argumenty działania znajdowały się w rejestrach FR0 i FR1. Różnią się one tylko początkiem - procedura FSUB najpierw zmienia znak liczby zawartej w FR1 - potem obie są identyczne.

```
0100 ADJ0 =  $DC3A
0110 ADJ1 =  $DC3E
0120 FR0 =   $D4
0130 FR1 =   $E0
0140 NFR0 =  $DC00
0150 ZTEMP2 = $F7
0160 ;
0170 ;Floating point SUBtraction
0180 ;
0190     *=  $DA60
0200 ;
0210 FSUB LDA FR1
0220     EOR #$80
0230     STA FR1
0240 ;
0250 ;Floating point ADDition
0260 ;
0270 FADD LDA FR1
0280     AND #$7F
0290     STA ZTEMP2
0300     LDA FR0
0310     AND #$7F
0320     SEC
0330     SBC ZTEMP2
0340     BPL PLUS
0350     LDX #$05
0360 NX1 LDA FR0,X
0370     LDY FR1,X
0380     STA FR1,X
0390     TYA
0400     STA FR0,X
0410     DEX
0420     BPL NX1
0430     BMI FADD
0440 PLUS BEQ BP1
0450     CMP #$05
0460     BCS BP2
0470     JSR ADJ1
0480 BP1 SED
0490     LDA FR0
0500     EOR FR1
0510     BMI BP3
0520     LDX #$04
0530     CLC
0540 NX2 LDA FR0+1,X
0550     ADC FR1+1,X
```

```
0560     STA FR0+1,X
0570     DEX
0580     BPL NX2
0590     CLD
0600     BCS ADJ
0610 BP2 JMP NFR0
0620 ADJ LDA #$01
0630     JSR ADJ0
0640     LDA #$01
0650     STA FR0+1
0660     JMP NFR0
0670 BP3 LDX #$04
0680     SEC
0690 NX3 LDA FR0+1,X
0700     SBC FR1+1,X
0710     STA FR0+1,X
0720     DEX
0730     BPL NX3
0740     BCC BP4
0750     CLD
0760     JMP NFR0
0770 BP4 LDA FR0
0780     EOR #$80
0790     STA FR0
0800     SEC
0810     LDX #$04
0820 NX4 LDA #$00
0830     SBC FR0+1,X
0840     STA FR0+1,X
0850     DEX
0860     BPL NX4
0870     CLD
0880     JMP NFR0
```

Ponieważ pierwszy argument (zawarty w FR0) nie może być mniejszy od drugiego (FR1), to najpierw porównywane są ich wykładniki. Gdy powyższe wymaganie jest niespełnione, liczby są zamieniane miejscami.

Gdy różnica między wykładnikami jest większa od 4, to wykonanie działania nie zmieni wartości cyfr znaczących mantysy większego argumentu. W takim przypadku następuje bezpośredni skok do procedury NFR0.

Różnica między wykładnikami mniejsza od 5 powoduje wywołanie procedury ADJ1, która wyrównuje wartości wykładników i odpowiednio przesuwa bajty mantysy liczby zawartej w FR1.

Jeśli liczby mają równe wykładniki i jednakowy znak, to kolejne bajty ich mantys są dodawane. Wystąpienie przepełnienia podczas dodawania sygnalizuje konieczność przesunięcia mantysy i zwiększenia wartości wykładnika.

Gdy liczby różnią się znakiem, to kolejne bajty ich mantys są odejmowane. Brak przepełnienia (bit Carry skasowany) po tej operacji sygnalizuje zmianę znaku wyniku. W takim razie wszystkie

bajty mantysy muszą być jeszcze zamienione na ich uzupełnienie.

W każdym przypadku procedury FADD i FSUB nie kończą się rozkazem RTS, lecz bezpośrednim skokiem do procedury NFR0, która doprowadza wynik do prawidłowego formatu.

Wykonanie dodawania lub odejmowania wymaga, aby wykładniki argumentów były jednakowe. Wyrównanie wykładników przeprowadzane jest przez procedury ADJ0 i ADJ1. Przed wywołaniem tych procedur w akumulatorze musi być umieszczona różnica między wykładnikami.

```
0100 ;ADJust FR0 or FR1
0110 ;
0120 FR0 =   $D4
0130 FR1 =   $E0
0140 ZTEMP2 = $F7
0150 ZTEMP3 = $F9
0160 ;
0170     *=  $DC3A
0180 ;
0190 ADJ0 LDX #FR0
0200     BNE BPS
0210 ADJ1 LDX #FR1
0220 BPS STX ZTEMP3
0230     STA ZTEMP2
0240     STA ZTEMP2+1
0250 NX1 LDY #$04
0260 NX2 LDA $04,X
0270     STA $05,X
0280     DEX
0290     DEY
0300     BNE NX2
0310     LDA #$00
0320     STA $05,X
0330     LDX ZTEMP3
0340     DEC ZTEMP2
0350     BNE NX1
0360     LDA $00,X
0370     CLC
0380     ADC ZTEMP2+1
0390     STA $00,X
0400     RTS
```

Na początku zawartość akumulatora jest zapisywana do obu bajtów rejestru ZTEMP2. Pierwszy bajt jest wykorzystywany jako licznik pętli, a drugi posłuży do uaktualnienia wartości wykładnika. Teraz wymaganą liczbę razy bajty mantysy są przesuwane w prawo i po dodaniu do wykładnika zawartości drugiego bajtu ZTEMP2 procedura się kończy.

Druga para procedur obliczeniowych (FMUL i FDIV) ma wspólny jedynie koniec. Jednak ze względu na położenie w pamięci i podobieństwo funkcji zostaną one opisane razem.

```
0100 ADD01 = $DD01
0110 ADD02 = $DD05
0120 ADDE1 = $DD09
```

```
0130 ADDE2 = $DD0F
0140 EEXP =  $ED
0150 MVARG = $DCE0
0160 FR0 =   $D4
0170 FR1 =   $E0
0180 FR2 =   $E6
0190 FRE =   $DA
0200 NFR0A = $DC04
0210 SGNEV = $DCCF
0220 SHR0 =  $DC62
0230 ZFR0 =  $DA44
0240 ZTEMP1 = $F5
0250 ;
0260 ;Floating point MULtiplication
0270 ;
0280     *=  $DADB
0290 ;
0300 FMUL LDA FR0
0310     BEQ EXC
0320     LDA FR1
0330     BEQ EXZ
0340     JSR SGNEV
0350     SEC
0360     SBC #$40
0370     SEC
0380     ADC FR1
0390     BMI EXS
0400     JSR MVARG
0410 NX1 LDA FRE+5
0420     AND #$0F
0430     STA ZTEMP1+1
0440 LP1 DEC ZTEMP1+1
0450     BMI EN1
0460     JSR ADD01
0470     JMP LP1
0480 EN1 LDA FRE+5
0490     LSR A
0500     LSR A
0510     LSR A
0520     LSR A
0530     STA ZTEMP1+1
0540 LP2 DEC ZTEMP1+1
0550     BMI EN2
0560     JSR ADD02
0570     JMP LP2
0580 EN2 JSR SHR0
0590     DEC ZTEMP1
0600     BNE NX1
0610 EXIT LDA EEXP
0620     STA FR0
0630     JMP NFR0A
0640 EXZ JSR ZFR0
0650 EXC CLC
0660     RTS
0670 EXS SEC
0680     RTS
0690 ;
0700 ;Floating point DIVision
0710 ;
```

```
0720 FDIV LDA FR1
0730     BEQ EXS
0740     LDA FR0
0750     BEQ EXC
0760     JSR SGNEV
0770     SEC
0780     SBC FR1
0790     CLC
0800     ADC #$40
0810     BMI EXS
0820     JSR MVARG
0830     INC ZTEMP1
0840     JMP BPS
0850 LP3 LDX #$00
0860 NX2 LDA FR0+1,X
0870     STA FR0,X
0880     INX
0890     CPX #$0C
0900     BNE NX2
0910 BPS LDY #$05
0920     SEC
0930     SED
0940 NX3 LDA FRE,Y
0950     SBC FR2,Y
0960     STA FRE,Y
0970     DEY
0980     BPL NX3
0990     CLD
1000     BCC LP4
1010     INC FR0+5
1020     BNE BPS
1030 LP4 JSR ADDE2
1040     ASL FR0+5
1050     ASL FR0+5
1060     ASL FR0+5
1070     ASL FR0+5
1080 NX4 LDY #$05
1090     SEC
1100     SED
1110 NX5 LDA FRE,Y
1120     SBC FR1,Y
1130     STA FRE,Y
1140     DEY
1150     BPL NX5
1160     CLD
1170     BCC LP5
1180     INC FR0+5
1190     BNE NX4
1200 LP5 JSR ADDE1
1210     DEC ZTEMP1
1220     BNE LP3
1230     JSR SHR0
1240     JMP EXIT
```

Po wywołaniu obu procedur najpierw sprawdzane są wykładniki argumentów. Jeżeli wykładnik pierwszej liczby jest równy zeru, to procedura się kończy. Gdy zerem jest wykładnik drugiej liczby, to w przypadku mnożenia pierwsza liczba jest zerowana i również następuje opuszczenie

procedury, a przy dzieleniu przed końcem procedury bit Carry jest ustawiany w celu zasygnalizowania błędu.

Jeśli oba wykładniki są niezerowe, to wywoływana jest procedura SGNEV, która oblicza znak wyniku i zapisuje go w rejestrze NSIGN (Number SIGN). Znaki obu argumentów są następnie kasowane.

```
0100 ;SiGN EValuation
0110 ;
0120 FR0 =   $D4
0130 FR1 =   $E0
0140 NSIGN = $EE
0150 ;
0160     *=  $DCCF
0170 ;
0180     LDA FR0
0190     EOR FR1
0200     AND #$80
0210     STA NSIGN
0220     ASL FR1
0230     LSR FR1
0240     LDA FR0
0250     AND #$7F
0260     RTS
```

Po zakończeniu SGNEV obliczany jest wykładnik wyniku. Jeżeli przekracza on dopuszczalny zakres, to procedura jest przerywana z ustawionym bitem Carry. W przeciwnym razie wywoływana jest procedura MVARG.

```
0100 ;MoVe ARGuments
0110 ;
0120 EEXP =  $ED
0130 FMOV0E = $DD34
0140 FMOV12 = $DD28
0150 FR0 =   $D4
0160 FR1 =   $E0
0170 FR2 =   $E6
0180 FRX =   $EC
0190 NSIGN = $EE
0200 ROLFR2 = $DBE7
0210 ZFR0 =  $DA44
0220 ZTEMP1 = $F5
0230 ;
0240     *=  $DCE0
0250 ;
0260     ORA NSIGN
0270     STA EEXP
0280     LDA #$00
0290     STA FR0
0300     STA FR1
0310     JSR FMOV12
0320     JSR ROLFR2
0330     LDA FRX
0340     AND #$0F
0350     STA FR2
```

```
0360     LDA #$05
0370     STA ZTEMP1
0380     JSR FMOV0E
0390     JSR ZFR0
0400     RTS
```

Najpierw dodaje ona znak do obliczonego bajtu wykładnika i całość zapisuje w rejestrze EEXP. Następnie zeruje wykładniki obu argumentów, a argumenty przepisuje z FR0 do FRE i z FR1 do FR2. Zawartość FR2 jest przy tym przesuwana w lewo o cztery bity tak, że pierwsza cyfra mantysy znajduje się teraz w bajcie wykładnika. Przed końcem MVARG zerowany jest jeszcze rejestr FR0.

Dalszy przebieg obu procedur (FMUL i FDIV) jest także bardzo podobny. Argumenty działania są dodawane lub odejmowane w pętli, aż do osiągnięcia prawidłowego wyniku. Zastosowane jest tu wielokrotne dodawanie zamiast mnożenia i wielokrotne odejmowanie zamiast dzielenia. Do wykonania tych operacji służy procedura NADD, która wywoływana jest zależnie od potrzeby od jednej z czterech etykiet.

```
0100 ;Numbers ADDition
0110 ;
0120 FR0 =   $D4
0130 FR1 =   $E0
0140 FR2 =   $E6
0150 FRE =   $DA
0160 ZTEMP2 = $F7
0170 ;
0180     *=  $DD01
0190 ;
0200 ADD01 LDX #FR0+5
0210     BNE AD1
0220 ADD02 LDX #FR0+5
0230     BNE AD2
0240 ADDE1 LDX #FRE+5
0250 AD1 LDY #FR1+5
0260     BNE CONT
0270 ADDE2 LDX #FRE+5
0280 AD2 LDY #FR2+5
0290 CONT LDA #$05
0300     STA ZTEMP2
0310     CLC
0320     SED
0330 NXT LDA $00,X
0340     ADC $00,Y
0350     STA $00,X
0360     DEX
0370     DEY
0380     DEC ZTEMP2
0390     BPL NXT
0400     CLD
0410     RTS
```

Dodaje ona w trybie dziesiętnym procesora zawartości rejestrów liczb FP parami (zależnie od miejsca wywołania). Po wywołaniu od ADD01 dodaje zawartość FR1 do FR0, od ADD02 ($DD05) FR2 do FR0, od ADDE1 ($DD09) FR1 do FRE, a od ADDE2 ($DD0F) FR2 do FRE.

```
0100 ;SHift Right FR0
0110 ;
0120 FR0 =   $D4
0130 ;
0140     *=  $DC62
0150 ;
0160     LDX #$0A
0170 NXT LDA FR0,X
0180     STA FR0+1,X
0190     DEX
0200     BPL NXT
0210     LDA #$00
0220     STA FR0
0230     RTS
```

Drugą procedurą pomocniczą jest SHR0, która na końcu każdej pętli obliczeń przesuwa w prawo zawartość rejestru FR0 i do bajtu wykładnika wpisuje zero.

Po zakończeniu obliczania mantysy wcześniej obliczony bajt wykładnika jest przepisywany z rejestru EEXP. Teraz w celu uzyskania poprawnego formatu wyniku wykonywany jest bezpośredni skok do procedury NFR0.

Kolejna procedura służy do przeliczeń wielomianowych i jest używana przy obliczeniach logarytmicznych i wykładniczych. Korzysta ona z tabeli współczynników, której adres jest umieszczany w rejestrach X i Y przed wywołaniem PLYARG. Liczba przejść pętli obliczeniowej jest podana w akumulatorze.

```
0100 ;PoLYnomial EVaLuation
0110 ;
0120 ESIGN = $EF
0130 FADD =  $DA66
0140 FLD0R = $DD89
0150 FLD1R = $DD98
0160 FMOV01 = $DDB6
0170 FMUL =  $DADB
0180 FPTR2 = $FE
0190 FST0R = $DDA7
0200 PLYARG = $05E0
0210 ;
0220     *=  $DD40
0230 ;
0240     STX FPTR2
0250     STY FPTR2+1
0260     STA ESIGN
0270     LDX # <PLYARG
0280     LDY # >PLYARG
0290     JSR FST0R
0300     JSR FMOV01
0310     LDX FPTR2
0320     LDY FPTR2+1
0330     JSR FLD0R
0340     DEC ESIGN
0350     BEQ EXIT
0360 LOOP JSR FMUL
```

```
0370     BCS EXIT
0380     CLC
0390     LDA FPTR2
0400     ADC #$06
0410     STA FPTR2
0420     BCC BPS
0430     LDA FPTR2+1
0440     ADC #$00
0450     STA FPTR2+1
0460 BPS LDX FPTR2
0470     LDY FPTR2+1
0480     JSR FLD1R
0490     JSR FADD
0500     BCS EXIT
0510     DEC ESIGN
0520     BEQ EXIT
0530     LDX # <PLYARG
0540     LDY # >PLYARG
0550     JSR FLD1R
0560     BMI LOOP
0570 EXIT RTS
```

Procedura PLYEVL wykonuje obliczenie wg wzoru:

$$((...((A_1*x+A_2)*x+A_3)...)*x+A_n)$$

gdzie x jest zawartością rejestru FR0 w chwili wywołania procedury (przechowywana w rejestrze PLYARG), a A1,2...n są kolejnymi współczynnikami z tabeli. Obliczenie jest wykonywane aż do wystąpienia przepełnienia lub do wyzerowania licznika przepisanego na początku procedury z akumulatora do rejestru ESIGN.

Procedura potęgowania posiada dwa punkty początkowe. Po wywołaniu od etykiety EXP ($DDC0) podnosi liczbę e do potęgi zawartej w FR0, zaś po wywołaniu od EXP10 ($DDCC) bazą potęgowania jest liczba 10.

```
0100 ;EXPonentation
0110 ;
0120 DIGRT = $F1
0130 FCHRFLG = $F0
0140 FDIV =  $DB28
0150 FLD0R = $DD89
0160 FLD1R = $DD98
0170 FMOV01 = $DDB6
0180 FMUL =  $DADB
0190 FPI =   $D9D2
0200 FR0 =   $D4
0210 FR1 =   $E0
0220 FR2 =   $E6
0230 FST0R = $DDA7
0240 FSUB =  $DA60
0250 IFP =   $D9AA
0260 PLYEVL = $DD40
0270 ;
0280     *=  $DDC0
0290 ;
```

```
0300 EXP LDX #$89
0310     LDY #$DE
0320     JSR FLD1R
0330     JSR FMUL
0340     BCS EX2
0350 EXP10 LDA #$00
0360     STA DIGRT
0370     LDA FR0
0380     STA FCHRFLG
0390     AND #$7F
0400     STA FR0
0410     SEC
0420     SBC #$40
0430     BMI EVL
0440     CMP #$04
0450     BPL EX2
0460     LDX #FR2
0470     LDY #$05
0480     JSR FST0R
0490     JSR FPI
0500     LDA FR0
0510     STA DIGRT
0520     LDA FR0+1
0530     BNE EX2
0540     JSR IFP
0550     JSR FMOV01
0560     LDX #FR2
0570     LDY #$05
0580     JSR FLD0R
0590     JSR FSUB
0600 EVL LDA #$0A
0610     LDX #$4D
0620     LDY #$DE
0630     JSR PLYEVL
0640     JSR FMOV01
0650     JSR FMUL
0660     LDA DIGRT
0670     BEQ FIN
0680     CLC
0690     ROR A
0700     STA FR1
0710     LDA #$01
0720     BCC BPS
0730     LDA #$10
0740 BPS STA FR1+1
0750     LDX #04
0760     LDA #$00
0770 NXT STA FR1+2,X
0780     DEX
0790     BPL NXT
0800     LDA FR1
0810     CLC
0820     ADC #$40
0830     BCS EX2
0840     BMI EX2
0850     STA FR1
0860     JSR FMUL
0870 FIN LDA FCHRFLG
0880     BPL EX1
```

```
0890     JSR FMOV01
0900     LDX #$8F
0910     LDY #$DE
0920     JSR FLD0R
0930     JSR FDIV
0940 EX1 RTS
0950 EX2 SEC
0960     RTS
```

Przy potęgowaniu liczby e wartość potęgi jest najpierw mnożona przez współczynnik z tabeli. Po tej operacji oba warianty potęgowania przebiegają identycznie.

Teraz sprawdzany jest znak współczynnika liczby zawartej w FR0. Dodatni wykładnik wymaga jeszcze sprawdzenia zakresu. Gdy jest większy od 3, tzn. liczba jest większa od 1000000, to wartość wyniku potęgowania przekroczy zakres dopuszczalny dla liczb FP. Następnie argument jest zamieniany na dwubajtową liczbę całkowitą i jeśli starszy bajt jest niezerowy, to również przekroczony zostanie dozwolony zakres wartości wyniku. W obu tych przypadkach ustawiany jest bit Carry i procedura się kończy.

Jeżeli wartość argumentu jest poprawna, to rozpoczyna się obliczanie wyniku. Są przy tym wykorzystywane procedury FMUL, FDIV i PLYEVL, lecz samo obliczenie jest dość skomplikowane i jego opis zostanie pominięty. Czytelnicy zainteresowani tym zagadnieniem mogą skorzystać z zamieszczonego programu źródłowego procedury.

Także procedura logarytmowania rozpoczyna się w dwóch różnych miejscach, zależnie od podstawy logarytmu. Od etykiety LOG ($DECD) obliczany jest logarytm naturalny, a od LOG10 ($DED1) logarytm dziesiętny.

```
0100 ;LOGarithm
0110 ;
0120 DIGRT = $F1
0130 FADD =  $DA66
0140 FCHRFLG = $F0
0150 FDIV = $DB28
0160 FLD1R = $DD98
0170 FMOV01 = $DDB6
0180 FMUL =  $DADB
0190 FR0 =   $D4
0200 FR1 =   $E0
0210 FR2 =   $E6
0220 FST0R = $DDA7
0230 IFP =   $D9AA
0240 PLYEVL = $DD40
0250 RSQT =  $DE95
0260 ;
0270     *=  $DECD
0280 ;
0290 LOG LDA #$01
0300     BNE CONT
0310 LOG10 LDA #$00
0320 CONT STA FCHRFLG
```

```
0330     LDA FR0
0340     BEQ EXS
0350     BMI EXS
0360     JMP PT1
0370 EXS SEC
0380     RTS
0390 PT2 SBC #$40
0400     ASL A
0410     STA DIGRT
0420     LDA FR0+1
0430     AND #$F0
0440     BNE DIG
0450     LDA #$01
0460     BNE BPS
0470 DIG INC DIGRT
0480     LDA #$10
0490 BPS STA FR1+1
0500     LDX #$04
0510     LDA #$00
0520 NXT STA FR1+2,X
0530     DEX
0540     BPL NXT
0550     JSR FDIV
0560     LDX #$66
0570     LDY #$DF
0580     JSR RSQT
0590     LDX #FR2
0600     LDY #$05
0610     JSR FST0R
0620     JSR FMOV01
0630     JSR FMUL
0640     LDA #$0A
0650     LDX #$72
0660     LDY #$DF
0670     JSR PLYEVL
0680     LDX #FR2
0690     LDY #$05
0700     JSR FLD1R
0710     JSR FMUL
0720     LDX #$6C
0730     LDY #$DF
0740     JSR FLD1R
0750     JSR FADD
0760     JSR FMOV01
0770     LDA #$00
0780     STA FR0+1
0790     LDA DIGRT
0800     STA FR0
0810     BPL CNV
0820     EOR #$FF
0830     CLC
0840     ADC #$01
0850     STA FR0
0860 CNV JSR IFP
0870     BIT DIGRT
0880     BPL ADD
0890     LDA #$80
0900     ORA FR0
0910     STA FR0
```

```
0920 ADD JSR FADD
0930     LDA FCHRFLG
0940     BEQ EXC
0950     LDX #$89
0960     LDY #$DE
0970     JSR FLD1R
0980     JSR FDIV
0990 EXC CLC
1000     RTS
1010 ;
1020     *=  $DFF6
1030 ;
1040 PT1 LDA FR0
1050     STA FR1
1060     SEC
1070     JMP PT2
```

Na początku sprawdzany jest wykładnik argumentu i gdy jest ujemny lub równy zero, to procedura kończy się z sygnalizacją błędu (ustawiony bit Carry). W przeciwnym razie wykładnik jest przepisywany z FR0 do FR1 i rozpoczyna się właściwa część obliczeniowa procedury.

Opis całej procedury logarytmowania również zostanie pominięty, należy jednak zwrócić uwagę na jej podział na trzy części. Zostało to spowodowane wprowadzeniem poprawek do pierwotnej wersji pakietu procedur zmiennoprzecinkowych przy zachowaniu adresów początkowych procedur.

Podczas operacji logarytmowania jest wywoływana jeszcze jedna, wcześniej nie omawiana procedura - RSQT.

```
0100 ;ReSult QuoTient
0110 ;
0120 FADD =  $DA66
0130 FDIV =  $DB28
0140 FLD0R = $DD89
0150 FLD1R = $DD98
0160 FPTR2 = $FE
0170 FR1 =   $E0
0180 FR2 =   $E6
0190 FST0R = $DDA7
0200 FSUB =  $DA60
0210 ;
0220     *=  $DE95
0230 ;
0240     STX FPTR2
0250     STY FPTR2+1
0260     LDX #FR1
0270     LDY #$05
0280     JSR FST0R
0290     LDX FPTR2
0300     LDY FPTR2+1
0310     JSR FLD1R
0320     JSR FADD
0330     LDX #FR2
0340     LDY #$05
0350     JSR FST0R
```

```
0360     LDX #FR1
0370     LDY #$05
0380     JSR FLD0R
0390     LDX FPTR2
0400     LDY FPTR2+1
0410     JSR FLD1R
0420     JSR FSUB
0430     LDX #FR2
0440     LDY #$05
0450     JSR FLD1R
0460     JSR FDIV
0470     RTS
```

Wykonuje ona obliczenie ilorazu różnicowego dwóch argumentów. Adres pierwszego argumentu musi być przed wywołaniem procedury umieszczony w rejestrach X i Y, a drugi argument znajduje się w rejestrze FR1. Wynik obliczony według wzoru

$$\frac{x_1-x_2}{x_1+x_2}$$

jest umieszczany w rejestrze FR0.

Jak wynika z opisu procedur, pakiet arytmetyki FP zawiera jeszcze tabele współczynników potęgowania i logarytmowania. Są one umieszczone w obszarach od $DE4D do $DE94 oraz od $DF66 do $DFF5.

Poza wyżej wymienionymi jeszcze cztery procedury arytmetyki FP wbudowane są do interpretera Atari Basic. Uniemożliwia to korzystanie z nich przy odłączonym interpreterze. Są to:

```
$BDA7   SIN - FP SINus routine
$BDB1   COS - FP COSinus routine
$BE77   ATAN - FP ArcTANgent routine
$BEE5   SQR - FP SQuare Root routine
```

Procedury należące do interpretera Atari Basic są opisane razem z tym interpreterem (zob. "Mapa pamięci ATARI XL/XE. Procedury interpretera Basica").

DODATKI

Dodatek A

Adresy procedur OS

```
$5000 - TESTST - początek testu komputera
$5003 - TESTINI - inicjowanie testu komputera
$BDA7 - SIN - sinus liczby FP
$BDB1 - COS - cosinus liczby FP
$BE77 - ATAN - arcus tangens liczby FP
$BEE5 - SQR - pierwiastek kwadratowy z liczby FP
$C000 - CHSRO1 - suma kontrolna pierwszej części ROM
$C00C - NMIENBL - inicjowanie przerwań MNI
$C018 - NMIFIRST - rozpoznanie przerwania NMI
$C02C - JMPIRQV - skok według wektora VIMIRQ
$C030 - SINDRYI - rozpoznanie przerwania IRQ
$C092 - BREAKIRQ - przerwanie klawisza BREAK
$C0CD - PLARTI - powrót z przerwania
$C0CE - RTI - powrót z przerwania
$C0CF - MASKTAB - tabela masek bitowych dla SINDRYI
$C0D7 - VECTAB - tabela wektorów przerwań dla SINDRYI
$C0DF - WAIT - oczekiwanie na RESET
$C0E2 - SYSVBL - systemowe przerwanie synchronizacji (VBLK)
$C24F - JMPTIM1 - skok według wektora TIMVEC1
$C252 - JMPTIM2 - skok według wektora TIMVEC2
$C255 - DECTIM - zmniejszenie licznika systemowego
$C272 - SETVBLV - ustawianie wektorów przerwania VBLK
$C28A - EXITVBL - zakończenie przerwania VBLK
$C290 - RESETWM - gorący start systemu
$C2AA - RESET - start systemu po naciśnięciu RESET
$C2C8 - RESETCD - zimny start systemu
$C3C1 - GOMEMTST - skok do testu komputera
$C423 - COLDCART - skok według wektora CARTRUN
$C426 - DOSVECC - skok według wektora DOSVEC
$C429 - INITCART - skok według wektora CARTINI
$C42C - CLCRTS - rozkazy CLC i RTS
$C42E - INIT31A - tabela wartości dla HATABS
$C43D - DERRMSG - meldunek błędu "BOOT ERROR"
$C448 - NAME - nazwa edytora "E:"
$C44B - INIT200 - tabela wartości dla wektorów OS
$C471 - CARTGO - sprawdzenie cartridge'a
$C4A9 - GRAMHI - sprawdzenie wielkości RAM
$C4C9 - NEWCART - sprawdzenie cartridge'a
$C4DA - IOPORTIN - inicjowanie portów układów I/O
$C535 - SYSINIT - inicjowanie systemu komputerowego
$C58B - BOOT - wstępny odczyt z dyskietki
$C5BB - BLOCK1 - wstępny odczyt przy zimnym starcie
$C629 - BLOAD - odczyt adresu inicjowania
$C63B - DOSINITC - skok według wektora DOSINI
$C63E - DRDERR - wyświetlenie meldunku błędu
$C642 - PUTLINE - wyświetlenie linii tekstu
$C659 - GETBLK - odczyt bloku danych z urządzenia
$C66E - CASBOOT - wstępny odczyt z kasety
$C6A0 - CASINITC - skok według wektora CASINI
```

```
$C6A3 - DSKINIT - inicjowanie obsługi stacji dysków
$C6B3 - DSKINT - główna procedura dyskowa
$C8FC - SWITROM - wywołanie testu komputera
$C90C - NEWINIT - inicjowanie nowych urządzeń
$C933 - SIOINT - procedura obsługi złącza szeregowego
$C96E - NEWIOREQ - przerwanie IRQ nowego urządzenia
$CA21 - BITMASK - maski bitowe dla NEWIOREQ
$CC00 - CHARSET2 - zestaw znaków międzynarodowych
$D803 - DEVID1 - kod identyfikacyjny nowego urządzenia
$D808 - DEVINT - przerwanie IRQ nowego urządzenia
$D80B - DEVID2 - kod identyfikacyjny nowego urządzenia
$D819 - DEVINIT - inicjowanie nowego urządzenia
$D800 - AFP - zamiana ciągu ASCII na liczbę FP
$D8E6 - FASC - zamiana liczby FP na ciąg ASCII
$D9AA - IFP - zamiana liczby całkowitej na FP
$D9D2 - FPI - zamiana liczby FP na całkowitą
$DA44 - ZFR0 - zerowanie FR0
$DA48 - AF1 - zerowanie wg rejestru X
$DA51 - STBV - zapis wektora bufora
$DA5A - ROLZ2 - przesunięcie w lewo ZTEMP2
$DA60 - FSUB - odejmowanie liczb FP
$DA66 - FADD - dodawanie liczb FP
$DADB - FMUL - mnożenie liczb FP
$DB28 - FDIV - dzielenie liczb FP
$DB94 - INBCN - zamiana znaku z bufora
$DB8D - INCIX - zwiększenie indeksu
$DBA1 - INBSS - przeszukiwanie bufora
$DBAF - ADBT - zamiana znaku ASCII na kod BCD
$DBBB - ASCSS - przeszukiwanie ciągu ASCII
$DBE7 - ROLFR2 - przesunięcie w lewo liczby w FR2
$DBEB - ROLFR0 - przesunięcie w lewo liczby w FR0
$DC00 - NFR0 - poprawienie formatu liczby FP
$DC3A - ADJ0 - adjustowanie liczby z FR0
$DC3E - ADJ1 - adjustowanie liczby z FR1
$DC62 - SHR0 - przesunięcie w prawo FR0
$DC70 - STALB - zapis znaku ASCII do bufora
$DC9D - BTAD - zamiana kodu BCD na znak ASCII
$DC9F - STLB - zapis do bufora
$DCA4 - LBSR - przeszukiwanie bufora
$DCB9 - IDEX - wydzielenie cyfry
$DCC1 - DECIBP - zmniejszenie licznika bufora
$DCCF - SGNEV - obliczenie znaku przy mnożeniu i dzieleniu
$DCE0 - MVARG - przeniesienie argumentów
$DD01 - ADD01 - dodanie rejestrów FR0 i FR1
$DD05 - ADD02 - dodanie rejestrów FR0 i FR2
$DD09 - ADDE1 - dodanie rejestrów FRE i FR1
$DD0F - ADDE2 - dodanie rejestrów FRE i FR2
$DD40 - PLYEVL - przeliczenie wielomianowe
$DD89 - FLD0R - zapis liczby FP do FR0 według X,Y
$DD8D - FLD0P - zapis liczby FP do FR0 według FLPTR
$DD98 - FLD1R - zapis liczby FP do FR1 według X,Y
$DD9C - FLD1P - zapis liczby FP do FR1 według FLPTR
$DDA7 - FST0R - zapis liczby FP z FR0 według X,Y
$DDAB - FST0P - zapis liczby FP z FR0 według FLPTR
$DDB6 - FMOV01 - przepisanie z FR0 do FR1
$DD34 - FMOV0E - przepisanie z FR0 do FRE
$DD28 - FMOV12 - przepisanie z FR1 do FR2
$DDC0 - EXP - potęgowanie o podstawie e
$DDCC - EXP10 - potęgowanie o podstawie 10
```

```
$DE4D - TP10 - tabela współczynników potęgowania
$DE95 - RSQT - iloraz różnicowy
$DECD - LOG - logarytm naturalny
$DED1 - LOG10 - logarytm dziesiętny
$DF66 - TLOG - tabela współczynników logarytmowania
$DFAE - TATAN - tabela współczynników funkcji arctg
$E000 - CHARSET1 - standardowy zestaw znaków
$E400 - EDTVEC - wektory obsługi edytora
$E410 - SCRVEC - wektory obsługi ekranu
$E420 - KBDVEC - wektory obsługi klawiatury
$E430 - PRTVEC - wektory obsługi drukarki
$E440 - CASVEC - wektory obsługi magnetofonu
$E450 - JMPTAB - tabela skoków
$E450 - JDSKINIT - skok do DSKINIT
$E453 - JDSKINT - skok do DSKINT
$E456 - JCIOMAIN - skok do CIOMAIN
$E465 - JSIOINIT - skok do SIOINIT
$E46B - JNMIEN - skok do NMIENBL
$E46E - JCIOINIT - skok do CIOINIT
$E47A - JCASRDBL - skok do CASRDBL
$E47D - JCASOPIN - skok do CASOPIN
$E483 - JTESTST - skok do TESTST
$E49B - NEWINITC - skok do NEWINIT
$E4C1 - CIOINIT - inicjowanie obsługi urządzeń
$E4DC - CIONOPN - kanał I/O nie otwarty
$E4DF - CIOMAIN - procedura obsługi urządzeń
$E739 - LINKSOM - procedura dołączania urządzeń
$E95C - SIOINIT - inicjowanie złącza szeregowego
$EAAD - ISRODN - przerwanie zapisu danych
$EAEC - ISRXD - przerwanie końca transmisji
$EB2E - ISRSIR - przerwanie odczytu danych
$EC11 - TIM1INT - przerwanie licznika 1
$EC17 - SNDENBL - zezwolenie na zapis danych
$EEBC - NEWDEVC - wywołanie nowego urządzenia
$EF6E - POWERON - inicjowanie edytora
$F223 - TESTROM - skok do SWITROM
$FB51 - KEYDEF - tabela definicji klawiszy
$FC11 - FKDEF - tabela definicji klawiszy funkcyjnych
$FC19 - CPUIRQ - przerwanie IRQ klawiatury
$FCC4 - FSDL - przerwanie NMI programu ANTIC-a
$FCDB - CASINIT - inicjowanie magnetofonu
$FCF7 - CASOPIN - początek odczytu z magnetofonu
$FD8D - CASRDBL - odczyt bloku z magnetofonu
$FE99 - PRINIT - inicjowanie drukarki
$FF73 - CKROM1 - sprawdzenie pierwszej części ROM
$FF92 - CKROM2 - sprawdzenie drugiej części ROM
$FFA9 - GETCKS - zliczanie sumy kontrolnej
$FFD7 - CKSTAB - tabela adresów bloków pamięci ROM
$FFF8 - CHSRO2 - suma kontrolna drugiej części ROM
$FFFA - NMIVEC - wektor procedury przerwania NMI
$FFFC - RESETVEC - wektor procedury przerwania RESET
$FFFE - IRQVEC - wektor procedury przerwania IRQ
```

Dodatek B

Rejestry OS w pamięci RAM

```
  $00 - LNFLG - rejestr pomocniczy procedury RESET
  $01 - NGFLAG - rejestr pomocniczy procedury RESET
  $02 - CASINI - wektor inicjacji po odczycie z kasety
  $04 - RAMLO - wektor RAM dla testu wielkości pamięci
  $06 - TRAMSZ - rejestr tymczasowy dla testu wielkości RAM
  $08 - WARMST - znacznik gorącego startu
  $09 - BOOT? - znacznik odczytu wstępnego
  $0A - DOSVEC - wektor startowy programu dyskowego
  $0C - DOSINI - wektor inicjacji po odczycie z dyskietki
  $0E - APPMHI - najwyższy adres RAM zajęty przez program
  $10 - IRQENS - rejestr-cień IRQEN
  $11 - IRQSTAT - rejestr-cień IRQST
  $12 - RTCLOCK - zegar czasu rzeczywistego
  $30 - STATUS - status aktualnej operacji SIO
  $31 - CHKSUM - suma kontrolna dla operacji SIO
  $32 - BUFR - adres bufora danych dla SIO
  $34 - BUFEN - adres końca bufora danych dla SIO
  $38 - BUFRFL - znacznik zapełnienia bufora SIO
  $39 - RECVND - znacznik końca odczytu
  $3A - XMTDON - znacznik końca transmisji
  $3B - CHKSNT - znacznik nadania sumy kontrolnej
  $3C - NOCKSM - znacznik braku sumy kontrolnej
  $3E - GAPTYP - wskaźnik długości przerwy między blokami
  $41 - IOSNDEN - znacznik dźwięku przy transmisji
  $42 - CRITIC - znacznik krytycznych czasowo operacji I/O
  $4D - ATRACT - licznik trybu przyciągania uwagi
  $4E - ATRMSK - maska trybu przyciągania uwagi
  $4F - COLRSH - maska zmiany kolorów
  $52 - LMARGIN - lewy margines obrazu
  $53 - RMARGIN - prawy margines obrazu
  $60 - FKDEFP - wektor tabeli definicji klawiszy F1-F4
  $62 - PALNTS - wskaźnik systemu TV
  $6A - RAMTOP - liczba stron pamięci RAM
  $79 - KEYDEFP - wektor tabeli definicji klawiszy
  $8B - CKSUM - tymczasowy rejestr sumy kontrolnej
  $9E - TMPREG - tymczasowy rejestr sumy kontrolnej
  $D4 - FR0 - zerowy rejestr liczb FP
  $DA - FRE - dodatkowy rejestr liczb FP
  $E0 - FR1 - pierwszy rejestr liczb FP
  $E6 - FR2 - drugi rejestr liczb FP
  $EC - FRX - rejestr pomocniczy
  $ED - EEXP - rejestr wartości wykładnika
  $EE - NSIGN - rejestr znaku liczby
  $EF - ESIGN - rejestr znaku wykładnika
  $F0 - FCHRFLG - znacznik pierwszego znaku liczby
  $F1 - DIGRT - liczba cyfr po przecinku
  $F2 - CIX - indeks znaku w buforze
  $F3 - INBUFP - adres bufora wejściowego
  $F5 - ZTEMP1 - rejestr tymczasowy
  $F7 - ZTEMP2 - rejestr tymczasowy
  $F9 - ZTEMP3 - rejestr tymczasowy
  $FC - FLPTR - adres liczby FP
  $FE - FPTR2 - adres liczby FP
$0100 - STACK - stos mikroprocesora 6502
```

```
$0200 - DLIV - wektor przerwania programu ANTIC-a
$0202 - VPRCED - wektor przerwania portu A PIA
$0204 - VINTER - wektor przerwania portu B PIA
$0206 - VBREAK - wektor przerwania rozkazu BRK
$020A - VSERIN - wektor przerwania odczytu szeregowego
$020C - VSEROR - wektor przerwania zapisu szeregowego
$020E - VSEROC - wektor przerwania końca transmisji
$0210 - VTIMR1 - wektor przerwania licznika 1 POKEY-a
$0212 - VTIMR2 - wektor przerwania licznika 2 POKEY-a
$0214 - VTIMR4 - wektor przerwania licznika 4 POKEY-a
$0216 - VIMIRQ - główny wektor przerwania IRQ
$0218 - TIMCNT1 - pierwszy licznik systemu
$021A - TIMCNT2 - drugi licznik systemu
$021C - TIMCNT3 - trzeci licznik systemu
$021E - TIMCNT4 - czwarty licznik systemu
$0220 - TIMCNT5 - piąty licznik systemu
$0222 - VVBLKI - wektor natychmiastowego przerwania VBLK
$0224 - VVBLKD - wektor opóźnionego przerwania VBLK
$0226 - TIMVEC1 - wektor przerwania licznika TIMCNT1
$0228 - TIMVEC2 - wektor przerwania licznika TIMCNT2
$022B - SRTIMER - zegar powtarzania klawiatury
$022D - INTEMP - rejestr przejściowy procedury SETVBLV
$022F - DMACTLS - rejestr-cień DMACTL
$0230 - DLPTRS - rejestr-cień DLPTR
$0232 - SKCTLS - rejestr-cień SKCTL
$0234 - LPENHS - rejestr-cień LPENH
$0235 - LPENVS - rejestr-cień LPENV
$0236 - VBRKKEY - wektor przerwania klawisza BREAK
$0238 - VPIRQ - wektor przerwania nowego urządzenia
$0240 - DFLAG - znacznik operacji dyskowych
$0241 - DSECCNT - licznik sektorów dla operacji dyskowych
$0242 - BOOTAD - adres ładowania przy stępnym odczycie
$0244 - COLDST - znacznik zimnego startu systemu
$0246 - DSKTIM - wartość Timeout dla stacji dysków
$0248 - PDVRS - rejestr-cień PDVREG
$0249 - PINTMSK - maska przerwań nowych urządzeń
$026B - CHSPTR - wektor nieużywanego zestawu znaków
$026C - VSFLAG - znacznik przesuwu pionowego obrazu
$026D - KEYDIS - znacznik zablokowania klawiatury
$026F - GTICTLS - rejestr-cień GTIACTL
$0270 - PADDL0 - rejestr-cień POT0
$0278 - JSTICK0 - położenie joysticka 0
$0279 - JSTICK1 - położenie joysticka 1
$027A - JSTICK2 - położenie joysticka 2
$027B - JSTICK3 - położenie joysticka 3
$027C - PTRIG0 - przycisk potencjometru 0
$027D - PTRIG1 - przycisk potencjometru 1
$0284 - TRIG0S - przycisk joysticka 0, rejestr-cień TRIG0
$0285 - TRIG1S - przycisk joysticka 1, rejestr-cień TRIG1
$0286 - TRIG2S - rejestr-cień TRIG2
$0287 - TRIG3S - rejestr-cień TRIG3
$028C - NEWIOP - tymczasowy wektor skoku przerwań IRQ
$02BE - SHFLOK - znacznik klawiszy SHIFT i START
$02C0 - COLPM0S - rejestr-cień COLPM0
$02C1 - COLPM1S - rejestr-cień COLPM1
$02C2 - COLPM2S - rejestr-cień COLPM2
$02C3 - COLPM3S - rejestr-cień COLPM3
$02C4 - COLPF0S - rejestr-cień COLPF0
$02C5 - COLPF1S - rejestr-cień COLPF1
```

```
$02C6 - COLPF2S - rejestr-cień COLPF2
$02C7 - COLPF3S - rejestr-cień COLPF3
$02C8 - COLBAKS - rejestr-cień COLBAK
$02D5 - DSCTLN - długość sektora dyskowego
$02D9 - KRPDEL - czas opóźnienia powtarzania klawisza
$02DA - KEYREP - częstotliwość powtarzania klawisza
$02DC - HLPFLG - znacznik klawisza HELP
$02DD - DMASAV - rejestr do przechowywania DMACTLS
$02E4 - RAMSIZ - liczba stron pamięci RAM
$02E5 - MEMTOP - adres górnej granicy wolnej pamięci RAM
$02E7 - MEMLO - adres dolnej granicy wolnej pamięci RAM
$02EE - CBAUD - prędkość transmisji z magnetofonu
$02F0 - CRSINH - znacznik widoczności kursora
$02F1 - KEYDEL - wartość opóźnienia odczytu klawisza
$02F2 - OLDKBC - kod poprzednio naciśniętego klawisza
$02F3 - CHACT - rejestr-cień CHRCTL
$02F4 - CHBAS - rejestr-cień CHBASE
$02FC - KBCODES - rejestr-cień KBCODE
$02FF - SSFLAG - znacznik start/stop dla przesuwu obrazu
$0300 - DDEVIC - kod identyfikacyjny urządzenia
$0301 - DUNIT - numer identyfikacyjny urządzenia
$0302 - DCMND - bajt rozkazu dla urządzenia
$0304 - DBUFA - adres bufora danych
$030A - DAUX1 - rejestr pomocniczy dla operacji I/O
$030B - DAUX2 - rejestr pomocniczy dla operacji I/O
$0314 - PTIMOT - wartość Timeout dla drukarki
$0317 - TIMFLG - znacznik upłynięcia czasu Timeout
$031A - HATABS - tabela wektorów procedur obsługi
$033D - PUPBT1 - bajt kontrolny zimnego startu
$033E - PUPBT2 - bajt kontrolny zimnego startu
$033F - PUPBT3 - bajt kontrolny zimnego startu
$0340 - ICCHID - indeks wpisu urządzenia w HATABS
$0342 - ICCMD - kod rozkazu operacji I/O
$0344 - ICBUFA - adres bufora danych
$0346 - ICPUTB - adres procedury przesyłania danych
$0348 - ICBUFL - długość bufora danych
$034A - ICAX1 - rejestr pomocniczy dla operacji I/O
$03E9 - CKEY - znacznik klawisza START przy zimnym starcie
$03EA - CASSBT - znacznik odczytu z magnetofonu
$03EB - CARTCK - suma kontrolna cartridge'a
$03EC - DERRF - znacznik błędu przy otwieraniu edytora
$03F8 - BASICF - znacznik interpretera Atari Basic
$03FA - GINTLK - znacznik cartridge'a (kopia TRIG3)
$0400 - CASBUF - bufor magnetofonu
$057F - LBPR2 - prefiks bufora
$0580 - LBUFF - bufor wyjściowy operacji FP
$05E0 - PLYARG - argument przeliczania wielomianowego
$BFF0 - CART - blok informacji o cartridge'u
$BFFA - CARTRUN - adres uruchomienia cartridge'a
$BFFC - CARTINS - znacznik zainstalowania cartridge'a
$BFFD - CARTOPT - rejestr rodzaju cartridge'a
$BFFE - CARTINI - adres inicjowania cartridge'a
$D010 - TRIG0 - stan przycisku joysticka 0
$D011 - TRIG1 - stan przycisku joysticka 1
$D012 - COLPM0 - rejestr koloru gracza 0
$D013 - COLPM1 - rejestr koloru gracza 1
$D013 - TRIG3 - znacznik dołączenia cartridge'a
$D014 - COLPM2 - rejestr koloru gracza 2
$D014 - PAL - znacznik systemu TV
```

```
$D015 - COLPM3 - rejestr koloru gracza 3
$D016 - COLPF0 - rejestr koloru pola gry 0
$D017 - COLPF1 - rejestr koloru pola gry 1
$D018 - COLPF2 - rejestr koloru pola gry 2
$D019 - COLPF3 - rejestr koloru pola gry 3
$D01A - COLBAK - rejestr koloru tła
$D01B - GTIACTL - rejestr kontroli układu GTIA
$D01F - CONSOL - rejestr stanu klawiszy konsoli
$D1FF - PDVREG - rejestr wyboru nowego urządzenia
$D200 - POT0 - rejestr położenia potencjometru 0
$D205 - AUDC3 - rejestr kontroli dźwięku generatora 3
$D207 - AUDC4 - rejestr kontroli dźwięku generatora 4
$D208 - AUDCTL - rejestr kontroli generatorów dźwięku
$D209 - KBCODE - kod ostatnio naciśniętego klawisza
$D20A - SKSTRES - reset statusu złącza szeregowego
$D20B - POTGO - znacznik przetwornika analogowo-cyfrowego
$D20D - SEROUT - szeregowy rejestr wyjściowy
$D20D - SERIN - szeregowy rejestr wejściowy
$D20E - IRQST - status przerwań IRQ
$D20E - IRQEN - zezwolenia przerwań IRQ
$D20F - SKCTL - rejestr kontroli złącza szeregowego
$D20F - SKSTAT - rejestr statusu złącza szeregowego
$D300 - PORTA - port A układu PIA
$D301 - PORTB - port B układu PIA
$D302 - PACTL - rejestr kontroli portu A
$D303 - PBCTL - rejestr kontroli portu B
$D400 - DMACTL - rejestr kontroli dostępu do pamięci
$D401 - CHRCTL - rejestr kontroli wyświetlania znaków
$D402 - DLPTR - adres programu ANTIC-a
$D405 - VSCROL - znacznik pionowego przesuwu obrazu
$D409 - CHBASE - adres zestawu znaków
$D40C - LPENH - poziome położenie pióra świetlnego
$D40D - LPENV - pionowe położenie pióra świetlnego
$D40A - WSYNC - znacznik oczekiwania na synchronizację poziomą
$D40E - NMIEN - rejestr zezwoleń na przerwania NMI
$D40F - NMIST - rejestr statusu przerwań NMI
```

Dodatek C

Zmienne systemowe

```
BASICF
    0 - interpreter Atari Basic dołączony
    1 - interpreter Atari Basic odłączony

BOOT?
    1 - udany odczyt ze stacji dysków
    2 - udany odczyt z magnetofonu
    3 - udany odczyt z magnetofonu i stacji dysków

CARTINS
    0 - cartridge zainstalowany
    nie 0 - brak cartridge'a w gnieździe

CARTOPT
bit 0 - wstępny odczyt z dyskietki (0 = zabroniony)
bit 2 - 1 = inicjowanie i uruchomienie cartridge'a
```

```
        0 = tylko inicjowanie
bit 7 - rodzaj cartridge'a (1 = diagnostyczny)
pozostałe bity niewykorzystane

CASSBT
    0 - wstępny odczyt ze stacji dysków
    1 - wstępny odczyt z magnetofonu

CHRCTL/CHACT
bit 0 - tłumienie znaków (1 = znaki w inverse niewidoczne)
bit 1 - odwracanie znaków (0 = inverse nie działa)
bit 2 - odbicie znaków (1 = lustrzane odbicie znaków w pionie)
bity 3-7 - niewykorzystane

CKEY
    0 - klawisz START nie był naciśnięty
    0 - klawisz START był naciśnięty przy włączaniu zasilania

CONSOL
bit 0 - klawisz START (0 = wciśnięty)
bit 1 - klawisz SELECT (0 = wciśnięty)
bit 2 - klawisz OPTION (0 = wciśnięty)

CRITIC
    0 - brak ograniczeń czasowych
    nie 0 - krytyczna czasowo operacja I/O

CRSINH
    0 - kursor widoczny
    nie 0 - kursor niewidoczny

DMACTL/DMACTLS
bity 0-1 - szerokość obrazu na ekranie
           00 = brak obrazu
           01 = obraz wąski (192 punkty)
           10 = obraz normalny (320 punktów)
           11 = obraz szeroki (394 punkty)
bit 2 - DMA dla pocisków (1 = włączony)
bit 3 - DMA dla graczy (1 = włączony)
bit 4 - rozdzielczość P/MG (1 = jednowierszowa)
bit 5 - DMA dla programu ANTIC-a (1 = włączony)
bity 6-7 - niewykorzystane

GINTLK
    0 - brak cartridge'a w gnieździe
    1 - cartridge zainstalowany

GTIACTL/GTICTLS
bity 0-3 - priorytet graczy i pocisków
bit 4 - łączenie pocisków w 5 gracza (1 = włączona)
bit 5 - gracze wielokolorowi (1 = włączona)
bity 6-7 - dodatkowe tryby graficzne GTIA:
           00 - GRAPHICS 8
           01 - GRAPHICS 9
           10 - GRAPHICS 10
           11 - GRAPHICS 11

HLPFLG
    $11 - wciśnięty klawisz HELP
```

```
    $51 - wciśnięte klawisze HELP i SHIFT
    $91 - wciśnięte klawisze HELP i CONTROL
    $D1 - wciśnięte klawisze HELP, SHIFT i CONTROL

IRQEN/IRQENS
zezwolenia przerwań IRQ - 0 = zabronione
bit 0 - przerwanie TIMERa 1
bit 1 - przerwanie TIMERa 2
bit 2 - przerwanie TIMERa 4
bit 3 - przerwanie końca transmisji
bit 4 - przerwanie zapisu przez złącze szeregowe
bit 5 - przerwanie odczytu przez złącze szeregowe
bit 6 - przerwanie klawiatury
bit 7 - przerwanie klawisza BREAK

IRQST/IRQSTAT
status przerwań IRQ - 0 = wystąpiło
bit 0 - przerwanie TIMERa 1
bit 1 - przerwanie TIMERa 2
bit 2 - przerwanie TIMERa 4
bit 3 - przerwanie końca transmisji
bit 4 - przerwanie zapisu przez złącze szeregowe
bit 5 - przerwanie odczytu przez złącze szeregowe
bit 6 - przerwanie klawiatury
bit 7 - przerwanie klawisza BREAK

NMIEN
zezwolenie przerwań NMI - 0 = zabronione
bity 0-5 - niewykorzystane
bit 6 - przerwanie synchronizacji (VBLKI)
bit 7 - przerwanie programu ANTIC-a (DLI)

NMIST
status przerwań NMI - 1 = wystąpiło
bity 0-4 - niewykorzystane
bit 5 - przerwanie klawisza RESET
bit 6 - przerwanie synchronizacji (VBLKI)
bit 7 - przerwanie programu ANTIC-a (DLI)

SHFLOK
    $00 - małe litery (naciśnięty CAPS)
    $40 - duże litery (stan normalny)
    $80 - znaki z CONTROL

SKCTL/SKCTLS
bit 0 - razem z bitem 1 resetuje POKEY
bit 1 - obsługa klawiatury
bit 2 - częstość przetwarzania A/C (0 = 20 ms, 1 = 128 µs)
bit 3 - przesyłanie dwutonowe (1 = włączone)
bity 4-6 - sterowanie szybkością transmisji
bit 7 - nadanie sygnału SPACE (1 = włączone)

SKSTAT
bit 0 - niewykorzystany (= 1)
bit 1 - przesyłanie danych (0 = trwa)
bit 2 - dowolny klawisz (0 = naciśnięty)
bit 3 - klawisz SHIFT (0 = naciśnięty)
bit 4 - kopia rejestru wejścia szeregowego
bit 5 - bufor klawiatury (0 = przepełniony)
```

```
bit 6 - bufor wejścia szeregowego (0 = przepełniony)
bit 7 - Framing Error (0 = wystąpił)

SSFLAG
    $00 - dane są wprowadzane na ekran
    $FF - zatrzymanie wyprowadzania danych
```

Dodatek D

Słownik terminów informatycznych

ANTIC

AlphaNumeric Television Interface Controller - drugi, dodatkowy mikroprocesor, którego zasadniczym zadaniem jest tworzenie obrazu. Układ specjalnie zaprojektowany dla komputerów Atari.

ASCII

American Standard Code of Information Intercharge - amerykański, standardowy kod wymiany informacji, kod przypisujący liczbom od 0 do 127 znaczenie liter, liczb i znaków kontrolnych, powszechnie używany w komputerach. Każda firma stosuje jednak nieco zmodyfikowany kod, np. w Atari jest używany ATASCII (ATari ASCII).

BCD

Binary Coded Decimal - liczba dziesiętna kodowana dwójkowo, kod zapisu liczb dziesiętnych, w którym każdej cyfrze odpowiadają cztery bity. W ten sposób w jednym bajcie można zapisać dwie cyfry dziesiętne. Maksymalna wartość półbajtu w kodzie BCD wynosi 9. Procesor 6502 po rozkazie SED pracuje w trybie dziesiętnym, czyli na liczbach w kodzie BCD.

CIO

Central Input/Output - zespół procedur obsługujących komunikację komputera z urządzeniami zewnętrznymi.

DCB

Device Control Block - blok kontroli urządzeń, obszar pamięci RAM od $0300 do $030B wykorzystywany przez procedury SIO do operacji wejścia/wyjścia.

DL

Display List - program ANTIC-a, program w kodzie maszynowym ANTIC-a wskazujący mu sposób generowania obrazu. Program ANTIC-a może wywoływać przerwania niemaskowalne zwane DLI (Display List Interrupt).

DMA

Direct Memory Access - bezpośredni dostęp do pamięci, sposób dostępu do pamięci komputera z pominięciem pośrednictwa procesora. Oszczędza to czas pracy procesora i zwiększa szybkość pracy systemu. W Atari DMA jest wykorzystywany przez ANTIC (procesor obrazowy).

FP

Floating Point - liczby, operacje i procedury na liczbach rzeczywistych czyli zmiennoprzecinkowych.

GTIA

Graphics Television Interface Adaptor - specjalizowany układ scalony służący do tworzenia kolorów i obsługi grafiki graczy i pocisków (Player/Missile Graphics).

I/O

Input/Output - wejście/wyjście, ogólna nazwa operacji służących do komunikacji komputera z urządzeniami zewnętrznymi.

IOCB

Input/Output Control Block - blok kontroli I/O, 16-bajtowy obszar pamięci RAM wykorzystywany przez procedury CIO do operacji wejścia/wyjścia. Istnieje IOCB strony zerowej i osiem IOCB w obszarze od $0340 do $3BF.

IRQ

Interrupt Request - żądanie przerwania, nazwą tą określa się wszystkie przerwania maskowalne, to znaczy takie, które mogą nie zostać przyjęte do realizacji przez procesor.

LSB

Least Significant Byte - mniej znaczący bajt, bajt adresu lub danej zawierający dwie bardziej

znaczące cyfry. MSB=INT(ADR/$0100). Przy adresowaniu wskazuje numer strony pamięci.

NMI

Non Maskable Interrupt - przerwanie niemaskowalne czyli takie, które musi zostać wykonane przez procesor (nie może być zignorowane).

nowe urządzenie

New Device - nazwa stosowana w systemie operacyjnym Atari dla określenia urządzeń zewnętrznym przyłączanych do szyny równoległej, które nie istniały jeszcze w czasie projektowania systemu.

OS

Operating System - system operacyjny, zestw procedur zapisanych przeważnie w pamięci ROM, które sterują pracą komputera i jego współpracą z urządzeniami zewnętrznymi.

PIA

Peripheral Interface Adaptor - standardowy układ I/O typu 6520. Jego zadaniem jest w Atari obsługa portów joysticków i zarządzanie pamięcią komputera.

POKEY

Potentiometr & Keyboard - specjalizowany układ scalony, którego zadaniem jest obsługa klawiatury, potencjometrów oraz komunikacji komputera z urządzeniami zewnętrznymi poprzez złącze szeregowe.

port

Rejestr służący do komunikacji komputera z urządzeniami peryferyjnymi, zwykle jest to rejestr specjalnego układu scalonego znajdujący się w przestrzeni adresowej procesora. Zwany jest także bramą.

rejestr

Zespół komórek (w Atari 1-6) pamięci RAM służących do przechowywania różnych wartości niezbędnych do pracy OS lub programu. Podstawowymi rodzajami rejestrów są wektory i znaczniki (wskaźniki).

rejestr-cień

Specjalizowane układy scalone komputerów Atari mają własne rejestry, które znajdują się w obszarze adresowym procesora. Większość z nich posiada kopie w normalnych rejestrach RAM. tzw. rejestry-cienie (shadow register). Zawartość rejestrów-cieni jest przepisywana do rejestrów sprzętowych lub odwrotnie podczas przerwania VBLK.

SIO

Serial Input/Output - zespół procedur obsługujących komunikację z urządzeniami zewnętrznymi poprzez złącze szeregowe.

strona

Część pamięci komputera o wielkości 256 bajtów. Starszy bajt adresu wskazuje zawsze numer strony, a młodszy - komórkę na stronie.

VBLK

Vertical Blank - synchronizacja pionowa, okres wygaszenia strumienia elektronów tworzących obraz i przesunięcia go z prawego, dolnego rogu ekranu do lewego, górnego. Podczas VBLK wywoływane jest przerwanie VBLKI (VBLK Interrupt).

wektor

Rejestr zawierający adres procedury lub danych (wskazujący adres). Wektor dwubajtowy zawiera pełny adres, a wektor jednobajtowy tylko numer strony.

wskaźnik

to samo co znacznik.

znacznik

Rejestr, którego zawartość sygnalizuje stan jakiegoś elementu systemu lub wariant operacji. Znacznik może być traktowany jako całość, ale często poszczególne bity znacznika mają odrębne znaczenie.

Dodatek E

Tabela przeliczeń DEC-BIN-HEX

Poniższa tabela może służyć do szybkiej zamiany liczb zapisanych w systemach: szesnastkowym (HEX), dziesiętnym (DEC) i dwójkowym (BIN). Dodatkowo dla ułatwienia orientacji w stosowanym przez procesor 6502 systemie adresowania podana jest wartość dziesiętna pomnożona przez 256 (strona).

HEX	DEC	strona	BIN	HEX	DEC	strona	BIN
00	0	0	00000000	80	128	32768	10000000
01	1	256	00000001	81	129	33024	10000001
02	2	512	00000010	82	130	33280	10000010
03	3	768	00000011	83	131	33536	10000011
04	4	1024	00000100	84	132	33792	10000100
05	5	1280	00000101	85	133	34048	10000101
06	6	1536	00000110	86	134	34304	10000110
07	7	1792	00000111	87	135	34560	10000111
08	8	2048	00001000	88	136	34816	10001000
09	9	2304	00001001	89	137	35072	10001001
0A	10	2560	00001010	8A	138	35328	10001010
0B	11	2816	00001011	8B	139	35584	10001011
0C	12	3072	00001100	8C	140	35840	10001100
0D	13	3328	00001101	8D	141	36096	10001101
0E	14	3584	00001110	8E	142	36352	10001110
0F	15	3840	00001111	8F	143	36608	10001111
10	16	4096	00010000	90	144	36864	10010000
11	17	4352	00010001	91	145	37120	10010001
12	18	4608	00010010	92	146	37376	10010010
13	19	4864	00010011	93	147	37632	10010011
14	20	5120	00010100	94	148	37888	10010100
15	21	5376	00010101	95	149	38144	10010101
16	22	5632	00010110	96	150	38400	10010110
17	23	5888	00010111	97	151	38656	10010111
18	24	6144	00011000	98	152	38912	10011000
19	25	6400	00011001	99	153	39168	10011001
1A	26	6656	00011010	9A	154	39424	10011010
1B	27	6912	00011011	9B	155	39680	10011011
1C	28	7168	00011100	9C	156	39936	10011100
1D	29	7424	00011101	9D	157	40192	10011101
1E	30	7680	00011110	9E	158	40448	10011110
1F	31	7936	00011111	9F	159	40704	10011111
20	32	8192	00100000	A0	160	40960	10100000
21	33	8448	00100001	A1	161	41216	10100001
22	34	8704	00100010	A2	162	41472	10100010
23	35	8960	00100011	A3	163	41728	10100011
24	36	9216	00100100	A4	164	41984	10100100
25	37	9472	00100101	A5	165	42240	10100101
26	38	9728	00100110	A6	166	42496	10100110
27	39	9984	00100111	A7	167	42752	10100111
28	40	10240	00101000	A8	168	43008	10101000
29	41	10496	00101001	A9	169	43264	10101001
2A	42	10752	00101010	AA	170	43520	10101010
2B	43	11008	00101011	AB	171	43776	10101011
2C	44	11264	00101100	AC	172	44032	10101100
2D	45	11520	00101101	AD	173	44289	10101101

2E	46	11776	00101110	AE	174	44544	10101110
2F	47	12032	00101111	AF	175	44800	10101111
30	48	12288	00110000	B0	176	45056	10110000
31	49	12544	00110001	B1	177	45312	10110001
32	50	12800	00110010	B2	178	45568	10110010
33	51	13056	00110011	B3	179	45824	10110011
34	52	13312	00110100	B4	180	46080	10110100
35	53	13568	00110101	B5	181	46336	10110101
36	54	13824	00110110	B6	182	46592	10110110
37	55	14080	00110111	B7	183	46848	10110111
38	56	14336	00111000	B8	184	47104	10111000
39	57	14592	00111001	B9	185	47360	10111001
3A	58	14848	00111010	BA	186	47616	10111010
3B	59	15104	00111011	BB	187	47872	10111011
3C	60	15360	00111100	BC	188	48128	10111100
3D	61	15616	00111101	BD	189	48384	10111101
3E	62	15872	00111110	BE	190	48640	10111110
3F	63	16128	00111111	BF	191	48896	10111111
40	64	16384	01000000	C0	192	49152	11000000
41	65	16640	01000001	C1	193	49408	11000001
42	66	16896	01000010	C2	194	49664	11000010
43	67	17152	01000011	C3	195	49920	11000011
44	68	17408	01000100	C4	196	50176	11000100
45	69	17664	01000101	C5	197	50432	11000101
46	70	17920	01000110	C6	198	50688	11000110
47	71	18176	01000111	C7	199	50944	11000111
48	72	18432	01001000	C8	200	51200	11001000
49	73	18688	01001001	C9	201	51456	11001001
4A	74	18944	01001010	CA	202	51712	11001010
4B	75	19200	01001011	CB	203	51968	11001011
4C	76	19456	01001100	CC	204	52224	11001100
4D	77	19712	01001101	CD	205	52480	11001101
4E	78	19968	01001110	CE	206	52736	11001110
4F	79	20224	01001111	CF	207	52992	11001111
50	80	20480	01010000	D0	208	53248	11010000
51	81	20736	01010001	D1	209	53504	11010001
52	82	20992	01010010	D2	210	53760	11010010
53	83	21248	01010011	D3	211	54016	11010011
54	84	21504	01010100	D4	212	54272	11010100
55	85	21760	01010101	D5	213	54528	11010101
56	86	22016	01010110	D6	214	54784	11010110
57	87	22272	01010111	D7	215	55040	11010111
58	88	22528	01011000	D8	216	55296	11011000
59	89	22784	01011001	D9	217	55552	11011001
5A	90	23040	01011010	DA	218	55808	11011010
5B	91	23296	01011011	DB	219	56064	11011011
5C	92	23552	01011100	DC	220	56320	11011100
5D	93	23808	01011101	DD	221	56576	11011101
5E	94	24064	01011110	DE	222	56832	11011110
5F	95	24320	01011111	DF	223	57088	11011111
60	96	24576	01100000	E0	224	57344	11100000
61	97	24832	01100001	E1	225	57600	11100001
62	98	25088	01100010	E2	226	57856	11100010
63	99	25344	01100011	E3	227	58112	11100011
64	100	25600	01100100	E4	228	58368	11100100
65	101	25856	01100101	E5	229	58624	11100101
66	102	26112	01100110	E6	230	58880	11100110
67	103	26368	01100111	E7	231	59136	11100111
68	104	26624	01101000	E8	232	59392	11101000

69	105	26880	01101001	E9	233	59648	11101001
6A	106	27136	01101010	EA	234	59904	11101010
6B	107	27392	01101011	EB	235	60160	11101011
6C	108	27648	01101100	EC	236	60416	11101100
6D	109	27904	01101101	ED	237	60672	11101101
6E	110	28160	01101110	EE	238	60928	11101110
6F	111	28416	01101111	EF	239	61184	11101111
70	112	28672	01110000	F0	240	61440	11110000
71	113	28928	01110001	F1	241	61696	11110001
72	114	29184	01110010	F2	242	61952	11110010
73	115	29440	01110011	F3	243	62208	11110011
74	116	29696	01110100	F4	244	62464	11110100
75	117	29952	01110101	F5	245	62720	11110101
76	118	30208	01110110	F6	246	62976	11110110
77	119	30464	01110111	F7	247	63232	11110111
78	120	30720	01111000	F8	248	63488	11111000
79	121	30976	01111001	F9	249	63744	11111001
7A	122	31232	01111010	FA	250	64000	11111010
7B	123	31488	01111011	FB	251	64256	11111011
7C	124	31744	01111100	FC	252	64512	11111100
7D	125	32000	01111101	FD	253	64768	11111101
7E	126	32256	01111110	FE	254	65024	11111110
7F	127	32512	01111111	FF	255	65280	11111111

Dodatek F

Tabela różnic asemblerów

Asemblery dostępne na Atari XL/XE różnią się nieco między sobą stosowanymi słowami kluczowymi. Poniższa tabela zawiera różnice występujące w kilku najpopularniejszych asemblerach: Macroassembler 65 (MAC/65), Atari Assembler/Editor (ASM/EDIT), Atari Macroassembler (AMAC) i Synapse Assembler (SYNASM).

MAC/65	ASM/EDIT	AMAC	SYNASM
*=	*=	ORG	.OR
*	*	*O	*
=	=	= lub EQU	.EQ
.BYTE	.BYTE	DB	.AT
.SBYTE	.BYTE	DC	.HS
.DBYTE	.BYTE	brak	.AS
.WORD	.WORD	DW	.DA
=+	*=*+	DS	.BS

Dodatek G

Bibliografia

1. Chadwick Ian: Mapping the Atari, COMPUTE! Books, USA, 1985.
2. Eichler Lutz, Grohmann Bernd: Atari 600XL/800XL Intern, Data Becker, Dusseldorf, 1984.
3. Praca zbiorowa: De Re Atari. A guide to effective programing 400/800 Home Computers, USA, 1981.

4. Ruszczyc Jan: Asembler 6502, SOETO, Warszawa, 1987.
5. Zientara Wojciech: PEEK-POKE 2 (opracowanie), B.U.K. "GLAD", Warszawa, 1987.
6. Zientara Wojciech: Mapa pamięci Atari XL/XE. Procedury wejścia/wyjścia, SOETO, Warszawa, 1988.
7. Zientara Wojciech: Mapa pamięci Atari XL/XE. Dyskowe systemy operacyjne, SOETO, Warszawa, 1988.
8. Zientara Wojciech: Mapa pamięci Atari XL/XE. Procedury interpretera Basica, SOETO, Warszawa, 1988.