ISSN 0867-390X

# MOJE Atari

CENA 8000 ZŁ

MAGAZYN UŻYTKOWNIKÓW KOMPUTERÓW ATARI ● MARZEC/KWIECIEŃ '91 (2/91)

4

HARDCOPY 1029

TURBO DO MAGNETOFONU ATARI

PORTFOLIO EMULATORY PC

LUCJAN DANIEL WENCEL

— rozmowa z człowiekiem Atari

# Drogi Czytelniku!

*Gdy piszę te słowa, jest połowa marca i za oknem zaczęły się ujawniać pierwsze ślady wiosny. Czytając je będziesz mógł stwierdzić, ile czasu trwa drukowanie pisma. Nie jest to jednak pełny cykl wydawniczy, ponieważ artykuł wstępny, który właśnie czytasz, jest oddawany do drukarni tuż przed rozpoczęciem druku. A przecież trzeba przedtem przygotować wszystkie materiały. W rzeczywistości pełny cykl wydawniczy (od zamknięcia numeru do ukazania się go w kioskach) trwa prawie trzy miesiące. Na przykład, numer „Mojego Atari", który trzymasz teraz w ręku, został zamknięty w pierwszych dniach stycznia, a według harmonogramu powinien ukazać się w kioskach 30 marca.*

*Ten długi wstęp powinien uzmysłowić Ci, że niemożliwa jest natychmiastowa reakcja na wydarzenia zachodzące na rynku komputerowym. To, co jest bowiem nowością i ostatnim krzykiem mody w chwili pisania artykułu przez autora, często jest już powszechne i dobrze znane, gdy kupujesz nasz magazyn. Tak samo przedstawia się sprawa ewentualnych poprawek i uzupełnień. Gdy są one przygotowane natychmiast po otrzymaniu kolejnego numeru z drukarni, to czytelnik musi na nie czekać cztery miesiące (ponieważ „Moje Atari" jest dwumiesięcznikiem). Jeżeli konieczność napisania uzupełnienia wynika z listów do redakcji, to „czas redakcji" przedłuża się o dalsze kilka tygodni, a więc w efekcie ujrzysz skutek dopiero po sześciu miesiącach.*

*Piszę to wszystko po to, abyś zrozumiał, że wydawanie czasopisma nie jest szybkie i w obecnych warunkach nie da się go przyspieszyć. Nie ma więc żadnego sensu zasypywanie redakcji listami i telefonami żądającymi wydrukowania czegoś natychmiast. W naszym wypadku „natychmiast" oznacza bowiem „za cztery miesiące". Jest to przykre, ale na razie nic nie możemy na to poradzić. Być może uda nam się skrócić cykl wydawniczy przez choćby częściowe wprowadzenie składu komputerowego lub inne zmiany w procesie wydawniczym. Związane jest to jednak z ogólnym tempem przemian w Polsce (np. z prywatyzacją drukarni) i z pewnością nie będzie szybkie. Zapewniam, że zdajemy sobie sprawę z niedogodności, jakie są Twoim udziałem i będziemy się starać je zmniejszać. A tymczasem zapraszam do lektury.*

*Wojciech Zientara*


MOJE Atari

MAGAZYN UŻYTKOWNIKÓW KOMPUTERÓW ATARI

Redaktor naczelny:
Wojciech Zientara
Sekretarz redakcji:
Tomasz Wiśniewski
Opracowanie graficzne:
Wanda Roszkowska
Redaktor techniczny:
Maria Radzimińska
Zdjęcia:
Leopold Dzikowski
Redakcja:
00-687 Warszawa, ul. Wspólna 61,
tel. 21-12-05
Wydawca:
Spółdzielnia "Bajtek"
00-687 Warszawa, ul. Wspólna 61,
tel. 21-12-05
Skład i druk:
Prasowe Zakłady Graficzne
Ciechanów, ul. Sienkiewicza 51
Fotoskład:
Grażyna Kurzątkowska
Montaż offsetowy:
Grażyna Ostaszewska
Korekta:
Maria Krajewska
Teresa Rutkowska
Nr zlecenia: 86430
Nakład 105 tys. egz.
Redakcja zastrzega sobie prawo skracania i adiustacji nadesłanych artykułów. Materiałów nie zamówionych nie zwracamy.
Za treść ogłoszeń redakcja nie odpowiada.


# co w numerze?

ATARI COMPUTERS


## PROGRAMOWANIE

**Action! (6) . . . . . 15**
Opis działania kompilatora Action!, prezentacja składni tego języka oraz zasady tłumaczenia programów z Basica na Action!.

**Tylko dla zaawansowanych (1) . . . 11**
Jak można najlepiej wykorzystać możliwości „małego" Atari? W pierwszym odcinku opisujemy zastosowanie przerwań synchronizacji pionowej.

**Operacje pierwotne w Kyan Pascalu . 12**
Chcesz wzbogacić swoją bibliotekę procedur Pascala? Skorzystaj z publikowanych tu procedur realizujących operacje wejścia/wyjścia.

**Kruczki i sztuczki . . . . . 21**
Kilka tricków, które ułatwią pisanie programów i pracę na komputerze.

## GRY

**Trening na klawiaturze . . . . . 6**
Prosta gra, która wymaga jednak błyskawicznego refleksu i doskonałej znajomości klawiatury komputera.

**Mercenary — The Second City . . . 20**
Mapa drugiego miasta z gry „Mercenary".

**Nowe i stare . . . . . 25**
Opisy gier „Hawkquest", „Trailblazer", „Colony" i „Druid".

## PROGRAMY UŻYTKOWE

**Hardcopy 1029 . . . . . 14**
Program, który umożliwia skopiowanie zawartości ekranu na papier przy pomocy popularnej drukarki Atari 1029.

**Suplement do Chars Designer . . . 14**
Poprawka programu z „Mojego Atari 1" dla użytkowników magnetofonów.

## SPRZĘT

**Turbo do magnetofonu . . . . . 7**
Schemat przeróbki magnetofonu XC12, która zwiększa prędkość transmisji do 5000 bodów.

**Rozszerzenie pamięci do 256 KB . . 7**
Test komputera Atari 260 XE, czyli 65 XE z pamięcią RAM powiększoną do 256 KB (rozszerzenie wykonane przez firmę TOMS).

## SZKÓŁKA

**Jak kodować? . . . . . 10**
Pisana przez nas gra zaczyna przyoblekać się w kształt — czyli, w jaki sposób przełożyć algorytm na język programowania.

## INSTRUKCJE

**Broadsides . . . . . 21**
Kolejny odcinek „Vademecum Atari", czyli skrócona instrukcja gry „Salwy burtowe".

## EDUKACJA

**Wykresy funkcji . . . . . 24**
Program rysujący wykresy funkcji jednej zmiennej — może być bardzo przydatny zarówno w szkole, jak i podczas nauki w domu.

## DZIAŁ ST

**Emulatory PC . . . . . 28**
Królujący na całym świecie standard IBM może być również dostępny dla Ciebie. Przeczytaj i wybierz najbardziej odpowiadający ci sposób uzyskania takiego efektu.

## INNE

**Nie tylko Atari . . . . . 8**
Wywiad z właścicielem Logical Design Works i PZ „Karen" — Lucjanem Danielem Wenclem.

**Atari Portfolio . . . . . 30**
Jeden z najmniejszych na świecie komputerów zgodnych z IBM PC produkowany przez firmę Atari.

**Recenzje . . . . . 22**
Podręcznik programowania Atari XL/XE dla stawiających pierwsze kroki w tej dziedzinie.

**Ocena — Matematyka . . . . . 19**

**Jak wygląda prawdziwy program? . 23**

**Wieści . . . . . 3**

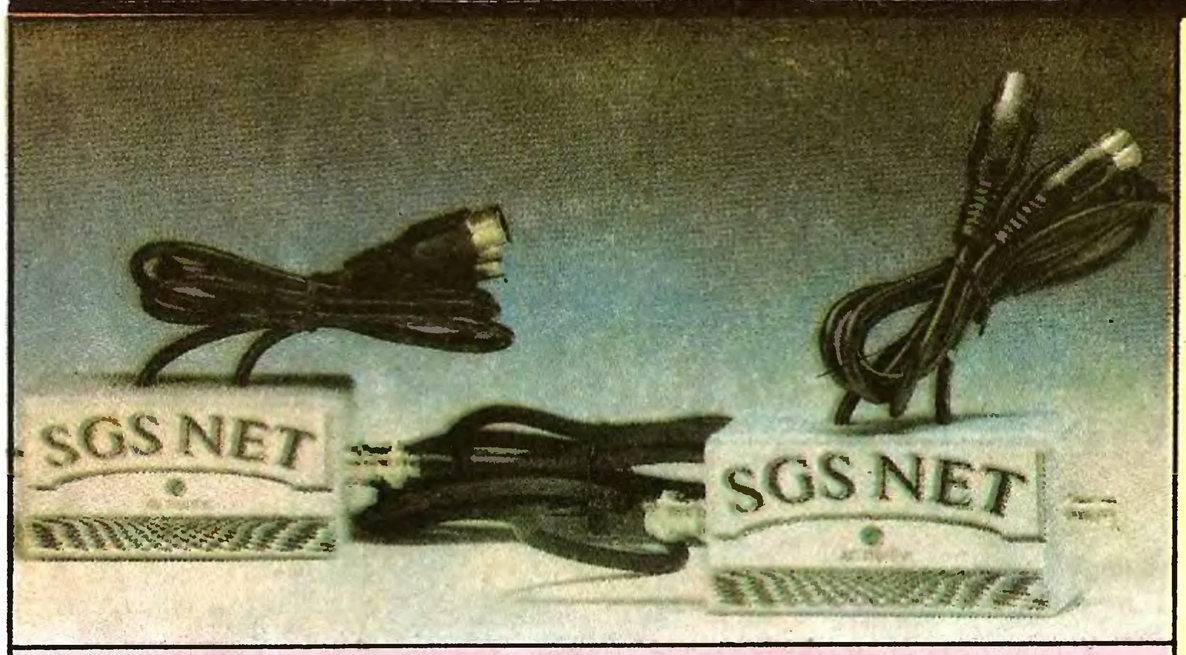
**Listy . . . . . 4**

**Errata . . . . . 23**

**Edytor Basica . . . . . 5**

**Wpisywanie znaków specjalnych . . 5**


**UWAGA!** W naszej redakcji (Warszawa, ul. Wspólna 61, wtorek–czwartek, godz. 10–14) można nabyć dyskietkę zawierającą wszystkie programy, których wydruki są opublikowane w tym numerze. Cena dyskietki wynosi 15000 zł. Niestety, z przyczyn technicznych nie rozpowszechniamy tych programów na kasetach.




# Sieć lokalna ST

Port MIDI w Atari St może być wykorzystany jako złącze komunikacyjne do pracy w sieci lokalnej. Amerykańska firma Paradise Computer Systems opracowała system umożliwiający połączenie do 32 komputerów. Komunikacja z innym ST jest możliwa dzięki urządzeniu o nazwie SGS Net, które przyłącza się do portu MIDI-In i MIDI-Out. Każdy komputer musi mieć oczywiście własny SGS Net. Przesyłanie danych odbywa się z szybkością 31,5 KBodów.

Podstawową korzyścią, jaką daje połączenie kilku Atari w sieć, jest oszczędność wydatków na drogie urządzenia peryferyjne. Najważniejsze dane są zgromadzone i przetwarzane w komputerze centralnym, który jest wyposażony w szybki twardy dysk o dużej pojemności, nawet do kilkuset megabajtów, oraz ma zainstalowany RAM-dysk. Komputer ten nazywa się potocznie *file serverem*. Użytkownicy nie muszą przechowywać własnych kopii programów i tworzyć indywidualnych banków danych. Wszystkie niezbędne informacje są udostępniane z file servera.

Podobnie dzieje się z innymi urządzeniami. Dla użytkowników sieci wystarczy jedna drukarka laserowa i jeden skaner. Terminale użytkowników mogą być bardzo prostymi, a więc tanimi komputerami. Jeśli programy systemowe umieścimy w module ROM, to terminal nie potrzebuje nawet stacji dysków elastycznych.

Praca całego systemu odbywa się pod kontrolą specjalnego oprogramowania, które dba o odpowiednie zabezpieczenie zbiorów i minimalizuje ryzyko wystąpienia błędów. Zgłoszenie się do pracy w sieci umożliwia program NODE.PRG, który może być zapisany na dyskietce lub module ROM.

# SPORT DLA LENIWYCH

Komputer jest wielkim dobrodziejstwem dla leniwych. Dzięki niemu każdy może wystartować we współzawodnictwie o tytuł światowego czempiona. Trudno znaleźć dyscyplinę sportu, której nie można uprawiać w towarzystwie komputera. Najpopularniejsze komputerowe symulacje to piłka nożna, lekkoatletyka, walki wschodnie, tenis, wyścigi samochodowe. Joystick umożliwia nawet poznanie rozgrywek angielskiej arystokracji. Nie musimy wcale wyjeżdżać na Wyspy Brytyjskie w poszukiwaniu pól golfowych. Wystarczy uruchomić program „Tournament Golf".

My jednak, w trosce o dobre samopoczucie i zdrowie miłośników gier, proponujemy przynajmniej odrobinę prawdziwego wysiłku na świeżym powietrzu.

# Programatory EPROM i GAL

Współczesne urządzenia elektroniczne, układy testujące, pomiarowe, kontrolujące przebieg procesów produkcyjnych i sterujące nowoczesnymi maszynami w niczym nie przypominają swoich przodków sprzed kilkunastu lat. Do wielu z tych układów pasuje bardziej określenie specjalizowany komputer.

Realizacja złożonych funkcji nie jest możliwa bez opracowania odpowiedniego programu, który oczywiście powinien być gdzieś zapamiętany. Stałe programy są zwykle zapisywane w ROM, który najczęściej jest zbudowany na bazie pamięci EPROM. Pod obudową urządzenia kryją się również układy logiczne, które sterują jego wewnętrzną pracą, odpowiednio włączają i wyłączają poszczególne podzespoły. Przełączanie obwodów, przez które płyną prądy o większej mocy, realizuje się przy pomocy styczników i przekaźników. Tam gdzie w grę wchodzą bardzo niewielkie prądy, tak jak we wnętrzu zwykłego komputera, można zastosować elektroniczny odpowiednik styczników: bramki TTL lub CMOS. Ale najnowocześniejszym rozwiązaniem są układy GAL, które można wielokrotnie programować. Układy logiczne typu GAL zastosowano między innymi w wielu urządzeniach rozszerzających możliwości Atari ST, np. układ podwyższający rozdzielczość Autoswitch-Overscan i emulator AT-Speed.

Do zaprogramowania EPROM lub GAL potrzebny jest specjalny programator. W takie urządzenie może zaopatrzyć się każdy użytkownik Atari ST. Dostarcza je firma Maxon Computer GmbH z RFN. Do programowania typowych EPROM oraz niektórych EEPROM i ROM służy Junior Prommer. Programator układów logicznych nosi nazwę MAXON GAL Prommer MGP 16/20. Posługując się MGP 16/20 można zaprogramować układy GAL 16V8 i 20V8. Instrukcja obsługi wyjaśnia zasadę, według której programuje się dla GAL-u podstawowe elementy układu logicznego — bramki NOR, NAND i NOT.

Oba programatory przyłącza się do portu joysticka w ST. Stąd też pochodzi niezbędne napięcie zasilające +5 V. Wraz z urządzeniami dostarczane jest odpowiednie oprogramowanie, które umożliwia przeglądanie i redagowanie zawartości EPROM lub GAL.

Programatory ułatwią pracę każdemu majsterkowiczowi, który zamierza opracować własne urządzenia elektroniczne. Inwestycja zwraca się bardzo szybko. MGP 16/20 kosztuje w RFN 195 DM, a Junior Prommer w wersji do ST 229 DM.

# KRADZIEŻ 75 TWARDYCH DYSKÓW

W Regensburgu w RFN nieznani sprawcy skradli 38 kartonów zawierających twarde dyski typu Megafile 30 i Megafile 60. Właściciel, towarzystwo ubezpieczeniowe i policja zwrócili się o pomoc do czytelników miesięcznika ST Computer. Nie wyklucza się, że skradzione przedmioty będą oferowane prywatnym odbiorcom. Dotychczas udało się odzyskać tylko niewielką część łupu. Za udzielenie informacji, które pomogą odnaleźć skradzione dyski i ująć przestępców przyznano nagrodę 2.000 DM.

# MEGA ST NA MANHATTANIE

„**MAN**nche Leute **HATTEN** einmal einen ganz normalen ST" — tak firma Tetra Computersysteme GmbH reklamuje swój komputer, Mega ST w obudowie typu *tower*. Hasło „niektórzy ludzie mieli kiedyś całkiem normalne ST" zachęca klienta do porzucenia klasycznych rozwiązań firmy Atari. Przyjrzyjmy się uważniej pierwszym słowom sloganu reklamowego. Wieża Mega ST nieprzypadkowo nosi sympatyczną nazwę Manhatten.

Architektura komputera jest każdorazowo dopasowywana do wymagań klienta. Obudowa ma wymiary 45,6 × 19 × 62 cm. W jej wnętrzu można pomieścić znacznie więcej podzespołów. Manhatten w jednej z wielu możliwych konfiguracji może zawierać: mikroprocesor 16 MHz z pamięcią podręczną (cache memory) 16 KB, pamięć RAM 4 MB, stację dysków elastycznych 3,5" o pojemności 1,44 MB/720 KB firmy TEAC, twardy dysk standardu SCSI 84 MB, interfejs równoległy, przełącznik Turbo, gniazdo do przyłączenia kolorowego monitora, zasilacz, mysz i klawiaturę z 5-metrowym kablem spiralnym.

*„Wieści" przygotował Janusz Jarmoch*

# *Na listy Czytelników odpowiada Wojciech Zientara*

***Mam zamiar kupić modem XM-301P. Mój problem polega na tym, że z jednej strony takie urządzenie może się przydać, a z drugiej w Polsce publiczne sieci komputerowe nie istnieją.***

***Rafał Mroczkowski***
***Krosno***

Niewątpliwie publicznie dostępne sieci komputerowe będą się w Polsce rozwijały. Jest to jednak kwestia kilku lat, choć nieśmiałe próby są już podejmowane. Wśród wielu zalet modem XM-301P ma jedną poważną wadę, która nie została wyraźnie wyeksponowana w artykule zamieszczonym w „Moim Atari" 1. Otóż pracuje on w standardzie amerykańskim (Bell), więc komunikacja z siecią publiczną będzie możliwa tylko wtedy, gdy będzie ona automatycznie rozpoznawać rodzaj standardu (w Europie stosuje się standard CCITT).

***W numerze 5–6/90 „Bajtka" przeczytałem, że jest możliwe skopiowanie programem „FCopy" programów z kasety na dyskietkę. Jak to zrealizować?***

***Adrian Mroczko***
***Lasowice Wielkie***

„FCopy" umożliwia skopiowanie z kasety na dyskietkę tylko dwuczęściowych programów „z wykrzyknikiem". W tym celu należy wczytać „FCopy" przy pomocy DOS-u, ustawić taśmę na początku drugiej części programu (za loaderem z wykrzyknikiem), wczytać kopiowany program podając urządzenie „C:", a następnie zapisać go podając "D:nazwa", gdzie „nazwa" musi być poprawną nazwą pliku dyskowego.

***Planuję kupno Atari 520ST. Słyszałem, że istnieją wersje ST ze zmienionym systemem operacyjnym, co uniemożliwia działanie nowszych programów na modelach ze starym TOS-em. Którego modelu zakup polecalibyście i dlaczego? Z jakimi kolorowymi monitorami może współpracować Atari 520ST?***

***Paweł Radzikowski***
***Warszawa***

Pierwsza wersja systemu operacyjnego dla Atari ST powstała w roku 1985, a od 1987 komputery te posiadają wersję zmodyfikowaną. Część (niewielka) programów pisanych dla starej wersji nie działa z nową i odwrotnie. Wszystkie nowe programy są dostosowane do nowej wersji TOS. Również niektóre stare programy zostały w ten sposób przerobione. Programów dla nowego TOS-u będzie coraz więcej, a komputery ze starą wersją systemu nie są produkowane już od kilku lat. Do Atari ST należy stosować firmowy monitor kolorowy SC1224. W przypadku innych monitorów może wystąpić kłopot z podłączeniem. Ponadto modele mające w nazwie literę „M" mogą współpracować z telewizorem kolorowym PAL.

***Mam Atari 65XE z magnetofonem i postanowiłem dokupić drukarkę. Ze względów finansowych w rachubę wchodzi chyba tylko Atari 1029. Niestety, w Pewexie o tych drukarkach dawno już zapomnieli. Co robić?***

***Krzysztof Kowalski***
***Kraków***

Do komputerów Atari można stosować dowolne drukarki, pod warunkiem użycia interfejsu Centronics (np. MicroPrint dostępny w Pewexie). Najtańszymi dobrymi drukarkami są obecnie Star NX-1000 i LC-10 oraz Citizen 120D. Stanowczo odradzam zakup drukarki Atari 1029, która jest obecnie zabytkiem i kwalifikuje się wyłącznie do muzeum. Ponadto uważam, że najbardziej uzasadnionym zakupem w przedstawionej sytuacji byłaby stacja dysków — korzyść będzie znacznie większa.

***Posiadam komputer Atari 65XE i telewizor SANYO. Podłączyłem komputer do normalnej anteny, ale dowiedziałem się, że można poprawić grafikę i dźwięk. Czy można podłączyć do telewizora oddzielnie wizję i fonię z komputera?***

***Artur Maculewicz***
***Warszawa***

Do wyjścia monitorowego Atari doprowadzony jest sygnał fonii (AUDIO) i złożony sygnał wizji (COMPOSITE VIDEO). Sygnały te można wykorzystać poprzez wejście magnetowidowe lub monitorowe OTV. Schemat gniazda MONITOR był zamieszczony w „Tylko o Atari — 2". Sygnał AUDIO jest wyprowadzony na styk 3, VIDEO na styk 1, a masa na styk 2. Odpowiedni przewód połączeniowy należy wykonać samodzielnie lub nabyć w sklepie ze sprzętem RTV. W drugim przypadku zwykle od strony telewizora przewód ma cztery wtyki — wykorzystuje się tylko dwa z nich.

***Podczas wczytywania programu „Polskie Logo" gra normalna muzyka. Niektóre programy wczytują się do 20 minut, więc taka muzyka uprzyjemnia oczekiwanie. Jak można uzyskać ten efekt?***

***Paweł Hajderek***
***Chybie***

Magnetofony Atari zapisują sygnał komputerowy tylko na jednej ścieżce (w lewym kanale). Prawy kanał można nagrać po doprowadzeniu do niego sygnału akustycznego. Wymaga to doprowadzenia sygnału akustycznego do styku „Audio" na szynie równoległej komputera (PARALLEL BUS w XL lub ENHANCED CARTRIDGE INTERFACE w XE). Konieczne jest przy tym zachowanie ostrożności, aby nie uszkodzić komputera — doprowadzony sygnał nie może być przedtem wzmacniany (powinien mieć taki poziom jak sygnał otrzymany z tunera lub gramofonu). Można także wykonać odpowiednią przeróbkę magnetofonu tak, aby sygnał do prawego kanału był doprowadzany z zewnątrz.

***W „Moim Atari" 1, omawiany był problem naprawy zasilacza do komputera Atari 800XL. Z moich doświadczeń wynika, że zasilacz ten daje się naprawić.***

***W przypadku przerwy w uzwojeniu pierwotnym transformatora przyczyną może być uszkodzenie zamontowanego na początku tego uzwojenia bezpiecznika termicznego. Jest on umieszczony od spodu zasilacza i można się do niego dostać przy pomocy wkrętaka, delikatnie odłamując żywicę na uzwojeniu pierwotnym. Jeżeli po jego zwarciu przerwa ustąpiła, należy wymienić wtyczkę sieciową na większą i umieścić w niej bezpiecznik 315 mA/250 V.***

***Jeżeli przerwa występuje nadal, należy przy pomocy wkrętaka i młotka wydłubać transformator z żywicy. Przedtem trzeba odlutować trzy wyprowadzenia uzwojenia wtórnego do bloku elektroniki zasilacza. Po rozłożeniu rdzenia transformatora i usunięciu starego uzwojenia nawijamy nowe uzwojenie pierwotne — 1450 zwojów drutu 0,2 mm w emalii. Ponieważ przy wymianie uzwojenia przeważnie uszkadza się bezpiecznik termiczny, należy pamiętać o dodatkowym bezpieczniku 315 mA/250 V po stronie sieci zasilającej.***

***W przypadku uszkodzenia w części elektronicznej trzeba ją wydłubać z żywicy jak transformator. Tutaj najczęściej ulegają uszkodzeniu diody prostownicze. Jako zamienniki można zastosować diody 5 A/50 V (np. BYP680/50).***

***Po zakończeniu naprawy transformator i płytkę elektroniki umieszcza się w obudowie przy pomocy kleju Epidian.***

***Zbigniew Król***
***Kraków***


Loader do gier dla Atari
współpracujący ze stacją XF 551,
Umożliwiający szybki (Turbo) odczyt z obu stron dysku jednocześnie.
Cena 50 000,-
Zamówienia (realizowane za zaliczeniem pocztowym)
**kierować na adres: Jacek Szmyd**
**Wiosenna 15**
**35-303 Rzeszów**
107



**Na życzenie licznych czytelników podaję adres firmy ICD, której produkty zostały opisane w „Bajtku" 5–6/90.**

**ICD, Inc.**
**1220 Rock Street**
**Rockford, IL 61101-1437**
**U.S.A.**

# EDYTOR BASIC-a

**Przy przepisywaniu programów w Basicu nie sposób ustrzec się błędów.**

Aby uniknąć żmudnego wyszukiwania popełnianych omyłek wszystkie programy w Basicu są publikowane wraz z kodami kontrolnymi. Zamieszczony obok program służy do kontroli tych kodów podczas przepisywania programu.

Wydruk „Edytora Basica" należy dokładnie przepisać i zapisać na kasecie lub dyskietce (najlepiej instrukcją LIST „C:" lub LIST „D:EDYTOR.LST"). Poprawność przepisania można sprawdzić samym „Edytorem" w opisany niżej sposób.

Przystępując do wpisywania dowolnego programu z naszego pisma trzeba najpierw wczytać i uruchomić „Edytor Basica". Następnie należy przepisywać kolejne wiersze programu. Po wpisaniu każdego wiersza i naciśnięciu klawisza RETURN wiersz ten pojawia się w dowolnej części ekranu wraz z obliczonym kodem kontrolnym. Jeżeli wyświetlony kod jest taki sam, jak wydrukowany przed numerem wiersza, można przystąpić do wpisywania następnego wiersza. Jeśli kody są różne, to ponowne naciśnięcie RETURN wyświetla wpisany wiersz w górnej części ekranu i umożliwia dokonanie w nim niezbędnych poprawek.

Samo naciśnięcie RETURN wywołuje zawsze ostatnio wpisany wiersz. W celu wywołania innego, wcześniej napisanego wiersza trzeba podać jego numer poprzedzony gwiazdką (np. 1000) i nacisnąć RETURN. Wpisanie samej liczby powoduje wymazanie z pamięci komputera wiersza programu o takim numerze.

Po poprawnym przepisaniu całego programu trzeba przerwać pracę „Edytora Basica" przez naciśnięcie klawisza BREAK lub RESET. Następnie w celu usunięcia „Edytora" i zbędnych zmiennych (z tablic nazw i wartości zmiennych) zapisujemy program na kasecie instrukcją LIST „C:",0,31999 lub na dyskietce instrukcją LIST „D:nazwa",0,31999. Teraz kasujemy zawartość pamięci komputera instrukcją NEW i odczytujemy program przy pomocy ENTER „C:" (z kasety) lub ENTER „D:nazwa" z dyskietki. Po wykonaniu tych czynności w pamięci komputera znajduje się tylko gotowy program bez „Edytora Basica" i można go już ostatecznie zapisać na odpowiedni nośnik.

Procedura ta jest może nieco kłopotliwa, lecz zabezpiecza w stu procentach przed popełnieniem omyłki przy przepisywaniu programu. PAMIĘTAJ: ZAWSZE UŻYWAJ „EDYTORA BASICA".

```
NC  32000 REM EDYTOR BASICA
SE  32010 REM wersja 1.1 dla "Bajtka"
JB  32020 CLR :DIM LINIA$(120):CLOSE #2:CL
    OSE #3
MN  32030 OPEN #2,4,0,"E:":OPEN #3,5,0,"E:
    "
ZR  32040 ? CHR$(125):POSITION 11,1:? "■ED
    YTOR BASICA ■"
YB  32050 TRAP 32040:POSITION 2,3:? "Wpisz
     linie programu"
KH  32060 POSITION 1,4:? " ":INPUT #2;LINI
    A$:IF LINIA$="" THEN POSITION 2,4:LIST
     B:GOTO 32060
XL  32070 IF LINIA$(1,1)="*" THEN B=VAL(LI
    NIA$(2,LEN(LINIA$))):POSITION 2,4:LIST
     B:GOTO 32060
TH  32080 POSITION 2,10:? "CONT"
BI  32090 B=VAL(LINIA$):POSITION 1,3:? " "
    ;
NY  32100 POKE 842,13:STOP
CN  32110 POKE 842,12
KI  32120 ? CHR$(125):POSITION 11,1:? "■ED
    YTOR BASICA ":POSITION 2,15:LIST B
BZ  32130 C=0:ODP=C
LL  32140 POSITION 2,16:INPUT #3;LINIA$:IF
     LINIA$="" THEN ? "LINIA ";B;" USUNIET
    A":GOTO 32050
GU  32150 FOR D=1 TO LEN(LINIA$):C=C+1:ODP
    =ODP+(C*ASC(LINIA$(D,D))):NEXT D
SR  32160 KOD=INT(ODP/676)
BX  32170 KOD=ODP-KOD*676
BM  32180 KODS=INT(KOD/26)
VF  32190 KODM=KOD-(KODS*26)+193
NF  32200 KODS=KODS+193
UW  32210 POSITION 0,16:? CHR$(KODS);CHR$(
    KODM)
HN  32220 POSITION 2,13:? "Jezeli kod sie
    nie zgadza, nacisnij ■RETURN■ i popr
    aw linie.":GOTO 32050
```

# WPISYWANIE ZNAKÓW SPECJALNYCH

**Część programów publikowana w naszym piśmie jest drukowana w trybie graficznym.**

Jest to spowodowane występowaniem w tych programach znaków, które są stosowane tylko w komputerach Atari i nie mogą być przedstawione przez drukarkę w trybie znakowym.

W trybie graficznym znaki znajdujące się obok siebie są drukowane bez odstępów (jak na ekranie monitora), lecz między poszczególnymi wierszami pozostaje niewielki odstęp (inaczej niż na ekranie). Poniżej pokazany jest wygląd wszystkich znaków uwidocznionych na klawiaturze.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
0123456789          0123456789
!"#$%&'@()<>-=;+*,./_:\^[]?
!"#$%&'@()<>-=;+*,./_:\^[]?
```

Zestaw znaków Atari zawiera ponadto znaki specjalne, które wymagają specjalnego wpisywania. Wygląd tych znaków oraz klawisze, które służą do ich uzyskania, są pokazane w tabelkach umieszczonych obok. Zawsze, gdy są używane klawisze ‹CONTROL› (w tabelach „CTRL") i ‹SHIFT›, trzeba trzymać je wciśnięte podczas naciskania następnego klawisza. Gdy konieczne jest użycie klawisza ‹ESC›, należy go nacisnąć tylko jeden raz i zwolnić przed naciśnięciem następnego klawisza. Znaki pokazane w tabeli „INVERSE" są negatywami znaków z tabeli „NORMALNIE". Przełączenie na znaki w negatywie i odwrotnie następuje po naciśnięciu klawisza z symbolem przekreślonego prostokąta (tzw. klawisz Atari). Przełączanie na pisanie w negatywie (lub odwrotnie) nie jest konieczne dla uzyskania znaków, które są uzyskiwane przy użyciu ‹ESC›. Są one zawsze wyświetlane w sposób pokazany w tabelach.

Błędy najczęściej występujące przy przepisywaniu programów są powodowane przez wpisanie dużej litery zamiast małej — dotyczy to szczególnie liter P, X i O. Ponadto łatwo pomylić literę O z cyfrą 0, kropkę (.) z przecinkiem (,) oraz średnik (;) z dwukropkiem (:). Podobnie, niektóre znaki specjalne mają wygląd bardzo zbliżony do zwykłych znaków alfanumerycznych. Na wydruku rozróżnić je można tylko według grubości linii. Dla porównania i zapamiętania znaki te są pokazane poniżej.

**SPECJALNE**

| | |
|---|---|
| ╱ | CTRL F |
| ╲ | CTRL G |
| ▬ | CTRL N |
| ▒ | CTRL R |
| ⁘ | CTRL S |

**NORMALNE**

| | |
|---|---|
| ╱ | znak / |
| ╲ | SHIFT + |
| ▬ | SHIFT - |
| ▒ | znak - |
| ⁘ | znak + |

**NORMALNIE**

| | |
|---|---|
| ♥ | CTRL , |
| ├ | CTRL A |
| │ | CTRL B |
| ┘ | CTRL C |
| ┤ | CTRL D |
| ┐ | CTRL E |
| ╱ | CTRL F |
| ╲ | CTRL G |
| ◢ | CTRL H |
| ▝ | CTRL I |
| ▙ | CTRL J |
| ▘ | CTRL K |
| ▖ | CTRL L |
| ▗ | CTRL M |
| ▁ | CTRL N |
| ▚ | CTRL O |
| ♣ | CTRL P |
| ▛ | CTRL Q |
| ▔ | CTRL R |
| ┼ | CTRL S |
| ● | CTRL T |
| ▃ | CTRL U |
| ▏ | CTRL V |
| ▎ | CTRL W |
| ▍ | CTRL X |
| ▌ | CTRL Y |
| └ | CTRL Z |
| ESC | ESC |
| ↑ | ESC CTRL - |
| ↓ | ESC CTRL = |
| ← | ESC CTRL + |
| → | ESC CTRL * |
| ● | CTRL . |
| ♠ | CTRL ; |
| ▏ | SHIFT = |
| ↖ | ESC CTRL CLEAR |
| ◀ | ESC DELETE |
| ▶ | ESC TAB |

**INVERSE**

| | |
|---|---|
| ♥ | CTRL , |
| ├ | CTRL A |
| │ | CTRL B |
| ┘ | CTRL C |
| ┤ | CTRL D |
| ┐ | CTRL E |
| ╱ | CTRL F |
| ╲ | CTRL G |
| ◢ | CTRL H |
| ▝ | CTRL I |
| ▙ | CTRL J |
| ▘ | CTRL K |
| ▖ | CTRL L |
| ▗ | CTRL M |
| ▁ | CTRL N |
| ▚ | CTRL O |
| ♣ | CTRL P |
| ▛ | CTRL Q |
| ▔ | CTRL R |
| ⁘ | CTRL S |
| ○ | CTRL T |
| ▃ | CTRL U |
| ▏ | CTRL V |
| ▎ | CTRL W |
| ▍ | CTRL X |
| ▌ | CTRL Y |
| └ | CTRL Z |
| ESC | ESC SHIFT DELETE |
| ESC | ESC SHIFT INSERT |
| ESC | ESC CTRL TAB |
| ESC | ESC SHIFT TAB |
| ○ | CTRL . |
| ♠ | CTRL ; |
| ▏ | SHIFT = |
| ESC | ESC CTRL 2 |
| ESC | ESC CTRL DELETE |
| ESC | ESC CTRL INSERT |

# TRENING NA KLAWIATURZE

**„Łamacze joysticków”! — to wyzwanie dla was. Machanie kijem nie jest żadną sztuką. Spróbujcie swojej szybkości w tej grze.**

Podstawowy cel tej gry jest jednak zupełnie inny — umożliwia ona szybkie opanowanie układu klawiatury komputera. Może być więc doskonałym ćwiczeniem zarówno dla amatorów programowania, jak i kandydatki na maszynistki.

Program ma trzy poziomy trudności, wybierane klawiszem ‹OPTION›. Na pierwszym czas na naciśnięcie klawisza z wyświetlonym znakiem wynosi trzy sekundy, na drugim dwie i na trzecim jedną. Ponadto klawiszem ‹SELECT› włącza się dodatkowe utrudnienie, gdyż wyświetlane są również znaki graficzne. Grę uruchamia klawisz ‹START›.

Po naciśnięciu klawisza, który nie odpowiada wyświetlonemu znakowi, gra jest przerywana i można zaczynać od początku. Program podaje aktualny wynik w punktach (jego wielkość zależy od poziomu trudności) oraz liczbę prawidłowo wybranych klawiszy. Poniżej wyświetlany jest najlepszy z uzyskanych wyników. Gdy zupełnie znudzi się ciągłe naciskanie klawiszy, wystarczy skorzystać z kombinacji ‹OPTION›+‹START›

*Konrad Makarewicz*

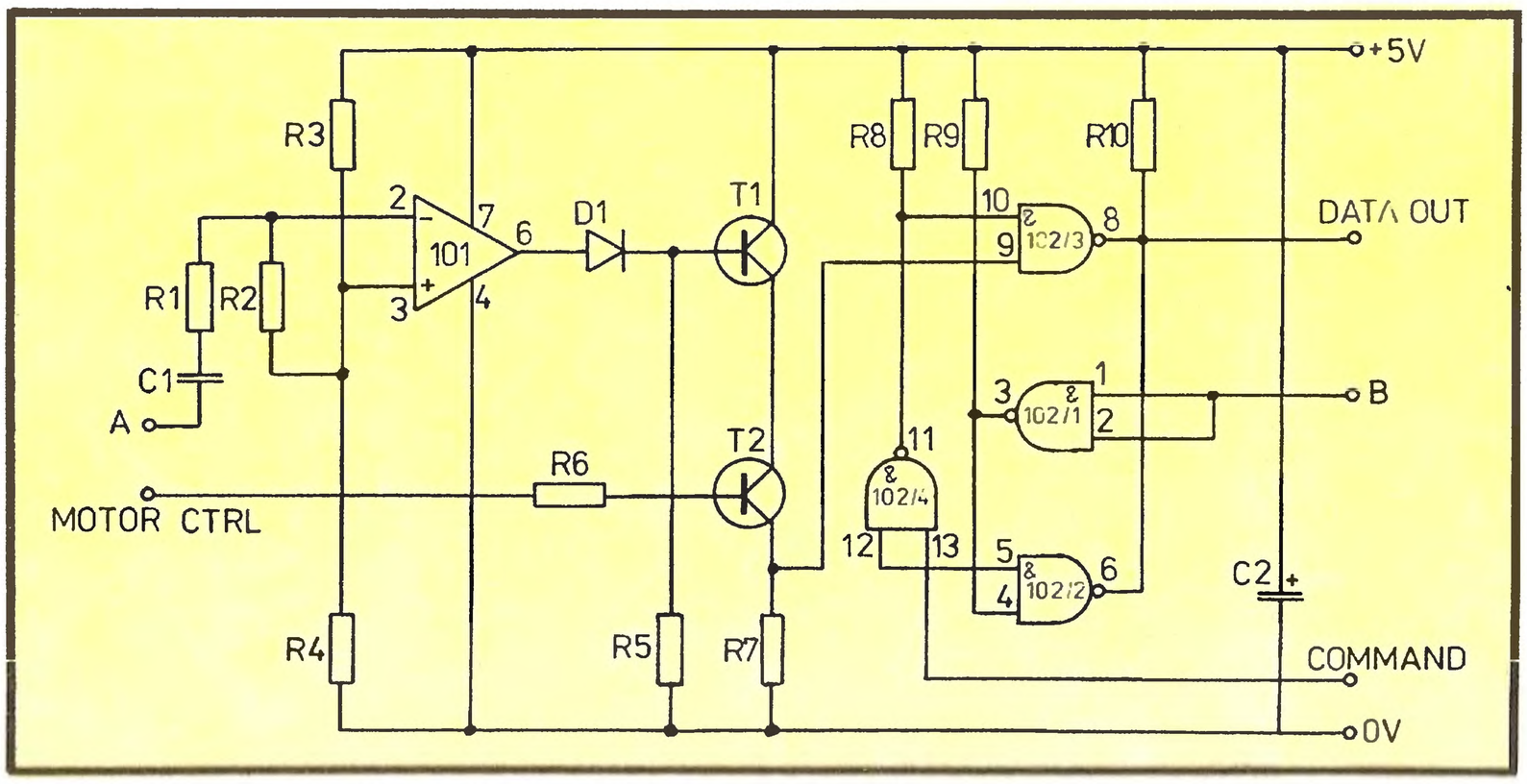
```
LZ    4 REM ****************************
GH    5 REM * TRENING  NA  KLAWIATURZE *
NK    6 REM * *  Konrad  Makarewicz  * *
FN    7 REM * * (c) 1991, Sp. Bajtek * *
MD    8 REM ****************************
NO    9 REM
SF    10 CLR :CLOSE #5:OPEN #5,4,0,"K:":GRAP
      HICS 7+16:POKE 566,205:GOTO 1000
LY    15 ON PEEK(FUN) GOTO S1,S2,S3,S1,S5,S6
      ,S1:GOTO S1
HD    20 GRAPHICS 7+16:POKE DL4,C0:POKE DL5,
      SE2:POKE 88,C0:POKE 89,SE2:POKE 708,18
      1:POKE 709,9:POKE 710,240
UW    25 PWK=C0:PLNK=C0:GOTO 70
CJ    30 FOR I=CZO TO C0 STEP -1:IF PEEK(764
      )<>255 THEN GET #5,D:K$=CHR$(D)
FT    35 IF D=DZ THEN POP :A=USR(AP,W32,C0,X
      ,Y):PLNK=PLNK+C1:PWK=PWK+I:RETURN
GB    40 NEXT I:POP :SOUND C0,255,L6,S1:A=US
      R(AP,W32,C0,X,Y):FOR I=1 TO 10:Z=ASC(K
      G$(I)):A=USR(AP,Z,C2,4+I,53):NEXT I
MD    45 GOTO 100
MJ    49 REM * Wyswietlenie znaku *
NF    50 X=3:Y=20:A=USR(AP,DZ,C1,X,Y):GOSUB
      30:RETURN
OB    51 X=3:Y=40:A=USR(AP,DZ,C1,X,Y):GOSUB
      30:RETURN
OX    52 X=3:Y=60:A=USR(AP,DZ,C1,X,Y):GOSUB
      30:RETURN
BN    53 X=16:Y=20:A=USR(AP,DZ,C1,X,Y):GOSUB
       30:RETURN
CL    54 X=16:Y=40:A=USR(AP,DZ,C1,X,Y):GOSUB
       30:RETURN
DJ    55 X=16:Y=60:A=USR(AP,DZ,C1,X,Y):GOSUB
       30:RETURN
RM    56 X=7:Y=45:A=USR(AP,DZ,C1,X,Y):GOSUB
      30:RETURN
DX    57 X=12:Y=45:A=USR(AP,DZ,C1,X,Y):GOSUB
       30:RETURN
FB    58 X=2:Y=2:A=USR(AP,DZ,C1,X,Y):GOSUB 3
      0:RETURN
RW    59 X=17:Y=2:A=USR(AP,DZ,C1,X,Y):GOSUB
      30:RETURN
OQ    69 REM * Losowanie znaku i Pozycji *
CV    70 DZ=INT(RND(0)*GKZ):ON BPZ GOTO 75,8
      5
OJ    75 IF DZ<W32 THEN 70
UJ    80 GOTO 90
LB    85 IF DZ>26 AND DZ<W32 THEN 70
WE    90 DS=INT(RND(0)*11):ON DS GOSUB 50,51
      ,52,53,54,55,56,57,58,59:GOTO 70
SJ    99 REM * Obliczanie wYnikow *
KO    100 WK(0)=INT(PWK/L1):WK(1)=INT(PWK/L2
      -WK(0)*L6):WK(2)=INT(PWK/L3-WK(0)*L5-W
      K(1)*L6)
VL    105 WK(3)=INT(PWK/L4-WK(0)*L4-WK(1)*L5
      -WK(2)*L6):WK(4)=INT(PWK/L5-WK(0)*L3-W
      K(1)*L4-WK(2)*L5-WK(3)*L6)
LZ    110 WK(5)=INT(PWK/L6-WK(0)*L2-WK(1)*L3
      -WK(2)*L4-WK(3)*L5-WK(4)*L6)
EQ    115 WK(6)=PWK-WK(0)*L1-WK(1)*L2-WK(2)*
      L3-WK(3)*L4-WK(4)*L5-WK(5)*L6
EK    120 LNK(0)=INT(PLNK/L4):LNK(1)=INT(PLN
      K/L5-LNK(0)*L6):LNK(2)=INT(PLNK/L6-LNK
      (0)*L5-LNK(1)*L6)
AV    125 LNK(3)=PLNK-LNK(0)*L4-LNK(1)*L5-LN
      K(2)*L6
DG    130 FOR I=0 TO 6:WK(I)=WK(I)+48:NEXT I
      :FOR I=0 TO 3:LNK(I)=LNK(I)+48:NEXT I
DN    135 FOR I=0 TO 255:SOUND C0,I,L6,S1:NE
      XT I:SOUND C0,100,C2,S1:FOR I=1 TO 10:
      A=USR(AP,INV,C2,4+I,53)
EB    140 FOR P=0 TO 49:NEXT P:NEXT I:SOUND
      C0,C0,C0,C0:GOTO 1010
WE    299 REM * Select *
AI    300 ON PEEK(DL2) GOTO 310,320
DW    310 SOUND C0,100,C2,S1:POSITION 0,8:?
      #6;"PODST. SPEC. I GRAF.":FOR I=0 TO 9
      9:NEXT I
XN    315 BPZ=C2:GKZ=97:POKE DL2,2:SOUND C0,
      C0,C0,C0:GOTO S1
UB    320 SOUND C0,40,C2,S1:POSITION 5,8:? #
      6;"AWOWE I SPEC.  ":FOR I=0 TO 99:NEXT
       I
VM    325 BPZ=C1:GKZ=96:POKE DL2,1:SOUND C0,
      C0,C0,C0:GOTO S1
IZ    399 REM * Option *
LB    400 PTG=PTG+C1:CZO=CZO-100:IF PTG>C2 T
      HEN PTG= NOT PTG:CZO=299
NY    405 POSITION 16,6:? #6;PTG:FOR I=15 TO
       0 STEP -1:SOUND C0,20,C2,I:FOR P=0 TO
       9:NEXT P:NEXT I:GOTO S1
HE    499 REM * Start+Option *
KI    500 A=USR(58496)
XE    999 REM * Strona tYtulowa *
LN    1000 GOSUB 30000:DIM WK(6),LNK(3),NWK(
      6),NLNK(3),KG$(10),K$(1)
YN    1005 FOR I=0 TO 6:WK(I)=48:NWK(I)=48:N
      EXT I:FOR I=0 TO 3:LNK(I)=48:NLNK(I)=4
      8:NEXT I:KG$="KONIEC GRY"
QH    1010 GRAPHICS C2:ANT=PEEK(560)+256*PEE
      K(561):POKE 559,0:POKE 752,1:POSITION
      4,0:? #6;"TRENING  NA":POSITION 4,1
NV    1015 ? #6;"KLAWIATURZE":POSITION 0,3:?
       #6;"wynik 0000000 (0000)":POSITION 0,
      4:? #6;"n.w.  0000000 (0000)"
OK    1020 POSITION 0,6:? #6;"Option - POZIO
      M ";PTG:POSITION 0,7:? #6;"select - ZN
      AKI :"
MI    1025 POSITION 0,8:? #6;"PODSTAWOWE I S
      PEC.":? "Konrad Makarewicz , Klimonto
      w 1990":?
YF    1030 ? "        Copyright (c) Bajtek":
      POKE DL2,1:BPZ=C1:GKZ=96
QP    1035 FOR I=0 TO 6:POSITION 6+I,3:? #6;
      CHR$(WK(I)+SE2):NEXT I:FOR I=0 TO 3:PO
      SITION 15+I,3:? #6;CHR$(LNK(I)+SE2)
ZK    1040 NEXT I:IF PWK>PNWK THEN PNWK=PWK:
      GOSUB 1100
PL    1045 GOSUB 1105:RESTORE 30500:FOR I=0
      TO 8:READ DDL:POKE ANT+6+I,DDL:NEXT I
BS    1050 POKE 559,34:GOTO S1
BK    1100 FOR I=0 TO 6:NWK(I)=WK(I):NEXT I:
      FOR I=0 TO 3:NLNK(I)=LNK(I):NEXT I
OB    1105 FOR I=0 TO 6:POSITION 6+I,4:? #6;
      CHR$(NWK(I)+SE2):NEXT I:FOR I=0 TO 3:P
      OSITION 15+I,4
CK    1110 ? #6;CHR$(NLNK(I)+SE2):NEXT I:RET
      URN
DG    24999 REM * Plansza GrY *
CX    25000 DIM CK1$(72),CK2$(72),CK3$(81),C
      K4$(81),CK5$(72),CK6$(100),CK7$(81),CK
      8$(81)
WD    25005 CK1$="BBBBBBBBABBBBBAAABBBBAAADBB
      BBAAAABBBAAAAABBAAAADABAAAAAAAABAADAAAA
      AAAAAAAA"
AH    25006 CK2$="DABBBBBBAAAABBBBAAADBBBBAA
      AAABBBADAADABBAAAAAABBAAAADAABADAAAAAB
      DAAAAADA"
KP    25007 CK3$="BAAAAADAAAAAAAAAAAAADAAAA
      AADAAAADAABAAAAAAAABBAAADAAABBBBAAAAAB
      BBBBBBBCBBBBBBBBC"
UK    25008 CK4$="AAAAAAAAAAAADAAAAAAAAAADAA
      BDAAAAAAAAAAADAADABDAAAAAABBAAAAABBBBC
      BBBBBBBBCBBBBBBBB"
ZF    25009 CK5$="DDDDADDDDDDAADDDDDDAADDDDD
      AADADDDDAAAADDDADAAAADDAAADAADAAAAAADA
      ADAAAAAA"
GX    25010 CK6$="DAAAADADADAAAADAAAAADAAAA
      AAADAAADAADAAAAAAAAAAAADAAAADADADDADA
      AAAAADDDAADADADDDDDAAAADDDDDDDCCDDDD"
JM    25011 CK7$="DAAADAADAAADAAAAAAAAAADAD
      DADADAAAADAAAADADDDDADAADDDDAAADDDDDDC
      BDDDDDDDCBDDDDDDD"
OG    25012 CK8$="ABDDDDDDDAAADDDDDDAAAAADDD
      DADAADADDDAAAAAAADDAADADADADAAAAAAAADD
      AADAAADAAADAADAAD"
RT    25015 COLOR C1:FOR I=80 TO 95:PLOT 0,I
      :DRAWTO 159,I:NEXT I:COLOR C2:FOR I=10
       TO 40:PLOT I,10:DRAWTO I,79:NEXT I
EL    25020 FOR I=40 TO 79:PLOT 41,I:DRAWTO
      119,I:NEXT I:FOR I=120 TO 150:PLOT I,1
      0:DRAWTO I,79:NEXT I
QR    25025 COLOR C0:FOR I=65 TO 79:PLOT 68,
      I:DRAWTO 90,I:NEXT I:PLOT 69,64:DRAWTO
       89,64:PLOT 71,63:DRAWTO 87,63
UL    25030 PLOT 74,62:DRAWTO 83,62:PLOT 78,
      61:PLOT 79,61:COLOR C3:FOR I=65 TO 75
      STEP 2:PLOT 68,I:DRAWTO 90,I:NEXT I
XR    25035 FOR I=69 TO 89 STEP 2:PLOT I,64:
      DRAWTO I,76:NEXT I:PLOT 70,63:DRAWTO 8
      8,63:FOR I=77 TO 83 STEP 2:PLOT I,62
AX    25040 NEXT I:FOR I=1 TO 10:POSITION 0,
      69+I:? #6;CK6$(I*10-9,I*10):NEXT I
IO    25045 FOR I=1 TO 8:POSITION 1,61+I:? #
      6;CK5$(I*8-7,I*8):NEXT I:FOR I=1 TO 9:
      POSITION 38,70+I
MK    25050 ? #6;CK3$(I*9-8,I*9):NEXT I:FOR
      I=1 TO 9:POSITION 47,70+I:? #6;CK4$(I*
      9-8,I*9):POSITION 140,70+I
GJ    25055 ? #6;CK3$(I*9-8,I*9):NEXT I:FOR
      I=1 TO 8:POSITION 39,62+I:? #6;CK1$(I*
      8-7,I*8):POSITION 141,62+I
JS    25060 ? #6;CK1$(I*8-7,I*8):NEXT I:FOR
      I=1 TO 8:POSITION 47,62+I:? #6;CK2$(I*
      8-7,I*8):NEXT I
YF    25065 FOR I=1 TO 9:POSITION 149,62+I:?
       #6;CK8$(I*9-8,I*9):POSITION 149,70+I:
      ? #6;CK7$(I*9-8,I*9):NEXT I
HK    25070 FOR I=6 TO 14 STEP 2:A=USR(AP,IN
      V,C2,I,35):NEXT I:A=USR(AP,INV,C2,1,7)
      :A=USR(AP,INV,C2,3,7)
RX    25075 A=USR(AP,INV,C2,5,7):A=USR(AP,IN
      V,C2,14,7):A=USR(AP,INV,C2,16,7):A=USR
      (AP,INV,C2,18,7)
CG    25080 FOR I=20 TO 60 STEP 20:A=USR(AP,
      W32,C0,3,I):A=USR(AP,W32,C0,16,I):NEXT
       I:A=USR(AP,W32,C0,7,45)
VR    25085 A=USR(AP,W32,C0,12,45)
DL    25100 RETURN
LH    29999 REM * Inicjalizacja *
EY    30000 POKE 559,0:FOR I=0 TO 252:READ D
      :POKE 1536+I,D:NEXT I:DL=PEEK(560)+256
      *PEEK(561)
JC    30005 DL2=DL-2:DL4=DL+4:DL5=DL+5:READ
      L1,L2,L3,L4,L5,L6,FUN,CZO,GKZ,BPZ,S1,S
      2,S3,S5,S6,SE2,AP,W32,RAN
BG    30010 C1=BPZ:C2=C1+C1:C3=C1+C2:INV=W32
      +SE2
CS    30015 POKE DL4,C0:POKE DL5,SE2:POKE 88
      ,C0:POKE 89,SE2:GOSUB 25000:RETURN
HS    30099 REM * TextPlot *
DD    30100 DATA 104,240,10,201,4,240
EN    30105 DATA 11,170,104,104,202,208
HV    30110 DATA 251,169,253,76,164,246
ZH    30115 DATA 104,133,195,104,201,128
MK    30120 DATA 144,4,41,127,198,195
CF    30125 DATA 170,141,250,6,224,96
DS    30130 DATA 176,15,169,64,224,32
BF    30135 DATA 144,2,169,224,24,109
HQ    30140 DATA 250,6,141,250,6,104
UO    30145 DATA 104,141,251,6,104,104
IJ    30150 DATA 141,252,6,14,252,6
ZX    30155 DATA 104,104,141,253,6,133
RY    30160 DATA 186,166,87,169,10,224
LJ    30165 DATA 3,240,8,169,20,224
IC    30170 DATA 5,240,2,169,40,133
IA    30175 DATA 207,133,187,165,88,133
HG    30180 DATA 203,165,89,133,204,32
PZ    30185 DATA 228,6,24,173,252,6
RJ    30190 DATA 101,203,133,203,144,2
EN    30195 DATA 230,204,24,165,203,101
ON    30200 DATA 212,133,203,165,204,101
BY    30205 DATA 213,133,204,173,250,6
XR    30210 DATA 133,187,169,8,133,186
WE    30215 DATA 32,228,6,165,212,133
TR    30220 DATA 205,173,244,2,101,213
LJ    30225 DATA 133,206,160,0,162,8
HP    30230 DATA 169,0,133,208,133,209
SL    30235 DATA 177,205,69,195,72,104
MM    30240 DATA 10,72,144,8,24,173
RQ    30245 DATA 251,6,5,208,133,208
GF    30250 DATA 224,1,240,8,6,208
CU    30255 DATA 38,209,6,208,38,209
QE    30260 DATA 202,208,228,104,152,72
EQ    30265 DATA 160,0,165,209,145,203
WL    30270 DATA 200,165,208,145,203,104
NG    30275 DATA 168,24,165,203,101,207
SX    30280 DATA 133,203,144,2,230,204
MT    30285 DATA 200,192,8,208,183,96
PL    30290 DATA 169,0,133,212,162,8
KR    30295 DATA 70,186,144,3,24,101
UB    30300 DATA 187,106,102,212,202,208
DW    30305 DATA 243,133,213,96,0,1
KS    30310 DATA 28
IR    30399 REM * Dane dla Programu *
KU    30400 DATA 1000000,100000,10000,1000,1
      00,10,53279,299,96,1,15,500,400,300,20
      ,128,1536,32,53770
IU    30499 REM * Program ANTIC-a *
JQ    30500 DATA 7,6,6,6,6,7,7,6,7
```

# TURBO DO MAGNETOFONU

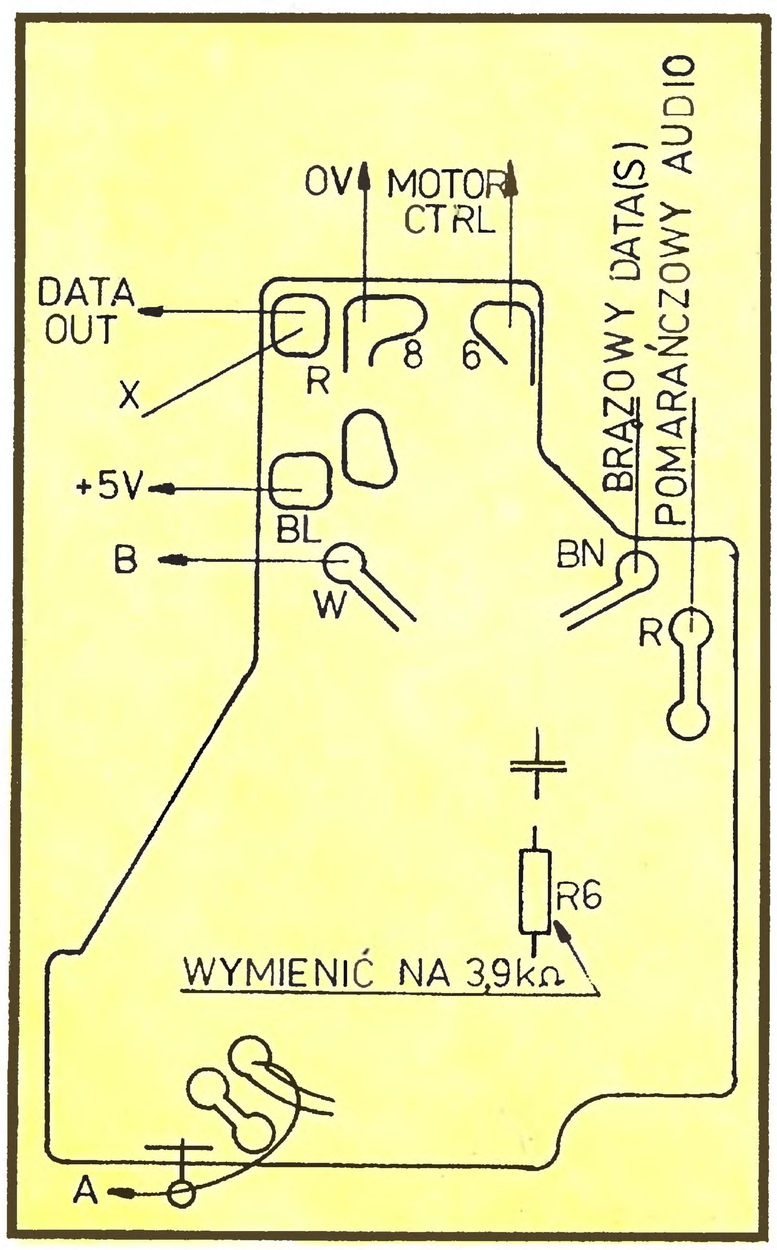
**Magnetofony Atari ze względu na małą prędkość transmisji są najczęściej przerabianym urządzeniem tej firmy. Spełniając życzenia licznych czytelników publikujemy schemat prostego interfejsu do magnetofonu XC12. Interfejs ten umożliwia zwiększenie prędkości transmisji do 5000 bodów.**

Schemat interfesju jest pokazany na rysunku 1. Po zgromadzeniu części według wykazu znajdującego się na końcu artykułu należy zaprojektować płytkę drukowaną i zmontować na niej układ. Jeżeli płytka będzie mała (nie więcej niż 50 na 35 mm), to można ją umieścić wewnątrz obudowy magnetofonu. W przeciwnym razie konieczne będzie wykonanie dla niej odrębnej obudowy i połączenie przewodem z magentofonem.

Montaż układu polega na przylutowaniu jego końcówek do miejsc oznaczonych takimi samymi symbolami na rysunku 2, który przedstawia płytkę drukowaną magnetofonu. Przewód musi być koniecznie ekranowany, a jego ekran lutujemy do płytki magnetofonu we wskazanym miejscu. Dodatkowy przewód COMMAND należy doprowadzić do styku 7 we wtyczce kabla łączącego magnetofon z komputerem. Ponadto należy odlutować biały przewód z pola oznaczonego literą W i przylutować go do




*Rys. 1 Schemat ideowy interfejsu TURBO.*




*Rys. 2. Miejsce połączeń na płytce magnetofonu.*

pola R (wskazanego przez X). Ostatnią czynnością będzie wymiana na płytce magentofonu rezystora R6 na rezystor o oporności 3,9 kΩ.

Poprawnie zmontowany układ powinien działać bez zarzutu, lecz można przy jego uruchamianiu skontrolować oscyloskopem lub sondą logiczną sygnał występujący na emiterze tranzystora T2. Ewentualną regulację można przeprowadzić przez dobór rezystorów R3 i R7.

*Marek Zachar*

## Wykaz części:

| | |
|---|---|
| 101 | — μA 741C |
| 102 | — UCY 7403 |
| T1,T2 | — BC 108B |
| D1 | — BAP 794 |
| C1 | — 15 pF |
| C2 | — 10 μF |
| R1,R2 | — 1 MΩ |
| R3 | — 1,8 KΩ |
| R4,R6 | — 10 kΩ |
| R5 | — 18 kΩ |
| R7 | — 270 Ω |
| R8,R9,R10 | — 3,3 kΩ |
| R dodatk. | — 3,9 kΩ |

# ROZSZERZENIE PAMIĘCI DO 256 KB

**Znane powiedzonko informatyków mówi, że każdy komputer ma zbyt mało pamięci. Jest to oczywiste dla każdego, kto wykorzystuje komputer do pracy, lecz również zapaleni gracze zdają sobie z tego sprawę.**

Prędzej, czy później każdy posiadacz komputera postanawia powiększyć pamięć w swoim sprzęcie. Stało się tak również w naszym przypadku. Posiadany przez redakcję Atari 65XE został wyposażony w pamięć RAM o pojemności 256 KB. Przeróbka ta jest dziełem autorów systemów **TOMS — Tadeusza Okonia (Warszawa, ul. Lachmana 4/44, tel. 641–54–29) i Marka Stolarskiego (Warszawa, ul. Kazury 13/26, tel. 46–01–02).**

Rozszerzenie polega na wymianie standardowych układów pamięci 64 KB (w 800XL i 65XE) lub 128 KB (w 130XE) na układy o pojemności 256 KB. Ponieważ zastosowany w Atari procesor 6502 potrafi rozpoznać tylko 64 KB pamięci, to dostęp do pozostałego jej obszaru należało zrealizować przez wymianę tzw. banków. Dodatkowe 192 KB pamięci zostały więc podzielone na 12 części (banków) po 16 KB. Każdy bank ma takie same adresy — od 16384 do 32767. Przełączanie poszczególnych banków wykonuje się przez zmianę zawartości rejestru PORTB (adres 54017).

Rejestr PORTB jest połączony z układem sterującym pamięcią komputerów Atari. Jego najstarszy bit (7) służy do włączania i wyłączania programu testującego. Bit 0 steruje pamięcią ROM zawierającą system operacyjny, a bit 1 — zawierającą interpreter Basica. Cztery z pozostałych bitów tego rejestru (2, 3, 5 i 6) zostały wykorzystane do sterowania dodatkową pamięcią. Starsza para bitów wybiera grupę banków, młodsza zaś para — bank w grupie. Wartość 253 standardowo zapisywana przez system operacyjny w rejestrze PORTB podczas uruchamiania komputera powoduje wybranie banków pamięci odpowiadające komputerowi, który posiada 64 KB RAM.

Wartości uaktywniające poszczególne banki pamięci są następujące:

| bin | hex | dec | bank |
|---|---|---|---|
| %10000001 | $81 | 129 | 0 260XE |
| %10000101 | $85 | 133 | 1 260XE |
| %10001001 | $89 | 137 | 2 260XE |
| %10001101 | $8D | 141 | 3 260XE |
| %10100001 | $A1 | 161 | 4 260XE |
| %10100101 | $A5 | 165 | 5 260XE |
| %10101001 | $A9 | 169 | 6 260XE |
| %10101101 | $AD | 173 | 7 260XE |
| %11000001 | $C1 | 193 | 0 130XE |
| %11000101 | $C5 | 197 | 1 130XE |
| %11001001 | $C9 | 201 | 2 130XE |
| %11001101 | $CD | 205 | 3 130XE |
| %11111101 | $FD | 253 | tryb 64 KB |

Zwiększenie wartości podanych w tabeli 2 odłącza pamięć ROM zawierającą interpreter Basica. Oczywiście wykonanie takiej operacji w Basicu spowoduje zawieszenie komputera.

Jak można wykorzystać powiększoną pamięć komputera? Najczęściej jest ona używana do założenia ramdysku. Jednakże nie wszystkie DOS-y potrafią właściwie rozpoznać pojemność pamięci i zainstalować odpowiedni ramdysk. Wynika to głównie z ograniczeń samych DOS-ów, które były pisane dla innych konfiguracji sprzętowych (jest to najbardziej widoczne w przypadku DOS 2.5). Niebagatelne znaczenie ma również duża liczba metod wyboru banków. Optymalne wykorzystanie dodatkowej pamięci wymaga więc zastosowania odpowiedniego DOS-u. Przeprowadzone testy dały następujące rezultaty:

| rodzaj DOS-u | pojemność ramdysku w sektorach | w bajtach |
|---|---|---|
| DOS 2.5 | 499 | 62872 |
| MYDOS 4.50 | 1522 | 194816 |
| SuperDOS | 1010 | 129280 |
| BiboDOS 5.4 | 1004 | 128512 |
| SpartaDOS 3.2 | 1019 | 130432 |
| SpartaDOS X | 1019 | 130432 |

Przy przeprowadzaniu tych testów stwierdziłem w niektórych przypadkach dziwne zjawisko. Niekiedy ramdysk po zainstalowaniu nie był formatowany i konieczne było oddzielne wykonanie tej operacji. Nieregularność występowania tego efektu nie pozwala jednak na wyciągnięcie żadnych wniosków. Przypuszczam, że powodem występowania takiego zjawiska może być zbyt długi czas „zapominania" zawartości pamięci po wyłączeniu zasilania albo niezerowanie jej zawartości podczas zimnego startu (system operacyjny zeruje tylko podstawowe 64 KB pamięci).

Wiele programów posiada wersje przeznaczone specjalnie dla komputerów o większej pamięci. Przetestowałem ponad dwadzieścia takich programów i stwierdziłem ich bezbłędne działanie z komputerem posiadającym rozszerzenie pamięci TOMS. Tylko w jednym przypadku wystąpiły pewne problemy — częściowo wadliwe było działanie programu „Atari Planetarium". Program ten wymaga niemal stuprocentowej zgodności komputera ze 130XE. W tym komputerze rejestr PORTB jest bowiem wykorzystywany dodatkowo do odrębnego sterowania dostępem do pamięci przez procesory 6502 i ANTIC. Opisywane zjawisko nie jest jednak wadą rozszerzenia, lecz kosztem jego instalacji. Nie jest bowiem możliwe powiększenie pamięci komputera ponad 128 KB bez takiej zmiany działania bitów rejestru PORTB, która zapewni stuprocentową zgodność ze 130XE.

Niemal każda polska firma, która zajmuje się Atari, ma w swojej ofercie powiększanie pamięci tych komputerów. Są to rozszerzenia działające na różnych zasadach i o różnym stopniu zgodności z istniejącym oprogramowaniem. Wśród znanych mi rozwiązań TOMS 260XE zajmuje jedno z pierwszych miejsc i mogę z czystym sumieniem polecić jego instalację wszystkim użytkownikom, którzy chcą powiększyć możliwości swojego sprzętu.

*Marek Zachar*

**Producent: TOMS:**
Tadeusz Okoń, Warszawa, ul. Lachmana 4/44, tel. 641–54–29
Marek Stolarski, Warszawa, ul. Kazury 13/26, tel. 46–01–02

***program testujący wielkość pamięci RAM komputerów Atari***

```
YY   10 GRAPHICS 0:DIM A$(6)
ZZ   20 ? :? "   test RAM":? :? " chwileczk
     e..."
SO   21 A=54017:B=32000:C=0:S=253
BG   22 POKE A,S
PA   23 POKE B,PEEK(A)
DU   24 POKE A,PEEK(A)-112
IJ   30 FOR X=1 TO 3
OX   31 POKE B,PEEK(A)
IF   32 FOR W=1 TO 3
FW   33 POKE A,PEEK(A)-4
PD   34 POKE B,PEEK(A)
NO   40 NEXT W
SJ   41 POKE A,PEEK(A)+44
OB   42 NEXT X
BK   43 POKE A,S
DW   44 POKE A,PEEK(A)-112
IU   45 FOR X=1 TO 3
CE   50 IF PEEK(A)<>PEEK(B) THEN GOSUB 1000
IE   60 FOR W=1 TO 3
FV   61 POKE A,PEEK(A)-4
CG   70 IF PEEK(A)<>PEEK(B) THEN GOSUB 1000
NS   80 NEXT W
SN   81 POKE A,PEEK(A)+44
OF   82 NEXT X
BO   83 POKE A,S
DD   84 IF PEEK(B)=193 THEN GOSUB 1000
MM   89 ? :? " test wykonany"
HN   90 WYNIK=1024*(256-16*C)
TV   91 ? :? "ten komputer posiada":? WYNIK
     ;:POSITION 8,8+C:? " bajty  pamieci op
     eracyjnej"
PX   100 A$=STR$(WYNIK)
CB   101 Z=VAL(A$(LEN(A$)))
KF   110 IF Z<2 OR Z>4 THEN POSITION 13,8+C
     :? "ow";
OQ   990 END
FJ   1000 P=PEEK(A)
MU   1010 ? "nieczynny obszar ";P;" - ";
HD   1020 C=C+1
QY   1030 IF P=193 THEN ? "BANK 0 130XE"
UD   1040 IF P=197 THEN ? "BANK 1 130XE"
OP   1050 IF P=201 THEN ? "BANK 2 130XE"
RU   1060 IF P=205 THEN ? "BANK 3 130XE"
MY   1070 IF P=129 THEN ? "BANK 10 260XE"
LQ   1080 IF P=133 THEN ? "BANK 11 260XE"
OW   1090 IF P=137 THEN ? "BANK 12 260XE"
MM   1100 IF P=141 THEN ? "BANK 13 260XE"
OQ   1110 IF P=161 THEN ? "BANK 14 260XE"
RW   1120 IF P=165 THEN ? "BANK 15 260XE"
VC   1130 IF P=169 THEN ? "BANK 16 260XE"
TU   1140 IF P=173 THEN ? "BANK 17 260XE"
RO   1150 IF P=253 THEN ? CHR$(30);CHR$(30)
     ;CHR$(30);CHR$(30);CHR$(30);CHR$(30);"
     193 - BANK 0 130XE"
AJ   1500 RETURN
```

# — NIE

**— Jak został Pan biznesmenem?**

— Osiemnaście lat temu — mając 23 lata — wyjechałem do Kanady, a potem do Stanów Zjednoczonych. Początkowo pracowałem w Dolinie Krzemowej jako konsultant komputerowy. Realizowaliśmy tam wiele różnych projektów, między innymi komputeryzowaliśmy fabrykę piwa w Milwaukee i zakłady General Motors. Moja rodzina posiada w Polsce firmę Kris, która została założona 25 lat temu, istnieje do dzisiaj i produkuje kosmetyki. Stąd znałem się trochę na prowadzeniu interesów. Będąc w Polsce zauważyłem, iż w dziedzinie, w której znalazłem pracę w Stanach, jest w Polsce dużo do zrobienia. Rynek komputerowy dopiero zaczynał się tworzyć. W 1983 roku założyłem w Stanach Zjednoczonych firmę LDW, Inc. i również w tym samy roku powstało oparte na LDW Przedsiębiorstwo Zagraniczne w Polsce „Karen". Zaczynaliśmy od produkcji oprogramowania na rynek amerykański.

**— Gdzie się mieściła ta firma?**

— Najpierw było to mieszkanie M-3 na Krochmalnej. Z czasem firma się powiększała, przybywało programistów i zaczęło brakować miejsca. Przenieśliśmy się na ulicę Podstarościch, później wynajęliśmy także pomieszczenia na Obrońców, a wreszcie doczekaliśmy się siedziby w Sulejówku.

**— Jak w tym czasie firma się rozwijała?**

— Zaczynaliśmy od produkcji gier, które LDW sprzedawała w Stanach Zjednoczonych, dając tym samym pracę licznym programistom, dla których nie było wówczas w Polsce zatrudnienia. Już wkrótce Polska zaczęła się komputeryzować. Jako jedni z pierwszych zaczęliśmy sprowadzać części z Tajwanu, montowaliśmy z nich komputery i sprzedawaliśmy w Polsce pod nazwą Quasar. Zainteresowanie komputerami w Polsce było olbrzymie, lecz urządzenia zgodne z IBM były zbyt drogie dla przeciętnego człowieka — prywatna osoba nie mogła sobie pozwolić na 10–20 mln złotych wówczas, gdy zarabiało się 50 tysięcy. Chcąc rozszerzyć rynek komputerowy LDW nawiązało współpracę z amerykańską firmą Atari. Jako wyłączny dystrybutor tej firmy na Polskę LDW zaczęło sprzedawać komputery Atari w sieci Pewexu. „Karen" stała się zaś zapleczem serwisowym tej sprzedaży. Dzięki dużemu zainteresowaniu odbiorców i masowej sprzedaży następował szybki rozwój przedsiębiorstwa. Obecnie LDW. Inc. zatrudnia około 30 osób w Stanach Zjednoczonych, a liczba pracowników PZ „Karen" dochodzi do 140.

**— A California Access i California Dreams?**

— Nie są to odrębne firmy, ale znaki firmowe zarejestrowane na rzecz Logical Design Works, Inc.

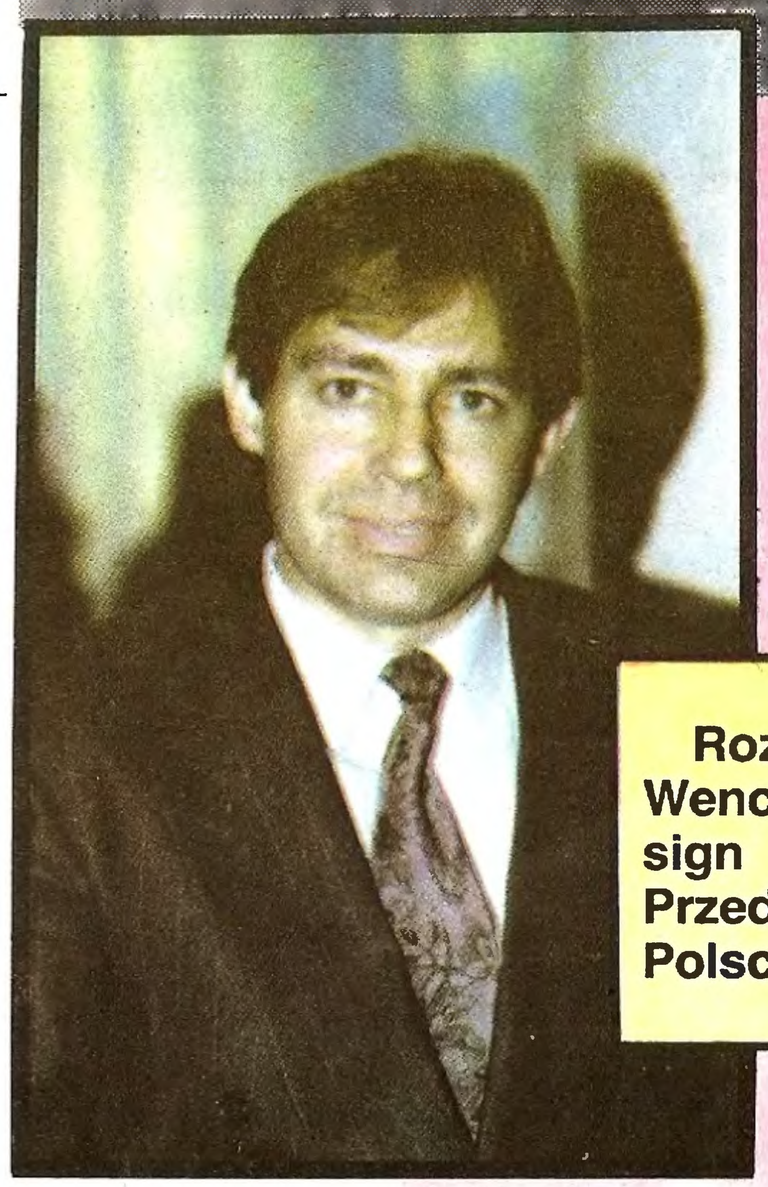
**— Ile komputerów Atari zostało sprzedanych w Polsce poprzez Pewex?**

— Około 250–270 tysięcy „małych" Atari. Obecnie około 20% światowej produkcji 8-bitowych komputerów Atari trafia do Polski, w sumie nasze kontrakty z Atari to około 1% produkcji tego koncernu.

**— Czy sprzedaż komputerów Atari w Polsce będzie kontynuowana w przyszłości?**

— Jak najbardziej. Na początku przyszłego roku będzie realizowane kolejne duże zamówienie. Sprowadzimy do Polski 60–70 tysięcy „małych" Atari. Komputerów serii ST dotychczas sprowadzaliśmy niewielką liczbę. Dopiero niedawno został podpisany kontrakt na dostawę około 10 tysięcy 520STFM. Liczymy, że dzięki naszej ofercie komputerowej w nowo otwieranym domu handlowym

# TYLKO ATARI



**Rozmowa z Lucjanem Danielem Wenclem, właścicielem Logical Design Works Inc., Kalifornia, USA i Przedsiębiorstwa Zagranicznego w Polsce „Karen".**

Pewexu na Złotej znajdzie się miejsce na specjalistyczne stoisko komputerowe pod nazwą „Pewex Computer — California Access".

**— Czy zamierza Pan również sprzedawać komputery zgodne z IBM PC?**

— Dotychczas wprowadziliśmy do Pewexu około 1500 komputerów tego standardu, monotowanych w Karenie. Nie był to sukces — w porównaniu do Atari sprzedają się one słabo. Prawdopodobnie powodują to przyzwyczajenia klientów, którzy szukają w Pewexie wszystkiego, tylko nie profesjonalnych urządzeń komputerowych. Mając na uwadze specyfikę Pewexu sprzedającego towary luksusowe na dużą skalę, zamierzamy wprowadzić do sprzedaży w sieci sklepów tego przedsiębiorstwa domowy komputer PC o standardowej pamięci, z wyjściem na telewizor. Produkcja tego modelu ruszy niebawem, a w sklepach powinien się on pojawić w maju, czerwcu 1991 roku. Cena będzie wynosiła około 400 $.

**— Czy Polska jest jedynym tak dużym Pańskim odbiorcą komputerów?**

— Nie. W zachodniej Europie też trochę sprzedajemy, co prawda nie tak duże ilości. Mniejsze ilości komputerów LDW. Inc. sprzedaje również w Ameryce Południowej, sporo w krajach arabskich. Na przykład, w Arabii Saudyjskiej sprzedajemy ogromne ilości. Trochę jeszcze w Rosji, na Węgrzech, w Jugosławii, w Czechosłowacji, ale w Polsce sprzedajemy najwięcej spośród krajów Europy Wschodniej.

**— Sądzimy, że komputery Atari są zbyt drogie. 65XE z magnetofonem kosztował w ubiegłym roku 170 $, a Spectravideo ze stacją dysków można było kupić już za 1 milion złotych.**

— Komputery Spectravideo, Spectrum, czy podobne można było kupić bardzo tanio. Na przykład, Commodore C-16 sprzedawano w hurcie po 10 $. Dlaczego? Nie ma na nie oprogramowania. Commodore zrobiło duży błąd i pozbywało się tego za pół darmo.

**— Więc Pan uważa, że sprzęt jest tak dobry, że ceny nie mogą być niższe?**

— Nie chodzi o to, że dobry. Na całym świecie obniża się cenę, gdy towar leży w magazynie, nie ma szansy sprzedania — jest to „cegła". Wtedy mamy wybór — sprzedać produkt za pół darmo lub wyrzucić do śmieci. I to jest często bardzo dobry wariant, nawet lepszy niż sprzedaż, ponieważ nie psuje marki. Często robi się właśnie tak albo sprzedaje się do innego kraju. Bo jeżeli się psuje markę, robi wyprzedaże, to kończy się tak jak amerykańskie firmy samochodowe. Wszyscy czekają na wyprzedaż, nikt nie kupuje po tej prawdziwej cenie i nic nie można sprzedać.

**— Ale komputery Atari są już starą konstrukcją.**

— Taki sprzęt wycofuje się z produkcji. Jeżeli ktoś zrobił błąd, wyprodukował 100 tysięcy, leży te 100 tysięcy w magazynach i trzeba się ich jakoś pozbyć, to obniża się cenę o połowę i rzuca się na rynek albo wywozi się na wysypisko i zakopuje, żeby nie psuć marki. Natomiast Atari wciąż znajdują nabywców. Jeżeli pan widzi Atari w Pewexie, to jest to egzemplarz wyprodukowany dwa miesiące temu. Niemożliwe jest robienie z tego wyprzedaży.

**— Mimo to uważamy, że jest zbyt drogi.**

— Marża LDW i wliczony w nią serwis Karen to tylko niewielki ułamek w cenie. Marża Pewexu jest również niewygórowana. Trzeba do tego jednak doliczyć 20% podatku obrotowego i razem wychodzi cena, jaką znamy ze sklepu. Aby ją obniżyć, trzeba by zmniejszyć płace robotnikom, na co nie zgodzą się ani związki zawodowe, ani dyrekcja Atari. Jednak dzięki takim cenom Atari mają zapewniony na długi jeszcze czas i części, i serwis, i nowe oprogramowanie. A nie można tego powiedzieć o wcześniej wymienianych firmach.

Technologia Atari jest taka, że — aby obniżyć ceny — trzeba by wprowadzić nowy układ scalony; który zastąpiłby kilka starych. Zrobienie nowego układu kosztuje około miliona dolarów. Jest to inwestycja zupełnie nieopłacalna. Dlatego my — LDW, Inc. — chcemy wprowadzić na polski rynek cenowo zbliżony do Atari komputer zgodny z IBM.

**— Czy nie można sprzedawać Atari bez pośrednictwa Pewexu, aby zmniejszyć cenę chociaż o jego marżę?**

— Nie sądzę, Pewex dodaje dużo wartości. Jest to sieć sklepów, gwarancja marki wyrobu, zapewnienie jego serwisu. Pewex jest znany, co daje nabywcy pewność. Jeżeli ktoś kupuje komputer, to do Pewexu ma większe zaufanie niż do prywatnej osoby. Nawet przy wyeliminowaniu Pewexu nie można obniżyć ceny o 20–30%, a tylko taka zmiana byłaby zauważalna dla klientów.

**— To są interesy firmy, a nie interes klienta.**

— Zadaniem firmy nie jest sprzedawanie jak najtaniej, ale sprzedawanie dobrego sprzętu i solidna obsługa klientów, aby nie stracić marki.

**— A co może Pan powiedzieć o ST?**

— W przyszłym roku Atari ST będą sprzedawane w Pewexie po nowych, niższych cenach. Chcemy się skoncentrować na STFM.

**— A nowe STE?**

— My jesteśmy handlowcami, chcemy się kierować motywacją handlową, a nie technologiczną. STE jest technologicznie lepszy — ma blitter, grafikę wektorową itd. To jest technologia. Teraz popatrzmy na handel. Wszystkie programy są pisane pod STFM, czyli „pod najmniejszy wspólny mianownik". Wprowadzanie na rynek STE, który jest droższy, a którego możliwości nie są wykorzystane w programach, nie ma sensu. Może się to wydawać paradoksalne, ale tak jest.

**— W zasadzie do podstawowego zestawu Atari — szczególnie „małego" — firma niemal nic nie oferuje. Dlaczego taka mała aktywność?**

— Trudno wprowadzać na rynek produkt, który mógłby liczyć na mniejszą liczbę odbiorców niż 5000. W taką sprzedaż mogą się angażować małe firmy, o niewielkich kosztach własnych, które będą prosperować zarabiając na pojedynczym kliencie. Jeżeli znajdą się takie firmy, to chętnie udostępnimy im technologię. Na przykład, opracowany przez nas interfejs do odbioru teletekstu ze zwykłego telewizora poprzez Atari XL/XE.

**— Co pozostanie dla Karen?**

— Firma Karen ma ograniczone możliwości przerobowe. Atari jest tylko jedną z wielu dziedzin jej zainteresowań. Zapewniamy i będziemy zapewniać serwis Atari. Jednak drobniejsze rzeczy powinny robić inne firmy. Nie chcemy zjeść wszystkich obiadów — nie chcemy wszystkiego robić sami. Niech inni też zarobią. Karen to nie Atari, my robimy jeszcze wiele innych rzeczy. Firma jest m.in. autoryzowanym dystrybutorem komputerów typu SUN, dostarcza systemy komputerowe „pod klucz" *(turn key systems)*, oprogramowanie dla dużych i średnich przedsiębiorstw.

**— Co jeszcze firma zaoferuje masowym odbiorcom w Polsce?**

— Wprowadzamy nowość dla takich ludzi, którzy nie mają pieniędzy, a chcieliby się zabawić. Nie jest to komputer, lecz konsola do gier oparta na Atari 2600. CA-160 wkrótce będzie dostępna w Pewexie za 59 dolarów. Wymaga ona tylko podłączenia do telewizora i już można grać. Konsola ta różni się od Atari 2600 tym, że ma wbudowane 160 gier. Można również wkładać dodatkowe moduły z grami. Zakładamy jednak, iż przy tej liczbie gier, możliwość ta będzie wykorzystywana bardzo rzadko.

**— Karen zajmuje się produkcją oprogramowania, ale robi to tylko na Zachód. Dlaczego nie piszecie oprogramowania edukacyjnego dla Polski?**

— Nie jest to opłacalne, z powodu kwitnącego tu piractwa. Trudno robić to, co w kilka dni po sprzedaży pierwszego egzemplarza wszyscy będą mieli za darmo (i to bez wiedzy i zgody autorów). Problem polega więc na tym, aby znalazł się ktoś, kto zapłaci raz pokrywając koszty włożone w wyprodukowanie takiego oprogramowania. Wstępne rozmowy z Ministerstwem Edukacji zostały już przeprowadzone, ale na efekty trzeba będzie jeszcze poczekać.

**— Dziękujemy za rozmowę i życzymy dalszych sukcesów handlowych i produkcyjnych.**

rozmawiali
*Jarosław Młodzki*
*Wojciech Zientara*

# JAK KODOWAĆ?

**Zgodnie z obietnicą daną w poprzednim odcinku zaczniemy teraz materializować dotychczasowe ustalenia teoretyczne. Czynność ta polega na zakodowaniu stworzonego algorytmu w postaci programu w jednym z języków programowania.**

Pierwszym problemem do rozstrzygnięcia jest wybór języka. W większości przypadków zależy to jedynie od upodobań programisty. Tylko czasami występują ograniczenia warunkujące wybór konkretnego języka. Najczęściej chodzi tu o uzyskanie odpowiedniej szybkości działania programu, choć ten problem można ominąć, na przykład kompilując program napisany w Atari Basic. Tę uniwersalność języków pokażę podczas kodowania naszej gry poprzez równoległe pisanie programów w Atari Basic i Action!

Program w Atari Basic jest automatycznie porządkowany przez komputer według numerów wprowadzanych wierszy. Niezależnie od kolejności wpisywania wierszy uzyskuje się więc zawsze taki sam program. Inaczej jest w Action!, który wymaga właściwej kolejności zapisu. Każda nazwa zmiennej, funkcji i procedury musi być zdefiniowana w programie zanim zostanie po raz pierwszy użyta. Przy kodowaniu programu w Action! należy więc zwracać uwagę na dopisywanie nowych części we właściwych miejscach.

Teraz już możemy przejść do meritum. Przypomnij sobie podstawowy schemat blokowy zbudowany w „Moim Atari 2". Zawiera on zasadnicze części gry. Uzupełniając je komentarzem informującym o tytule i autorze oraz zapisując to w Action! uzyskasz następujący efekt:

```
;***  KONWOJ  ***
;Wojciech Zientara
;(c) 1991, Sp. Bajtek

BYTE q

PROC Konwoj()
  Tytul()
  DO
    Inicjowanie()
    PutE()
    Print("    CZY CHCESZ INSTRUKCJE? (T/N)")
    IF Decyzja()=84 THEN Instrukcja() FI
    Gra()
    Wynik(q)
    PrintE("          Jeszcze raz? (T/N)")
  UNTIL Decyzja()=78 OD
  Graphics(0)
  Close(3)
RETURN
```

Wyraźnie widoczne są tu podstawowe elementy naszego programu: tytuł oraz wykonywane w pętli inicjowanie, gra właściwa i zakończenie. Wątpliwości mogą budzić tylko deklaracja BYTE na początku oraz procedury Graphics() i Close() na końcu programu. BYTE jest definicją zmiennej 'q', która określa wynik gry w procedurze Wynik(). Procedury Graphics() i Close() tworzą zakończenie całego programu. Jest ono bardzo krótkie, więc można zrezygnować z wyodrębniania go w samodzielną procedurę Koniec(). Ten sam fragment programu zrealizowany w Basicu będzie miał odmienny wygląd:

```
BV   0 REM ***  KONWOJ  ***
EE   1 REM Wojciech Zientara
DU   2 REM (c) 1991, Sp. Bajtek
NI   3 REM
JE   10 GOTO 5000
KU   4997 REM
CU   4998 REM *** Konwoj ***
LC   4999 REM
ZM   5000 GOSUB 2500
RX   5010 GOSUB 3000:? :? "    CZY CHCESZ I
     NSTRUKCJE? (T/N)";
SY   5020 GOSUB 100:IF KY=84 THEN GOSUB 400
     0
WM   5030 GOSUB 1500:GOSUB 2000:? "
      Jeszcze raz? (T/N)"
GU   5040 GOSUB 100:IF KY=84 THEN 5010
TA   5050 GRAPHICS 0:CLOSE #3:END
```

Najważniejszą różnicą, którą można zauważyć na pierwszy rzut oka, jest brak nazw procedur zastąpionych numerami wierszy rozpoczynających procedury. Powoduje to znacznie mniejszą czytelność programu. Można to poprawić stosując zmienne o odpowiednich nazwach (np. TYTUL=2500:GOSUB TYTUL), lecz wydłuża i komplikuje to program. Trzeba też pamiętać, że liczba zmiennych, które można użyć w programie, jest ograniczona do 128.

Pewne zdziwienie może budzić instrukcja GOTO użyta w wierszu 10 powyższego programu. Rzeczywiście obecnie jest ona zbędna, lecz podczas rozbudowy programu będziemy przecież dodawać kolejne części, które muszą być ominięte przy uruchamianiu. Oczywiście, można zmienić kolejność wierszy i umieścić tę część programu na jego początku. Napotkanie podczas wykonywania programu w Basicu instrukcji wskazującej numer wiersza powoduje przeszukiwanie treści programu od jego początku. Należy więc najczęściej wykonywane fragmenty umieszczać na początku programu i dlatego u nas początek znalazł się na końcu.

Inne są natomiast przyczyny umieszczenia początku na końcu programu w Action!. Pierwsza jest oczywista i wynika wprost z żądania, aby wszystkie użyte w niej nazwy były wcześniej zadeklarowane. Drugą przyczyną jest sposób wykonywania programu w Action! — jako pierwsza jest zawsze wykonywana ostatnia procedura programu.

Pokazane fragmenty programu różnią się jeszcze jednym rozwiązaniem — sposobem realizacji procedury umożliwiającej podjęcie decyzji przez gracza. W Action! została ona wykonana jako funkcja zwracająca kod wybranego klawisza. Odczyt klawisza przebiega przy tym w pętli, która jest przerywana dopiero po naciśnięciu <T> (kod 84) lub <N> (kod 78). Wygląda to następująco:

```
BYTE FUNC Decyzja()
 BYTE key
  DO
    key=GetD(3)
  UNTIL key=78 OR key=84 OD
RETURN(key)
```

Ponieważ Atari Basic nie pozwala na definiowanie funkcji, konieczne było zastosowanie procedury przypisującej kod odpowiedzi zmiennej 'KY', a dopiero później sprawdzanie wartości tej zmiennej. Zabezpieczenie przed naciśnięciem niewłaściwego klawisza jest w tym przypadku realizowane przez zastosowanie instrukcji ON/GOTO zamiast pętli DO/OD z warunkiem UNTIL.

```
BW   98 REM
LK   99 REM *** Decyzja ***
SX   100 GET #3,KY:ON KY<>78 AND KY<>84 GOT
     O 100:RETURN
```

Inna struktura programu zmusiła także do użycia innego warunku jego zakończenia. W Action! program jest bowiem przerywany, gdy gracz odpowie NIE, w Basicu zaś jest on kontynuowany gdy odpowiedź brzmi TAK. Proszę zwrócić uwagę, że pod względem logicznym oba powyższe stwierdzenia oznaczają dokładnie to samo, lecz ich realizacja programowa jest zupełnie inna. Jest to doskonały przykład ilustrujący znaczenie wyboru odpowiedniego sformułowania warunku przy układaniu programu.

Czy taki niekompletny program można już uruchomić, aby skontrolować poprawność jego działania? Oczywiście nie. Programu w Action! nie da się nawet skompilować, a w Basicu natychmiast otrzymamy komunikat "ERROR 12 AT LINE 5000".

Jeżeli jednak koniecznie chcesz uruchomić taki niepełny program, to można to zrealizować po dopisaniu kilku instrukcji. W obu językach trzeba w tym celu określić wszystkie występujące w programie procedury. W Action! wykorzystuje się do tego procedury, które nic nie robią, na przykład:

```
PROC Tytul ()
RETURN
```

Tę samą funkcję pełnią w Basicu wiersze zawierające tylko instrukcję RETURN. Pokazanej wyżej procedurze Tytul() będzie odpowiadał w Basicu wiersz:

```
2500 RETURN
```

Po takim zdefiniowaniu wszystkich procedur uzyskamy działający program, ale taki, który choć bezbłędnie działa, to jednak nic nie robi.

Kodowanie dalszych części programu — ale działających i wykonujących założone zadania — będzie kontynuowane w następnych odcinkach.

*Wojciech Zientara*

Następny numer:
MOJE Atari 5
w kioskach
„Ruchu"
już od
6 czerwca.

# TYLKO DLA ZAAWANSOWANYCH (1)

**„Małe" Atari, mimo sędziwego wieku — to przecież ponad 10 lat — w dalszym ciągu kryje w swym wnętrzu wiele tajemnic i nie wykorzystanych dotychczas możliwości. W niniejszym cyklu postaram się zaprezentować kilka z nich.**

Z konieczności sedno większości programów tkwi w procedurach maszynowych, dość obszernie korzystających z systemu. Do pełnego ich zrozumienia — poza narzędziami, jak asembler i disasembler, mapa pamięci i systemu — potrzebna jest znajomość języka maszynowego oraz dużo cierpliwości i zapału. Osoby nie zainteresowane programowaniem na takim poziomie mogą jednak również wykorzystywać przedstawiane efekty w swoich własnych dziełach.

Na początek o zagadnieniu, które było już poruszane na łamach „Bajtka" — o przerwaniu synchronizacji pionowej. Jego obsługa odbywa się kilkustopniowo. Najpierw wykonywana jest procedura systemowa SYSVBL. Realizuje ona — między innymi — zmiany zawartości liczników (TIMER), czytanie klawiatury, sprawdzanie pozycji joysticka, odświeżanie rejestrów-cieni — słowem wszystkie czynności, bez których niemożliwe byłoby prawidłowe funkcjonowanie komputera. Potem z tej procedury następują skoki do obsługi procedur liczników i przerwania opóźnionego VBLKD. Korzystanie z nich jest bardzo wygodne, ale ma jedną wadę — nie są one realizowane w czasie operacji wejścia/wyjścia, kiedy ustawiony jest znacznik CRITIC ($42).

Niemal każdy spotkał się z płynącym w czasie odczytu napisem czy innymi tego typu efektami. Aby je uzyskać, konieczna jest ingerencja w obsługę właściwego przerwania synchronizacji pionowej. Najprościej można tego dokonać ustawiając wektor przerwania VBLKI na adres naszej procedury, na której końcu następuje skok do SYSVBL. Tak też jest to rozwiązane we wszystkich prezentowanych dziś programach. Zmiany wektora dokonuje się za pomocą procedury SETVBLV — przed jej wywołaniem trzeba umieścić w akumulatorze wartość 6 — zmiana VVBLKI (7 — VVBLKD — przerwanie opóźnione), w rejestrach X i Y zaś starszy i młodszy bajt adresu naszej procedury.

Program 1 powoduje zmienianie się odgłosów podczas operacji wejścia/wyjścia — rekordy są wczytywane raz cicho, raz głośno, choć zmiany te nie są regularne. Procedura obsługi przerwania okresowo zmienia zawartość komórki IOSNDEN ($41) — zero powoduje trzaski, inne wartości normalny dźwięk. Listing 2 jest programem źródłowym procedury użytej w tym programie.

Program 2 daje podczas transmisji danych efekt bulgotania w głośniku. O ile w trakcie odczytu jest on nieszkodliwy, o tyle podczas zapisu może się okazać tragiczny w skutkach. Proszę spróbować, ale raczej bez ważnych danych w pamięci. Od strony formalnej działanie programu (tekst źródłowy w listingu 4) polega na przepisywaniu pseudolosowych wartości z rejestru RANDOM ($D20A) do AUDF1 i AUDF2 ($D200 i $D202). Zapis do pozostałych rejestrów dźwięku uniemożliwia przeprowadzenie jakichkolwiek operacji wejścia/wyjścia. Rejestry te służą bowiem nie tylko dc określenia częstotliwości dźwięku, ale również do definiowania prędkości transmisji (działają jako asynchroniczne liczniki wyjściowe).

Program 3 pozwala na uzyskanie płynącego napisu. Dodatkowo połączyłem standardowy przesuw poziomy z pionowym, uzyskując w ten sposób „falujący napis". Operacje wejścia/wyjścia nie zatrzymują go. Listing 6 zawiera tekst źródłowy tego programu.

W następnym odcinku jeszcze raz o polskich znakach.

*Andrzej Zalewski*

**1**

```
TR   0 REM TYLKO DLA ZAAWANSOWANYCH (1)
KT   1 REM PROGRAM 1
LU   2 REM Andrzej Zalewski
DV   3 REM (c) 1991, Sp. Bajtek
NH   10 POKE 82,0:? CHR$(125):FOR N=1536 TO
      1578
FT   20 READ A:S=S+A:POKE N,A:NEXT N
DI   30 IF S<>4880 THEN ? "Popraw dane 'DAT
     A'":END
KS   40 X=USR(1536):? "Wczytaj dowolny  pro
     gram w Basicu,    najlepiej jak najdlu
     zszy.":END
UF   50 DATA 104,169,0,133,65,160,14,162,6,
     169,6,76,92,228,173,255,5,240,6,206,25
     5,5,76,40,6
YL   60 DATA 169,50,141,255,5,165,65,208,4,
     230,65,208,2.198,65.76,95.228
```

**2**

```
0100 ; Trzaski+przerwanie NMI
0110 ; Andrzej Zalewski
0120 ; (c) 1991, Sp. Bajtek
0130 ;
0140 LICZNIK = $05FF
0150 IOSNDEN = $41
0160 SETVBLV = $E45C
0170 SYSVBL = $E45F
0180 ;
0190      *=  $0600
0200 ;
0210 ; Zmiana wektora VBLKIVKT
0220      PLA
0230      LDA #0
0240      STA IOSNDEN
0250      LDY # <NMI   ;mlodszy i
0260      LDX # >NMI   ;starszy bajt
0270      LDA #$06     ;nowej procedury
0280      JMP SETVBLV
0290 ;
0300 ; procedura przerwania
0310 NMI LDA LICZNIK
0320      BEQ ZMIANA
0330      DEC LICZNIK
0340      JMP EX
0350 ZMIANA LDA #50  ; odliczanie od
0360      STA LICZNIK ; poczatku
0370      LDA IOSNDEN
0380      BNE ZERO
0390      INC IOSNDEN ; normalnie
0400      BNE EX
0410 ZERO DEC IOSNDEN ;wyciszenie
0420 EX  JMP SYSVBL
```

**3**

```
TR   0 REM TYLKO DLA ZAAWANSOWANYCH (1)
LI   1 REM PROGRAM 2
LU   2 REM Andrzej Zalewski
DV   3 REM (c) 1991, Sp. Bajtek
MF   10 FOR N=1536 TO 1569:READ A:POKE N,A:
     NEXT N:X=USR(1536):END
QY   20 DATA 104,160,10,162,6,169,6,76,92,2
     28,165,66,240,17,173,10,210,141,0,210,
     141,2,210
BW   30 DATA 169,239,141,1,210,141,3,210,76
     ,95,228
```

**4**

```
0100 ;Bulgotanie - VBLKI
0110 ;
0120 CRITIC = $42
0130 AUDF1 = $D200
0140 AUDC1 = $D201
0150 AUDF2 = $D202
0160 AUDC2 = $D203
0170 RANDOM = $D20A
0180 SETVBLV = $E45C
0190 SYSVBL = $E45F
0200 ;
0210      *=  $0600
0220 ;
0230      PLA
0240      LDY # <NMI
0250      LDX # >NMI
0260      LDA #$06
0270      JMP SETVBLV
0280 ;
0290 NMI  LDA CRITIC
0300      BEQ EX
0310      LDA RANDOM
0320      STA AUDF1
0330      STA AUDF2
0340      LDA #239
0350      STA AUDC1
0360      STA AUDC2
0370 EX   JMP SYSVBL
```

**5**

```
TR   0 REM TYLKO DLA ZAAWANSOWANYCH (1)
LX   1 REM PROGRAM 3
LU   2 REM Andrzej Zalewski
DV   3 REM (c) 1991, Sp. Bajtek
LL   10 POKE 82,0:POKE 83,39:? CHR$(125);:R
     EM W wierszach 20-40 umieszczamy swoj
     napis
HE   20 ? "                    ***   Andrzej
     ZAlEwSki przedstawia falujaco-plynacy
     NAPIS   ***                  (c) 1991 ";
XF   30 ? "SilenT SoftwarE Inc.   .........
     ........ prOGram opracowany sPecjalNIE
      dla cyklu ...........";
TN   40 ? "TYLKO dla ZAAWANSOWANYCH .......
     ..........   "
PT   50 FOR N=0 TO 255:POKE 24576+N,PEEK(40
     000+N):NEXT N
GQ   60 FOR N=1536 TO 1670:READ A:POKE N,A:
     NEXT N:X=USR(1536):END
EH   100 DATA 104,169,100,141,48,2,169,6,14
     1,49,2,169,0,141,138,6,141,137,6,169,6
     ,162,6,160,28
YZ   110 DATA 76,92,228,174,136,6,240,9,202
     ,142,136,6,142,4,212,208,42,174,103,6,
     224,255,240,3,232,208,2,162,0,142
GX   120 DATA 103,6,169,8,141,4,212,141,136
     ,6,174,137,6,173,138,6,208,14,232,224,
     7,240,17,142,5,212,142,137,6,76
FT   130 DATA 95,228,202,208,244,238,138,6,
     240,239,206,138,6,208,234,6,38,118,152
     ,96,64,66,64,156,2,2,2,2,2,2
AB   140 DATA 2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
     ,2,2,65,100,6
```

**6**

```
0100 ;Falujacy napis
0110 ;
0120 DLPTRS = $0230
0130 HSCROL = $D404
0140 VSCROL = $D405
0150 SETVBLV = $E45C
0160 SYSVBL = $E45F
0170 ;
0180 HLICZ = TOP+1
0190 VLICZ = TOP+2
0200 FLAG =  TOP+3
0210 ;
0220 MIN =    $00
0230 MAX =    $FF
0240 ;
0250      *=  $0600
0260 ;
0270      PLA
0280      LDA # <DL
0290      STA DLPTRS
0300      LDA # >DL
0310      STA DLPTRS+1
0320      LDA #0
0330      STA FLAG
0340      STA VLICZ
0350      LDA #6
0360      LDX # >NMI
0370      LDY # <NMI
0380      JMP SETVBLV
0390 ;
0400 ;procedura przerwania
0410 NMI LDX HLICZ
0420      BEQ ZWANT
0430      DEX
0440      STX HLICZ
0450      STX HSCROL
0460      BNE EXIT
0470 ZWANT LDX ANT
0480      CPX #MAX
0490      BEQ ZERO
0500      INX
0510      BNE DALEJ
0520 ZERO LDX #MIN
0530 DALEJ STX ANT
0540      LDA #8
0550      STA HSCROL
0560      STA HLICZ
0570 ;
0580      LDX VLICZ
0590      LDA FLAG
0600      BNE ZMN
0610      INX
0620      CPX #7
0630      BEQ FLFF
0640 ST   STX VSCROL
0650      STX VLICZ
0660 EXIT JMP SYSVBL
0670 ;
0680 ZMN DEX
0690      BNE ST
0700 FLOO INC FLAG
0710      BEQ ST
0720 FLFF DEC FLAG
0730      BNE ST
0740 ;
0750 DL   .BYTE 6,38,118
0760 ANT  .BYTE 0,96,64,66,64,156
0770      .BYTE 2,2,2,2,2,2,2,2,2,2
0780      .BYTE 2,2,2,2,2,2,2,2,2,2
0790      .BYTE 2,2,2,65, <DL, >DL
0800 TOP
```

# OPERACJE PIERWOTNE WEJŚ

**Po ukazaniu się książki* opracowałem na jej podstawie realizację większości operacji pierwotnych w Kyan Pascalu, będących otoczką niezbędną do pracy z programami zawartymi we wspomnianej publikacji.**

Koncepcja operacji pierwotnych została tam dokładnie omówiona w Dodatku od str. 356. Przedstawiono również implementacje w innych dialektach Pascala. Stały się one dla autora wzorem dla własnych realizacji.

Różnią się one od pierwowzorów w następujących szczegółach:

1. w zbiorze GLOBDEF.IO (listing 1) zawarte są tylko trzy stałe „znakowe", przy realizacji większości programów zbiór ten należy uzupełnić;
2. zrezygnowałem z czystości koncepcji i deskryptory plików standardowych STDIN oraz STDOUT zdefiniowałem jako zmienne, a nie stałe; wersja ta ułatwi uzupełnienie otoczki o operacje przełączania plików;
3. nazwy operacji PUTC i GETC zastąpiłem nazwami PUTCHAR i GETCHAR;
4. procedura ERROR przejmuje kontrolę nad systemem, a w rezultacie kończy program po każdym wystąpieniu błędu wejścia/wyjścia;
5. operacje CREATE i OPEN zachowują się tak samo, jak opisane na stronach 92 i 109 w (1) operacje MUSTCREATE i MUSTOPEN;
6. w operacji CREATE zrezygnowałem z parametru MODE równego zawsze IOWRITE;
7. w operacjach CREATE, REMOVE i OPEN ze względu na długość parametru FNAME przekazuje się go przez zmienną;
8. nie zrealizowałem operacji GETARG i NARGS, chociaż przykład ze stron 391 i 393 w (1) przedstawiający realizację wczytywania dyrektyw wraz z możliwością przełączania plików jest bardzo zachęcający, a wymaga tylko rozbudowania procedury IOINIT oraz kilku dodatkowych deklaracji w zbiorze GLOBDEF.IO;
9. nie zrealizowałem operacji MESSAGE i SEEK.

Opracowana przeze mnie otoczka — składająca się z definicji stałych, typów i zmiennych zawartych w zbiorze GLOBDEF.IO oraz operacji IOERROR, CREATE, REMOVE, OPEN, CLOSE, PUTCF, GETCF, PUTCHAR, GETCHAR, PUTSTR, GETLINE, IOINT zawartych w zbiorze PRIMS.IO (listing 2) — powinna być uzupełniona o zbiór UTILITY zawierający inne użyteczne procedury i funkcje. Tak zbudowane stanowisko może być wykorzystywane w programach, których zarys przedstawiałby się następująco:

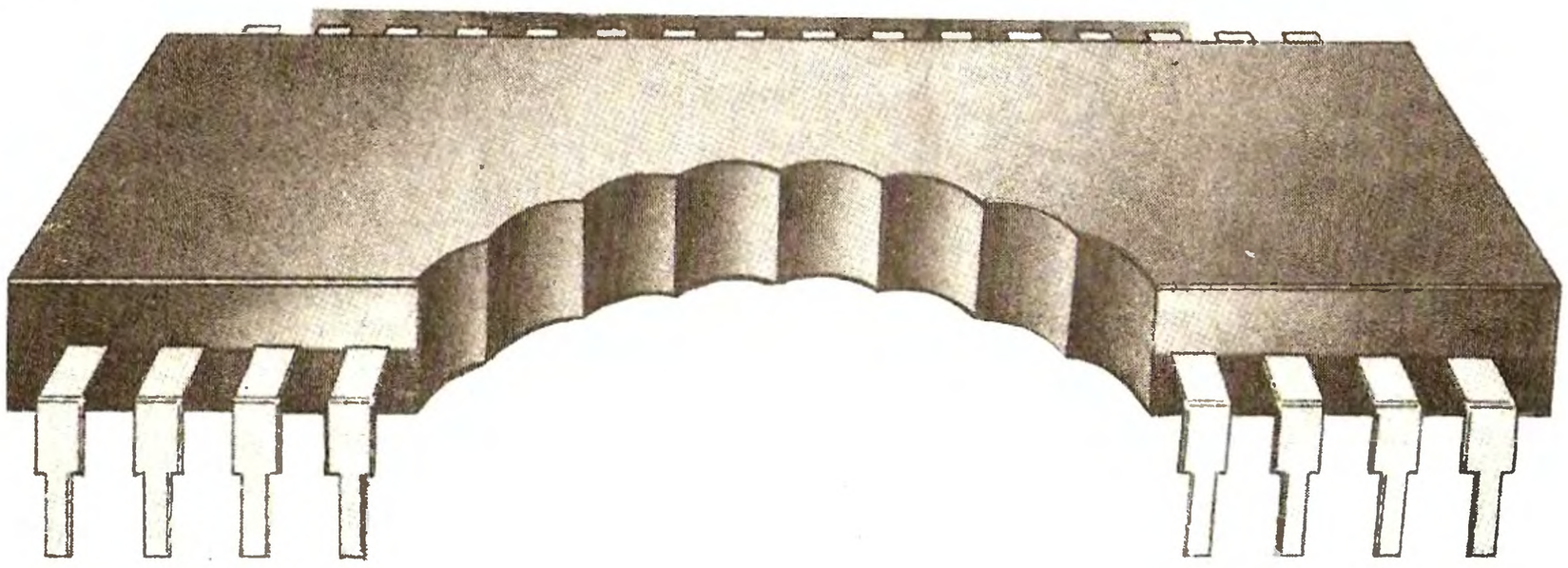
```
PROGRAM abc;

#I D1:GLOBDEF.IO
#I D1:PRIMS.IO
#I D1:UTILITY

#I D1:MAIN

    BEGIN
     ioinit;
     main
    END.
```

W programie tym procedura MAIN jest najbardziej zewnętrzną procedurą użytkownika.

W przyszłości otoczkę należałoby rozbudować o operację wczytywania dyrektyw i przełączania plików, zmienić działanie procedury ERROR, aby użytkownik zachowywał kontrolę nad awaryjnym zakończeniem programu, a nawet uzupełnić ją o mechanizmy będące namiastką kompilacji niezależnej programów zawartych w zbiorach PRIMS.IO oraz MAIN.

*Ryszard Wiech*

* Kernighan B., Plauger P. „Narzędzia programistyczne w Pascalu", WNT, Warszawa, 1989.



## LISTING 1

```
(*****************************************)
(******      "GLOBDEF.IO"           ******)
(* DEFINICJE STALYCH,TYPOW,ZMIENNYCH *)
(*    DO OPERACJI PIERWOTNYCH I/O    *)
(*          w KYAN Pascal            *)
(*****      Ryszard Wiech          ******)
(*      <c> 1990, Sp. Bajtek         *)
(*****************************************)

 CONST
  IOREAD=4;          (*zapis          *)
  IOWRITE=8;         (*odczyt         *)
  IOAPPEND=9;        (*dolaczanie     *)
  IOCORRECT=12;      (*zapis,odczyt   *)

  MAXSTR=128;

  ENDFILE=-1;        (* CONTROL 3 *)
  ENDSTR=0;
  NEWLINE=155;


 TYPE
  character=-1..255;
  filedesc=0..5;
  tryb=IOREAD..IOCORRECT;
  string=ARRAY[1..MAXSTR] OF character;

 VAR
    STDOUT:0..5;
    STDIN :1..5;

(*****************************************)
(****     KONIEC  GLOBDEF.IO         ****)
```

## LISTING 2

```
(*****************************************)
(******       " PRIMS.IO "         ******)
(** OPERACJE PIERWOTNE INPUT/OUTPUT **)
(*           w KYAN Pascal           *)
(*           Ryszard Wiech           *)
(*****    <c> 1990, Sp. Bajtek    *****)
(*****************************************)

(* Sluzy do sygnalizacji bledow I/O.  *)
(* Po wykonaniu kazdej operacji I/O   *)
(* sprawdzany jest status uzytego     *)
(* IOCB. Jesli wskazuje on wystapie-  *)
(* nie bledu, nastepuje przeskok do   *)
(* procedury "ioerror". Ta z kolei    *)
(* wypisuje numer bledu i skacze do   *)
(* obslugi przerwania towarzyszacego  *)
(* nacisnieciu klawisza BREAK.        *)
(* Powoduje to awaryjne zakonczenie   *)
(* programu.                          *)

PROCEDURE ioerror;

  FUNCTION kod:char;
    BEGIN
#a
        LDA IOE1
        LDY #3
        STA (SP),Y
#
    END;

  BEGIN
   BEGIN
#a
IOE1   DB 0
IOE2    CLD
#
   END;
   writeln('        blad: # ',ord(kod));
   BEGIN
#a
     JMP(566)
#
   END
  END;

(*************************************)
(* Tworzy na dysku zbior okreslony   *)
(* przez zmienna "fname" i zwiazuje  *)
(* z nim deskryptor pliku. Jesli plik*)
(* o podanej nazwie istnieje, to jest*)
(* on kasowany. Nowo utworzony plik  *)
(* ma dlugosc zero i ustawiony jest  *)
(* do zapisu. Funkcja zwraca wartosc *)
(* rowna deskryptorowi zwiazanemu z  *)
(* plikiem.                          *)

  FUNCTION create(VAR fname:string):fil
edesc;
    BEGIN
#a
        TXA
        PHA
        CLD
        LDX #16
IOC1    LDA #255
        CMP 832,X
        BEQ IOC2
        TXA
        CLC
    -   ADC #16
        TAX
        CMP #96
        BNE IOC1
        LDA #161
        STA IOE1
        PLA
        TAX
        JMP IOE2
IOC2    TXA
        PHA
        SEC
        LDA SP
        SBC #16
        STA SP
        LDA SP+1
        SBC #0
        STA SP+1
        LDY #21
        LDA (SP),Y
        STA 212
        LDY #22
        LDA (SP),Y
        STA 213
        LDX #0
        CLC
IOC3    TXA
        ASL A
        TAY
        LDA (212),Y
        PHA
        TXA
        ADC #3
        TAY
        PLA
        STA (SP),Y
        INX
        CPX #15
        BNE IOC3
        PLA
        TAX
        LDA #3
        STA 834,X
        CLC
        LDA SP
        ADC #3
        STA 836,X
        LDA SP+1
        ADC #0
        STA 837,X
        LDA #16
        STA 840,X
        LDA #0
        STA 841,X
        LDA #8
```

# CIA/WYJŚCIA w KYAN PASCALU

```
        STA 842,X
        JSR 58454
        CLC
        LDA SP
        ADC #16
        STA SP
        LDA SP+1
        ADC #0
        STA SP+1
        LDA 835,X
        CMP #1
        BEQ IOC4
        STA IOE1
        LDA #12
        STA 834,X
        JSR 58454
        PLA
        TAX
        JMP IOE2
IOC4    TXA
        LSR A
        LSR A
        LSR A
        LSR A
        LDY #3
        STA (SP),Y
        LDA #0
        LDY #4
        STA (SP),Y
        PLA
        TAX
#
    END;
(*****************************************)
(* Usuwa z dyskietki plik okreslony      *)
(* przez zmienna "fname".                *)

  PROCEDURE remove(VAR fname:string);
   BEGIN
#a
        TXA
        PHA
        CLD
        LDX #16
IOR1    LDA #255
        CMP 832,X
        BEQ IOR2
        TXA
        CLC
        ADC #16
        TAX
        CMP #96
        BNE IOR1
        LDA #161
        STA IOE1
        PLA
        TAX
        JMP IOE2
IOF2    TXA
        PHA
        SEC
        LDA SP
        SBC #16
        STA SP
        LDA SP+1
        SBC #0
        STA SP+1
        LDY #19
        LDA (SP),Y
        STA 212
        LDY #20
        LDA (SP),Y
        STA 213
        LDX #0
        CLC
IOR3    TXA
        ASL A
        TAY
        LDA (212),Y
        PHA
        TXA
        ADC #3
        TAY
        PLA
        STA (SP),Y
        INX
        CPX #15
        BNE IOR3
        PLA
        TAX
        LDA #33
        STA 834,X
        CLC
        LDA SP
        ADC #3
        STA 836,X
        LDA SP+1
        ADC #0
        STA 837,X
        LDA #16
        STA 840,X
        LDA #0
        STA 841,X
        JSR 58454
        CLC
        LDA SP
        ADC #16
        STA SP
        LDA SP+1
        ADC #0
        STA SP+1
        LDA 835,X
        STA IOE1
        LDA #12
        STA 834,X
        JSR 58454
        LDA #1
        CMP IOE1
        BEQ IOR4
        PLA
        TAX
        JMP IOE2
IOR4    PLA
        TAX
#
    END;
(*****************************************)
(* Udostepnia urzadzenie lub plik na     *)
(* dyskietce o nazwie okreslonej         *)
(* przez zmienna "fname" dla operacji    *)
(* okreslonej przez parametr "mode".     *)
(* W zadnym przypadku zawartosc pliku    *)
(* nie jest zmieniana. Funkcja zwraca    *)
(* wartosc rowna deskryptorowi           *)
(* zwiazanemu z plikiem                  *)

  FUNCTION open(VAR fname:string;mode:t
ryb):filedesc;
    BEGIN
#a
        TXA
        PHA
        CLD
        LDX #16
IOD1    LDA #255
        CMP 832,X
        BEQ IOD2
        TXA
        CLC
        ADC #16
        TAX
        CMP #96
        BNE IOD1
        LDA #161
        STA IOE1
        PLA
        TAX
        JMP IOE2
IOD2    TXA
        PHA
        SEC
        LDA SP
        SBC #16
        STA SP
        LDA SP+1
        SBC #0
        STA SP+1
        LDY #23
        LDA (SP),Y
        STA 212
        LDY #24
        LDA (SP),Y
        STA 213
        LDX #0
        CLC
IOD3    TXA
        ASL A
        TAY
        LDA (212),Y
        PHA
        TXA
        ADC #3
        TAY
        PLA
        STA (SP),Y
        INX
        CPX #15
        BNE IOD3
        PLA
        TAX
        LDA #3
        STA 834,X
        CLC
        LDA SP
        ADC #3
        STA 836,X
        LDA SP+1
        ADC #0
        STA 837,X
        LDA #16
        STA 840,X
        LDA #0
        STA 841,X
        LDA #12
        STA 842,X
        JSR 58454
        CLC
        LDA SP
        ADC #16
        STA SP
        LDA SP+1
        ADC #0
        STA SP+1
        LDA 835,X
        CMP #1
        BEQ IOD4
        STA IOE1
        LDA #12
        STA 834,X
        JSR 58454
        PLA
        TAX
        JMP IOE2
IOD4    TXA
        LSR A
        LSR A
        LSR A
        LSR A
        LDY #3
        STA (SP),Y
        LDA #0
        LDY #4
        STA (SP),Y
        LDY #5
        LDA (SP),Y
        STA 842,X
        PLA
        TAX
#
    END;
(*****************************************)
(* Zwalnia deskryptor "fd" zwiazany      *)
(* z urzadzeniem lub plikiem na          *)
(* dyskietce w wyniku wykonania          *)
(* operacji "create" lub "open".         *)

  PROCEDURE close(fd:filedesc);

    BEGIN
#a
        TXA
        PHA
        CLD
        LDY #3
        LDA (SP),Y
        ASL A
        ASL A
        ASL A
        ASL A
        TAX
        LDA #12
        STA 834,X
        JSR 58454
        PLA
        TAX
#
    END;
(*****************************************)
(* Wypisuje znak "c" na urzadzenie       *)
(* lub plik na dyskietce okreslone       *)
(* poprzez deskryptor pliku "fd".        *)

  PROCEDURE putcf(c:character;fd:filede
sc);
    BEGIN
#a
        TXA
        PHA
        CLD
        LDY #3
        LDA (SP),Y
        ASL A
        ASL A
        ASL A
        ASL A
        TAX
        LDA #11
        STA 834,X
        CLC
        LDA SP
        ADC #5
        STA 836,X
        LDA SP+1
        ADC #0
        STA 837,X
        LDA #1
        STA 840,X
        LDA #0
        STA 841,X
        JSR 58454
        LDA 835,X
        CMP #1
        BEQ IOP1
        STA IOE1
        LDA #12
        STA 834,X
        JSR 58454
        PLA
        TAX
        JMP IOE2
IOP1    PLA
        TAX
#
    END;
(*****************************************)
(* Wczytuje jeden, kolejny znak z        *)
(* pliku na dyskietce lub urzadzenia     *)
(* okreslonego przez deskryptor "fd".    *)
(* Gdy na pliku nie sa dostepne zadne    *)
(* znaki, zwraca wartosc ENDFILE, a      *)
(* gdy koniec wiersza, to wartosc        *)
(* NEWLINE. W przeciwnym przypadku       *)
(* zwraca wartosc wczytanego znaku.      *)
(* Zwracana wartosc przypisywana jest    *)
(* jednoczesnie zmiennej "c".            *)
  FUNCTION getcf(VAR c:character;fd:fil
edesc):character;

    BEGIN
#a
        TXA
        PHA
        CLD
        LDY #5
        LDA (SP),Y
        ASL A
        ASL A
        ASL A
        ASL A
        TAX
        LDA #7
        STA 834,X
        LDY #7
        LDA (SP),Y
        STA 836,X
        STA 212
        LDY #8
        LDA (SP),Y
        STA 837,X
        STA 213
        LDA #1
        STA 840,X
        LDA #0
        STA 841,X
        LDA #0
        LDY #0
        STA (212),Y
        LDY #1
        STA (212),Y
        JSR 58454
        LDA 835,X
        CMP #1
        BEQ IOG2
        CMP #3
        BEQ IOG2
        CMP #136
        BEQ IOG1
        STA IOE1
        LDA #12
        STA 834,X
        JSR 58454
        PLA
        TAX
        JMP IOE2
IOG1    LDA #255
        LDY #0
        STA (212),Y
        LDY #1
        STA (212),Y
IOG2    LDY #0
        LDA (212),Y
        LDY #3
        STA (SP),Y
        LDY #1
        LDA (212),Y
        LDY #4
        STA (SP),Y
        PLA
        TAX
#
    END;
(*****************************************)
(* Wypisuje znak "c" na standardowe      *)
(* wyjscie. Gdy znak jest rowny          *)
(* NEWLINE, to przechodzi do nowego      *)
(* wiersza.                              *)

  PROCEDURE putchar(c:character);
   BEGIN
    putcf(c,STDOUT)
   END;
(*****************************************)
(* Wczytuje jeden, kolejny znak ze       *)
(* standardowego wejscia. Gdy nie sa     *)
(* tam dostepne zadne znaki, zwraca      *)
(* wartosc ENDFILE, gdy na wejsciu       *)
(* jest koniec wiersza zwraca wartosc    *)
(* NEWLINE. W przeciwnym przypadku       *)
(* zwraca wartosc wczytanego znaku.      *)
(* Wartosc ta jest przypisywana          *)
(* zmiennej "c". Wczytywanie jest        *)
(* niebuforowane, w przypadku klawia-    *)
(* tury nie wymaga nacisniecia           *)
(* klawisza RETURN i nie zostawia        *)
(* echa na ekranie.                      *)
  FUNCTION getchar(VAR c:character):cha
racter;
    BEGIN
     getchar:=getcf(c,STDIN)
    END;
(*****************************************)
(* Wypisuje znaki tworzace zmienna       *)
(* "str", az do koncowej wartosci        *)
(* ENDSTR, lecz bez niej, na urzadze-    *)
(* nie lub plik na dyskietce okres-      *)
(* lony przez deskryptor pliku "fd".     *)

  PROCEDURE putstr(VAR str:string;fd:fi
ledesc);
   VAR
    i:integer;
   BEGIN
    i:=1;
    WHILE str[i]<>ENDSTR DO BEGIN
     putcf(str[i],fd);
     i:=i+1
    END
  END;
(*****************************************)
(* Wczytuje jeden, kolejny wiersz z      *)
(* urzadzenia lub pliku na dyskietce     *)
(* okreslonego przez deskryptor "fd".    *)
(* Wczytywane, kolejne znaki sa wpi-     *)
(* sywane do zmiennej "str", az do       *)
(* wartosci NEWLINE wlacznie, po         *)
(* ktorej jest umieszczana wartosc       *)
(* ENDSTR. Wczytywane jest maksimum      *)
(* "maxsize-1" znakow. Funkcja zwraca    *)
(* wartosc "true", gdy wczytanie         *)
(* zakonczy sie pomyslnie, oraz          *)

(* "false" w przeciwnym przypadku.       *)

  FUNCTION getline(VAR str:string;fd:fi
ledesc;maxsize:integer):boolean;
   VAR
    i:integer;
    c:character;
   BEGIN
    i:=1;
    REPEAT
     str[i]:=getcf(c,fd);
     IF fd=STDIN THEN
      putcf(c,0);
     i:=i+1
    UNTIL ((c=ENDFILE) OR (c=NEWLINE)
OR (i>=maxsize));
    IF (c=ENDFILE) THEN
     i:=i-1;
    str[i]:=ENDSTR;
    getline:=(c<>ENDFILE)
  END;

(*****************************************)
(* Inicjuje zmienne STDIN i STDOUT.      *)
(* W rezultacie standardowym wejsciem    *)
(* jest klawiatura korzystajaca z        *)
(* pierwszego IOCB, standardowym         *)
(* wyjsciem monitor korzystajacy z       *)
(* zerowego IOCB.                        *)

  PROCEDURE ioinit;
   VAR
    fname:string;
   BEGIN
    STDOUT:=0;
    STDIN:=1;
    close(STDIN);
    fname[1]:=ord('K');
    fname[2]:=ord(':');
    fname[3]:=ENDSTR;
    STDIN:=open(fname,IOREAD);
  END;


(*****************************************)
(***  KONIEC OPERACJI PIERWORNYCH  ***)
```

## SUPLEMENT DO

# CHARS DESIGNER

**Program „Chars Designer" Andrzeja-Piotra Szyndrowskiego („Moje Atari" 1) umożliwia zapis zaprojektowanego zestawu znaków na kasetę. Do jego odczytu służy specjalny podprogram „ChD's Generator Loader". Zapis i odczyt realizowane są z zachowaniem długich przerw między rekordami.**

Użytkownicy magnetofonów mogą ominąć tę niedogodność stosując zmodyfikowaną wersję loadera pokazaną na listingu 1. Liczba i numeracja wierszy pozostają zgodne z wersją pierwotną, natomiast przerwy między rekordami ulegają skróceniu. Komentarze w odnośnych wierszach listingu przybliżają sposób rozwiązania zagadnienia.

Aby zestaw zaprojektowanych przez użytkownika znaków graficznych mógł być odczytany zmodyfikowanym loaderem, należy go wcześniej odpowiednio na kasecie zapisać. Wymaga to zmian w wierszach od 6035 do 6055 podprogramu „Zapis generatora". Zmiany te są podane w listingu 2.

Posiadaczy magnetofonowych systemów Turbo ucieszy zapewne fakt, że to właśnie z myślą o nich pojawiła się litera „T" w wierszu 6035 listingu 2 i, że nie symbolizuje ona tym razem modemu. Najistotniejszą zaletą proponowanych modyfikacji zapisu i odczytu jest bowiem ich sprawdzone działanie w systemach Turbo Blizzard i Atari Super Turbo. W obu wypadkach trzeba jeszcze zmienić wiersz 31030 w pierwszym listingu na:

```
TG    6015 TRAP 6055
```

Użytkownicy Atari Super Turbo muszą dodatkowo zmienić w obu listingach poniższe wiersze:

```
IL    31005 GRAPHICS 0:TRAP 31055
NW    31030 CLOSE #2:OPEN #2,4,128,"T:":REM
      KROTKIE PRZERWY
```

Ponadto w tym wypadku zalecane jest podanie nazwy zapisywanego zestawu. Magnetofon przy odczycie każdorazowo zatrzyma się na jego nagłówku, oczekując na naciśnięcie dowolnego klawisza.

*Marek Kulczycki*

## LISTING 1

```
HN    0 REM * ChD's Generator "C:" Loader *
IT    1 REM *      Marek A. Kulczycki      *
UV    2 REM *    (c) 1990, Sp. Bajtek      *
EI    3 REM *  Modyfikacja "ChD Loadera"   *
KG    4 REM *     A. Szyndrowskiego        *
AZ    5 REM *       Moje Atari 1           *
NL    6 REM
DW    31000 POKE 106,PEEK(106)-10
CF    31005 GRAPHICS 0:POKE 559,0
GJ    31010 X=PEEK(106)+2
FS    31015 IF X=INT(X/4)*4 THEN 31025
PB    31020 X=X+1:GOTO 31015
PW    31025 POKE 764,12:REM SAMOSTART
TP    31030 CLOSE #2:OPEN #2,4,128,"C:":REM
      KROTKIE PRZERWY
EH    31035 POKE 866,7:REM GET #2
QB    31040 POKE 868,0:POKE 869,X:REM ADRES
      BLOKU DANYCH (X*256)
HR    31045 POKE 872,0:POKE 873,4:REM DLUGOS
      C BLOKU DANYCH (1 KB)
QH    31050 Y=USR(ADR("hhh■LV■"),32):REM PRO
      CEDURA CIOMAIN
SC    31055 CLOSE #2
ME    31060 GRAPHICS 0
XO    31065 POKE 756,X
ED    31070 RETURN
```

## LISTING 2

```
GI    6035 INPUT NAM$:IF NAM$(1,1)<>"C" AND
      NAM$(1,1)<>"T" THEN 6900
VB    6040 POKE 764,12:CLOSE #2:OPEN #2,4,12
      8,NAM$
WZ    6045 POKE 866,11:POKE 868,0:POKE 869,A
      DRES/256:POKE 872,0:POKE 873,4
TW    6050 Y=USR(ADR("hhh■LV■"),32)
OK    6055 CLOSE #2
```

# HARDCOPY 1029

**„Hardcopy" to angielskie słowo oznaczające trwałą kopię. W żargonie komputerowym niemal wszystkich języków oznacza ono kopię zawartości ekranu wykonaną na drukarce. Realizujących to zadanie programów jest wiele, lecz są one zwykle przeznaczone dla drukarek zgodnych ze standardem Epson lub IBM.**

Prezentowany program (listing 1) umożliwia skopiowanie ekranu w ósmym trybie graficznym na drukarce Atari 1029. Posługując się nim można otrzymać zarówno rysunek w takiej postaci, jak na ekranie, jak też jego negatyw. Program drukujący wywołuje się z poziomu Basica następującą sekwencją instrukcji.

```
OPEN #3,8,0,„P:"
A=USR(1536)
CLOSE #3
```

Aby otrzymać negatyw, należy przed USR wykonać jeszcze instrukcję **POKE 1659,176.** Ponowny druk pozytywu powodowany jest przez **POKE 1659,144.**

### Opis programu

Program drukujący jest umieszczony na szóstej stronie pamięci od adresu 1536. Korzysta on z 327-bajtowego bufora, który został ulokowany na stronie 126 (dwie strony przed początkiem programu ANTIC-a dla trybu 8).

Algorytm przekształcania obrazu w ciąg bajtów zrozumiałych dla drukarki zawiera pętla „LOOP". Z leżących „poziomo" bajtów pamięci ekranu należy skompletować „pionowe" bajty sterujące głowicą drukarki. Pierwszy bajt danych dla drukarki powstaje z najbardziej znaczących bitów pierwszego, czterdziestego pierwszego, 81, 121, 161, 201 i 241 bajtu pamięci ekranu. Pętla „LOOP" kompletuje więc wiersz o szerokości 7 punktów ekranu i długości 320 punktów.

Według zawartości rejestru NUMTXLIN (703) program orientuje się, czy występuje okno tekstowe i odpowiednio ustawia parametry. Pętla „STER" kieruje całością, procedura „MASKUJ" czyści bufor po wydrukowaniu kolejnego wiersza oraz obcina niepotrzebne bity z ostatniej linii obrazu (ani 192, ani 160 nie dzieli się bez reszty przez 7). Procedura „PUT" jest standardową procedurą zapisu ciągu bajtów korzystającą z kanału 3. Etykieta „KODY" wskazuje kody ustawiające odstęp wierszy (27, 65), tryb graficzny (27, 57) oraz liczbę danych graficznych (320 = $0140) dla drukarki 1029.

Program został napisany w asemblerze MAC/65. Ponadto listing 2 zawiera program drukujący zapisany w postaci danych w Basicu oraz prosty program demonstrujący jego działanie.

*Arkadiusz Lew-Kiedrowski*

```
HN    0 REM HARDCOPY 1029
NQ    1 REM Arkadiusz Lew-Kiedrowski
HZ    2 REM (c) 1990, Bajtek
NI    3 REM
VA    4 S=0:FOR T=1536 TO 1780:READ A:S=S+A:
      POKE T,A:NEXT T
WK    5 IF S<>29976 THEN ? "BLAD":END
QX    6 GRAPHICS 24:COLOR 1
BQ    7 FOR G=1 TO 20:PLOT 190,90:DRAWTO RND
      (0)*319,RND(0)*191:NEXT G
LI    8 POKE 1659,144:GOSUB 100:REM DRUK POZ
      YTYWU
LJ    9 POKE 1659,176:GOSUB 100:REM DRUK NEG
      ATYWU
YP    10 END
VI    100 OPEN #3,8,0,"P:"
YI    110 A=USR(1536)
HZ    120 CLOSE #3:RETURN
IN    1000 DATA 104,169,0,133,208,173,191,2,
      201,4,240,6,162,27,160,112,208,4
FW    1010 DATA 162,22,160,126,134,209,132,0
      ,162,6,189,238,6,157,255,125,202
UU    1020 DATA 208,247,165,88,133,205,165,8
      9,133,206,169,0,133,30,169,71
PS    1030 DATA 133,29,169,155,141,70,127,32
      ,177,6,32,96,6,32,208,6,169,2
TL    1040 DATA 133,30,169,69,133,29,198,209
      ,208,235,32,177,6,165,0,133,208
VS    1050 DATA 32,96,6,32,177,6,32,208,6,96
      ,162,7,169,128,133,207,70,207
VV    1060 DATA 169,6,133,203,169,126,133,20
      4,138,72,160,0,152,72,177,205
LC    1070 DATA 160,0,10,144,8,170,177,203,5
      ,207,145,203,138,200,192,8,208
KG    1080 DATA 240,165,203,24,105,8,133,203
      ,165,204,105,0,133,204,104,168
BV    1090 DATA 200,192,40,208,214,165,205,2
      4,105,40,133,205,165,206,105,0
EP    1100 DATA 133,206,104,170,202,208,182,
      96,169,6,133,203,169,126,133,204
AN    1110 DATA 160,0,162,2,177,203,37,208,1
      45,203,136,208,247,198,203,230
DX    1120 DATA 204,160,64,202,208,238,96,16
      2,48,169,11,157,66,3,165,30,157
LD    1130 DATA 68,3,169,126,157,69,3,165,29
      ,157,72,3,169,1,157,73,3,32,86
BN    1140 DATA 228,96,27,57,27,65,1,64
```

# ACTION! (6)

**Ostatnia już część opisu języka Action! dotyczy działania kompilatora oraz zasad programowania i uruchamiania programów w tym języku. Na końcu zostały ponadto podane zasady składni Action! i konwersji programów z Atari Basic.**

## KOMPILATOR ACTION!

Atari Basic pozwala pisać program w języku zbliżonym do angielskiego, a następnie bezpośrednio sprawdzić jego działanie bez realizowania dodatkowych operacji. Zaleta ta jest okupiona koniecznością istnienia specjalnego programu (zwanego interpreterem Basica), który rozpoznaje i wykonuje każdą instrukcję w każdym wierszu podczas realizacji programu.

**Action!** jest nieco bardziej złożony. Wymaga przekształcenia twojego programu przed jego realizacją przez specjalny program (zwany kompilatorem). Konieczne jest w tym celu wykonanie dodatkowej operacji między wpisaniem programu a jego uruchomieniem. Operacja ta nazywa się kompilacją. Podczas kompilacji kompilator **Action!** analizuje twój program wiersz po wierszu i przekształca go na program w języku maszynowym komputera. Dopiero tak przekształcony (skompilowany) program może być uruchomiony i działa wtedy znacznie szybciej niż interpretowany program w Atari Basic.

W tej części opisu będziemy używać kilku terminów, omawianych już wcześniej. Są to:

<ident> dowolny poprawny identyfikator
<wart> dowolna wartość dziesiętna lub szesnastkowa
<st_komp> obliczony adres identyfikatora
<adres> adres komórki pamięci

Ponadto zajmiemy się ponownie — nieco dokładniej — dyrektywami kompilatora. Przypominam, że dyrektywy te są wykonywane podczas kompilacji programu, a nie podczas jego działania i nie można ich użyć do zmiany parametrów systemu po uruchomieniu programu.

## WYKORZYSTANIE PAMIĘCI

Opiszę teraz, jak kompilator **Action!** wykorzystuje dostępny obszar pamięci komputera do umieszczenia w nim skompilowanego programu, zmiennych, procedur i tablic symboli.

Po wywołaniu kompilator najpierw określa miejsce umieszczenia kodu wynikowego, uzyskanego po komplikacji programu źródłowego w **Action!.** Wykonuje on to przez sprawdzenie zawartości komórek pamięci 14 i 15. Zapisana w nich liczba typu **CARD** jest adresem początku wolnej pamięci. Adres ten jest zmienny i zależy od rozmiaru programu zapisanego w edytorze. Jeżeli nie wskażesz inaczej, to skomplikowany program jest umieszczany w pamięci od tego adresu. Dla wskazania kompilatorowi, gdzie ma umieścić skompilowany program, należy bezpośrednio przed kompilacją podać monitorowi dwa następujące polecenia:

**SET 14=<adres>**
**SET $491=<adres>**

**gdzie** <adres> jest adresem początku skompilowanego programu.

Trzeba przy tym pamiętać, że komentarze, dyrektywy **SET** i dyrektywy **DEFINE** nie tworzą żadnego kodu wynikowego. Jest tak, ponieważ nie działają one podczas pracy programu, a więc ich kod wynikowy jest zbędny.

### Umieszczenie zmiennych

Informacje o zmiennych są przez kompilator **Action!** umieszczane w dwóch różnych miejscach — wewnątrz samego programu wynikowego i w tablicy symboli. Tablica ta jest opisana nieco dalej.

Wewnątrz programu zmienne są umieszczane przed tą częścią kodu maszynowego, w której zostały użyte. Niektóre zmienne są deklarowane przed pierwszą procedurą. Zmienne te (zwane globalnymi) mogą być użyte przez każdą znajdującą się dalej procedurę. Nie wymagają one dodatkowych deklaracji wewnątrz procedury.

Miejsce przeznaczane na zapisanie zmiennych zależy od ich typu. Poniższa tabela pomoże ci zrozumieć, w jaki sposób kompilator umieszcza zmienne.

| typ danej | miejsce | komentarz |
|---|---|---|
| BYTE | 1 bajt | typ podstawowy |
| CHAR | 1 bajt | typ podstawowy |
| CARD | 2 bajty | typ podstawowy |
| INT | 2 bajty | typ podstawowy |
| ARRAY | rozmiar typu razy liczba elementów | typ rozszerzony |
| TYPE | suma rozmiarów typów podanych w deklaracji | typ rozszerzony |
| ciąg | liczba znaków w ciągu plus bajt określający długość | każdy ciąg jest zapisywany oddzielnie nawet, jeśli ma taki sam identyfikator |

# Procedury i funkcje

Kompilator przeznacza dla procedur i funkcji miejsce w pamięci za zmiennymi globalnymi. Zmienne deklarowane lokalnie w procedurze poprzedzają rozkazy kodu wynikowego tej procedury. Tekst programu (instrukcje wewnątrz procedury lub funkcji) jest zamieniany bezpośrednio na rozkazy kodu maszynowego.

# Programy dołączone

W dowolnym miejscu programu można do niego dołączyć inne programy przy użyciu dyrektywy **INCLUDE**. Oczywiście, dołączany tekst nie może kolidować z aktualnie kompilowanym programem. Chodzi tu przede wszystkim o zbieżność identyfikatorów. Gdy w dołączonym programie zostanie wykryty błąd, jest on zwykle wyświetlany w obszarze ekranu przeznaczonym na komunikaty. Numer błędu zawsze jest pokazywany w wierszu poleceń monitora i sygnalizowany dźwiękiem.

# Dodatkowe zmienne globalne

Dodatkowe globalne zmienne, tablice i rekordy mogą być dodawane w razie konieczności przez użycie słowa kluczowego **MODULE**. Zmienne te zajmują miejsce za kodem wynikowym ostatniej poprzedzającej je procedury. Ich identyfikatory są dołączane do globalnej tablicy symboli kompilatora i mogą być wykorzystywane przez wszystkie następujące po nich procedury i funkcje.

# Tablice symboli

Kompilator **Action!** przechowuje dwie tablice symboli — jedną dla zmiennych globalnych i jedną dla zmiennych lokalnych z ostatnio kompilowanej procedury. Tablice symboli są dostępne dla użytkownika z monitora **Action!** poprzez polecenia **?**, * i **SET** (patrz opis monitora w „Tylko o Atari 1"). Mogą one być również użyte przez kompilator, gdy wymagany jest adres zmiennej.

Kompilator przeznacza na te tablice osiem stron pamięci (2 KB) znajdujące się na szczycie dostępnej pamięci. Ponieważ w ten sposób znajdują się one bezpośrednio poniżej pamięci obrazu, to są niszczone przy zmianie trybu graficznego na taki, który wymaga więcej miejsca niż tryby 0. Oznacza to, że nie możesz wrócić do monitora podczas wykonywania programu w celu sprawdzenia wartości zmiennych. Nie ma to jednak żadnego znaczenia dla poprawnego działania programu.

# UŻYCIE MENU WARIANTÓW

Menu wariantów oferuje szereg sposobów rozszerzenia lub zmiany działania kompilatora **Action!**. Warianty te były już częściowo opisane poprzednio.

**Zwiększenie szybkości kompilatora:**

Możesz osiągnąć zwiększenie szybkości kompilacji maksymalnie o 30% przez wykorzystanie menu wariantów do wyłączenia obrazu podczas dyskowych operacji wejścia/wyjścia i kompilacji programu. Po prostu wpisz „N<RETURN>" w odpowiedzi na pytanie „Screen?" wyświetlone w menu wariantów.

UWAGA: Wyłącza to również obraz dla innych funkcji systemu **Action!**, więc po zakończeniu kompilacji powinieneś ponownie przywrócić wyświetlanie obrazu.

**Wyłączenie brzęczyka:**

Gdy testujesz nowy program zawierający wiele błędów, możesz wyłączyć dźwięk brzęczyka. Wpisz tylko „N<RETURN>" w odpowiedzi na „Bell?" w menu wariantów.

**Rozpoznawanie wielkości liter**

Niekiedy, szczególnie gdy jesteś początkującym programistą, możesz żądać, aby kompilator pomagał ci w programowaniu przez wskazywanie słów kluczowych **Action!** wpisanych małymi literami. Możesz także chcieć, aby duże i małe litery w identyfikatorach były rozróżniane. Uzyskasz te funkcje wpisując w menu wariantów „Y<RETURN>" w odpowiedzi na pytanie „Case sensitive?".

Wykorzystanie tego wariantu nie jest konieczne do poprawnego programowania w **Action!**. Jednakże jest on użyteczny, gdy stosujesz dużą liczbę różnych identyfikatorów.

**Listing kompilacji**

Możesz nakazać kompilatorowi wyświetlanie każdego wiersza programu, jaki jest komplikowany. Nie jest to konieczne, gdyż większość występujących błędów jest wyświetlana na ekranie podczas procesu kompilacji. Możesz jednak kompilować długi program, który zawiera procedury dołączane (**INCLUDE**) z innych plików źródłowych. Jeśli tak, to nigdy nie możesz obejrzeć w inny sposób pełnego listingu programu źródłowego. Możesz go otrzymać na ekranie, a nawet wydrukować (przez zmianę

# TABELA 1

**Stałe Action!**

**Stałe liczbowe**

```
<st_liczb> ::= <liczba_dec> | <liczba_hex> | <znak>
<liczba_dec> ::= <liczba_dec><cyfra> | <cyfra>
<liczba_hex> ::= <liczba_hex><cyfra_hex> | $<cyfra_hex>
<znak> ::= '<dowolny drukowalny znak>
<cyfra_hex> ::= <cyfra> | A | B | C | D | E | F
<cyfra> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

**Stałe tekstowe**

```
<st_tekst> ::= "<ciag>"
<ciag> ::= <ciag><znak_tekst> | <znak_tekst>
<znak_tekst> ::= <kazdy drukowalny znak oprocz ">
```

**Stałe kompilatora**

```
<st_komp> ::= <st_komp>+<podst_st_komp> | <podst_st_komp>
<podst_st_komp> ::= <ident> | <st_liczb> | <odw_wsk> | *
```

**Operatory i podstawowe typy danych**

**Operatory**

```
<op_spec> ::= AND | OR | & | %
<op_rel> ::= XOR | ! | = | # | <> | < | <= | > | >=
<op_dod> ::= + | -
<op_mnoż> ::= * | / | MOD | LSH | RSH
<op_1arg> ::= @ | -
```

**Podstawowe typy danych**

```
<typ_podst> ::= CARD | CHAR | BYTE | INT
```

**Struktura programu**

**Program w Action!**

```
<program> ::= <program> MODULE <mod_prog> | (MODULE) <mod_prog>
<mod_prog> ::| (<dekl_syst>) <lista_proc>
```

**Deklaracje systemowe**

```
<dekl_syst> ::= <dyr_DEFINE> | <dekl_TYPE> | <dekl_zm>
```

**Dyrektywa DEFINE**

```
<dyr_DEFINE> ::= <DEFINE><lista_def>
<lista_def> ::= <lista_def>,<def> | <def>
<def> ::= <ident>=<st_tekst>
```

**Deklaracja TYPE**

```
<dekl_TYPE> ::= TYPE <lista_id_rek> <id_rek> | <id_rek>
<id_rek> ::= <nazwa_rek>=[<pole_pocz>]
<nazwa_rek> ::= <ident>
<pole_pocz> ::= <dekl_podst>
```

**Deklaracja zmiennej**

```
<dekl_zm> ::= <dekl_zm> <podst_dekl_zm> | <podst_dekl_zm>
<podst_dekl_zm> ::= <dekl_podst> | <dekl_wsk> | <dekl_tabl> | <dekl_rek>
```

**Deklaracja zmiennej typu podstawowego**

```
<dekl_podst> ::= <dekl_podst> <dekl_prosta> | <dekl_prosta>
<dekl_prosta> ::= <typ_podst> <lista_id_podst>
<typ_podst> ::= CARD | CHAR | BYTE | INT
<lista_id_podst> ::= <lista_id_podst>,<id_podst> | <id_podst>
<id_podst> ::= <ident>{=<war_pocz>}
<war_pocz> ::= <adres> | [<wartość>]
<adres> ::= <st_komp>
<wartość>::= <st_liczb>
```

**Deklaracja zmiennej wskaźnikowej**

```
<dekl_wsk> ::= <typ_wsk> POINTER <lista_id_wsk>
<typ_wsk> ::= <typ_podst> | <nazwa_rek>
<lista_id_wsk> ::= <lista_id_wsk>,<id_wsk> | <id_wsk>
<id_wsk> ::= <ident>{=<wartość>}
```

**Deklaracja zmiennej tablicowej**

```
<dekl_tab> ::= <typ_podst> ARRAY <lista_id_tab>
<lista_id_tab> ::= <lista_id_tab>,<id_tab> | <id_tab>
<id_tab> ::= <ident>{(<rozm>)}{=<war_pocz_tab>}
<rozm> ::= <st_liczb>
<war_pocz_tab> ::= <adres> | [<lista_wart>] | <st_tekst>
<adres> ::= <st_komp>
<lista_war> ::= <lista_war><wartość> | <wartość>
<wartość> ::= <st_komp>
```

wyjścia), gdy odpowiesz „Y<RETURN>" na pytanie „List?" w menu wariantów.

# INFORMACJE TECHNICZNE

## Przepełnienie

Kompilator **Action!** nie sprawdza przepełnienia arytmetycznego (ang. *overflow* lub *underflow*). A co to jest przepełnienie?

Jezeli masz zmienną typu BYTE o wartości 255 i dodasz do niej 1, to otrzymasz 0, a nie 256 (ponieważ pojedynczy bajt może zawierać tylko wartość do 255). Podobnie, jeśli używasz systemu dziesiętnego i możesz wyświetlić tylko dwie cyfry, to natrafisz na taki sam problem dodając 1 do 99. Wiesz, że jest to 100, lecz możesz wyświetlić tylko dwie cyfry, więc zobaczysz „00". Jest to właśnie przepełnienie (*overflow*).

Przepełnienie ujemne (*underflow*) jest dokładnie odwrotne. Jeżeli odejmiesz 1 od zera, to otrzymasz 255.

Jak to było wcześniej opisane, niektóre operatory matematyczne dają określone typy wartości. Możesz więc czasem uniknąć tego problemu przez wykorzystanie automatycznej zmiany typu wartości.

Podobnie, przepełnienie jest powodowane przez operatory przesunięca (**LSH** i **RSH**). Przesunięcie zawartości zmiennej daje wynik podobny (lecz nie identyczny) jak mnożenie lub dzielenie przez 2.

## Zgodność typów

Musisz również uważać, gdyż kompilator **Action!** nie sprawdza granic zmiennych prostych i tablic. Pominięcie tej kontroli pozwala na większą elastyczność w operowaniu danymi. Ceną za to jest konieczność zwiększenia uwagi. Możesz w tym celu napisać własne procedury kontrolujące granice tablic i wyświetlające komunikaty błędów. Procedury takie najlepiej wykorzystywać wielokrotnie w różnych programach przy pomocy dyrektywy **INCLUDE**.

## Korzystanie z kanału 7

Po uruchomieniu systemu **Action!** kanał 7 jest otwierany do odczytu z klawiatury (K:). Możesz wykorzystywać ten kanał do tego samego celu, lecz nie zmieniaj jego atrybutów przez zamykanie go i ponowne otwieranie.

UWAGA: Jeżeli używasz kanału 7 i zakładasz, że jest on otwarty, to twój program nie będzie działał bez modułu **Action!**.

## Dostępna pamięć

Przy pisaniu długiego programu wielkość dostępnej pamięci może okazać się niewystarczająca. Gdy to nastąpi, możesz zrobić jedną z trzech rzeczy, zależnie od chwili wystąpienia błędu.

**Podczas redagowania programu**

Zapisz tworzony program na kasecie lub dyskietce (<CTRL><SHIFT>W), przejdź do monitora i ponownie uruchom system (**BOOT**). Następnie wróć do edytora i odczytaj zapisany uprzednio program.

Jeżeli to nie pomaga (program jest bardzo długi), to trzeba podzielić go na mniejsze części i połączyć przy pomocy dyrektywy **INCLUDE**.

**Podczas kompilacji programu**

Przejdź do edytora i zapisz program. Następnie wróć do monitora, uruchom ponownie system i kompiluj program z nośnika (kasety lub dyskietki), na którym został zapisany. Jeżeli nie da to rezultatu, konieczne jest skrócenie programu lub zmiana jego wersji (patrz niżej).

## Wersje Action!

Oryginalny język **Action!** jest sprzedawany w postaci modułu ROM (*cartridge*). Ponadto dostępne są pirackie (kradzione) wersje na kasetach i dyskietkach. Cały system **Action!** zajmuje 16 KB pamięci, lecz w module ROM jest on podzielony na części włączane w razie potrzeby i w rzeczywistości zajmuje tylko 8 KB. Wersje pirackie nie mają tej możliwości i w związku z tym pozostawiają użytkownikowi o 8 KB, czyli o 1/3 pamięci mniej. Ponadto działają one znacznie wolniej podczas redagowania i kompilacji programu.

Ze względu na procedury i funkcje biblioteczne skompilowany program wymaga obecności **Action!** w pamięci komputera. Ominięcie tego warunku jest możliwe trzema różnymi sposobami. Po pierwsze, firma **ICD** sprzedaje wersję języka, która do skompilowanego programu dołącza procedury biblioteczne. Nie słyszałem jednak, aby ktokolwiek w Polsce posiadał tą wersję. Po drugie, istnieją programy umożliwiające dołączenie procedur do kompilowanego programu. Zwykle są to procedury „wydobyte" z systemu (ten sposób jest najczęściej stosowany). Po trzecie, można napisać program, który nie będzie zawierał biblioteki procedur i mimo to będzie działał poprawnie nawet bez **Action!**. Wymaga to wprowadzenia wielu ograniczeń, przede wszystkim nie wolno w takim programie stosować procedur bibliotecznych.

## Efektywne programowanie

Program w **Action!** jest w procesie kompilacji zamieniany na kod maszynowy w sposób bezpośredni, czyli tak, jak został zapisany przez użytkownika. W celu uzyskania szybkiego i zwartego programu wynikowego należy więc stosować w programie konstrukcje przeznaczone specjalnie dla **Action!** unikać zaś struktur przeniesionych z Basica. Jaskrawym przykładem są tu procedury biblioteczne **Poke** i **Peek**. Ich wygląd po kompilacji jest następujący:

```
Poke (752,1)          key=Peek(764)
    LDA #240              LDA #252
    LDX #2                LDX #2
    LDY #1                JSR PEEK
    JSR POKE              STA KEY
```

W rezultacie kod wynikowy w pierwszym wypadku ma długość dziewięciu bajtów, a w drugim — dziesięciu. Definiując komórki pamięci jako zmienne o określonym adresie uzyskujemy znacz-

# TABELA 2

```
     Deklaracja zmiennej rekordowej
<dekl_rek> ::= <ident> <lista_id_rek>
<lista_id_rek> ::= <lista_id_rek>,<id_rek> | <id_rek>
<id_rek> ::= <ident>[=<adres>]
<adres> ::= <st_komp>

          Odwołania do zmiennych
<odw_pam> ::= <zaw_pam> | @<ident>
<zaw_pam> ::= <odw_podst> | <odw_tab> | <odw_wsk> | <odw_rek>
<odw_podst> ::= <ident>
<odw_tab> ::= <ident>(<wyr_arytm>)
<odw_wsk> ::= <ident>^
<odw_rek> ::= <ident>.<ident>

          Procedury Action!

     Procedury
<lista_proc> ::= <lista_proc> <proc> | <proc>
<proc> ::= <procedura> | <funkcja>

     Struktura procedury
<procedura> ::= <dekl_proc> [<dekl_syst>] [<lista_instr>] [RETURN]
<dekl_proc> ::= PROC <ident>[=<adres>][(<dekl_par>)]
<adres> ::= <st_komp>

     Struktura funkcji
<funkcja> ::= <dekl_funkc> [<dekl_syst>] [<lista_instr>] [RETURN(<wyr_arytm>)]
<dekl_funkc> ::= <typ_podst> FUNC <ident>[=<adres>][(<dekl_par>)]
<adres> ::= <st_komp>

     Wywołanie procedury
<wyw_proc> ::= <wyw_FUNC> | <wyw_PROC>
<wyw_FUNC> ::= <ident>([<param>])
<wyw_PROC> ::= <ident>([<param>])
     Parametry
<dekl_par> ::= <dekl_zm>

          Instrukcje
<lista_instr> ::= <lista_instr> <instr> | <instr>
<instr> ::= <instr_prosta> | <instr_strukt> | <blok_kodu>
<instr_prosta> ::= <instr_przyp> | <instr_EXIT> | <wyw_proc>
<instr_strukt> ::= <instr_IF> | <pętla_DO> | <pętla_WHILE> | <pętla_FOR>

     Instrukcja przypisania
<instr_przyp> ::= <zaw_pam>=<wyr_arytm>

     Instrukcja EXIT
<instr_EXIT> ::= EXIT

     Instrukcja IF
<instr_IF> ::= IF <wyr_war> THEN [<lista_instr>] [|:<rozsz_ELSEIF>:|]
                  [<rozsz_ELSE>] FI
<rozsz_ELSEIF> ::= ELSEIF <wyr_war> THEN [<lista_instr>]
<rozsz_ELSE> ::= ELSE [<lista_instr>]
<wyr_war> ::= <rel_złoż>

     Pętla DO-OD
<pętla_DO> ::= DO [<lista_instr>] [<instr_UNTIL>] OD
<instr_UNTIL> ::= UNTIL <wyr_war>

     Pętla WHILE
<pętla_WHILE> ::= WHILE <wyr_war> <pętla_DO>

     Pętla FOR
<pętla_FOR> ::= FOR <ident>=<start> TO <stop> [STEP <krok>] <pętla_DO>
<start> ::= <wyr_arytm>
<stop> ::= <wyr_arytm>
<krok> ::= <wyr_arytm>

     Blok kodu
<blok_kodu> ::= [<lista_st-komp>]
<lista_st_komp> ::= <lista_st_komp> <st_komp> | <st_komp>

          Wyrażenia

     Wyrażenia relacji
<rel_złoż> ::= <rel_złoż><op_spec><rel_prosta> | <rel_prosta><op_spec><rel_prosta>
<rel_prosta> ::= <wyr_arytm><op_rel><wyr_arytm>

     Wyrażenia arytmetyczne
<wyr_arytm> ::= <wyr_arytm><op_dod><wyr_mult> | <wyr_mult>
<wyr_mult> ::= <wyr_mult><op_mnoż><wartość> | <wartość>
<wartość> ::= <st_liczb> | <odw_pam> | <wyr_arytm>
```

## TABELA 2

| Instrukcje Basica | Odpowiedniki w Action! |
|---|---|
| C=D+I*A | c=d+i*a |
| IF A<>0 THEN B=1 | IF a<>0 THEN b=1 FI |
| 10 IF A=0 THEN 30<br>20 B=1:C=A*2<br>30 REM | IF a<>0 THEN<br>b=1 c=a*2<br>FI |
| 10 IF A=0 THEN B=1:GOTO 30<br>20 B=7<br>30 REM | IF a=0 THEN b=1<br>ELSE b=7<br>FI |
| FOR I=1 TO 100 ...<br>NEXT I | FOR i=1 TO 100 DO ...<br>OD |
| PRINT "WITAJ" | PrintE("WITAJ") |
| PRINT "WITAJ"; | Print("WITAJ") |
| PRINT #5;"WITAJ" | PrintDE(5,"WITAJ") |
| PRINT #5;"WITAJ"; | PrintD(5,"WITAJ") |
| PRINT I | PrintIE(i) |
| PRINT "I=";I | PrintF("I=%I%E",i)<br>lub<br>Print("I=") PrintIE(i) |
| PRINT #3;B*3; | PrintBD(3,b*3) |
| INPUT I | Put('?) i=InputI() |
| INPUT B$ | Put('?) InputS(ba) |
| PUT #0,65 | Put('A)<br>lub<br>Put(65)<br>lub<br>Put($41) |
| GET #C,B | b=GetD(c) |
| OPEN #1,4,0,"K:" | Open(1,"K:",4,0) |
| CLOSE #3 | Close(3) |
| NOTE #1,C,B | Note(1,@c,@b) |
| POINT #1,C,B | Point(1,c,b) |
| XIO 18,#6,0,0,"S:" | XIO(6,0,18,0,0,"S:")<br>albo procedura Fill() |
| B=PEEK(C) | b=Peek(c)<br>lub lepiej<br>ba=c b=ba^ |
| POKE C,B | Poke(c,b)<br>lub lepiej<br>ba=c ba^=b |
| GRAPHICS 8 | Graphics(8) |
| COLOR 3 | color=3 |
| DRAWTO C,D | DrawTo(c,d) |
| LOCATE C,D,B | b=Locate(c,d) |
| PLOT C,D | Plot(c,d) |
| POSITION C,D | Position(c,d) |
| SETCOLOR 0,1,C | SetColor(0,1,c) |
| GRAPHICS 24:COLOR C:<br>PLOT 200,150:DRAWTO 120,20:<br>POSITION 40,150:POKE 765,C:<br>XIO 18,#6,0,0,"S:" | Graphics(24) color=c<br>Plot(200,150)<br>DrawTo(120,20)<br>Fill(40,150) |
| SUUND 0,121,10,6 | Sound(0,121,10,6) |
| C=PADDLE(B)<br>C=PTRIG(B)<br>C=STICK(B)<br>C=STRIG(B) | c=Paddle(b)<br>c=Ptrig(b)<br>c=Stick(b)<br>c=Strig(b) |
| B$=S$ | SCopy(ba,s) |
| B$=S$(3,5) | SCopyS(ba,s,3,5) |
| B$(3,5)=S$ | SAssign(ba,s,3,5) |
| B=INT(6*RND(0))+1 | b=Rand(6)+1 |
| FOR C=4000 TO 5000:<br>POKE C,0:NEXT C | Zero(4000,1001) |
| STOP | Break() |
| B$=STR$(I) | StrI(i,ba) |
| I=VAL(S$) | i=ValI(s) |

ną oszczędność miejsca (definicja występuje tylko raz i nie daje ani jednego bajtu kodu wynikowego). Jeżeli więc pokazane wyżej rejestry zdefiniujemy jako:

```
BYTE cinh=752, kb=764
```

to odpowiedni fragment programu będzie wyglądał następująco:

```
cinh=1          key=kb
  LDA #1        LDA KB
  STA CINH      STA KEY
```

Długość kodu wynikowego wynosi teraz w pierwszym przypadku 4 bajty, a w drugim 5, czyli jest on dwukrotnie krótszy.

## SKŁADNIA ACTION!

W tabeli 1 podana jest składnia języka **Action!** w notacji Backusa-Naura. W notacji tej występuje pięć symboli o specjalnym znaczeniu:

| symbol | znaczenie |
|---|---|
| : := | jest określone jako |
| \| | lub |
| {} | wariantowo |
| [: :] | wielokrotnie |
| ‹ident› | identyfikator (nazwa) |

# Konwersja programów z Basica na Action!

Tabela 2 prezentuje liczne funkcje procedury i instrukcje Basica. Dla każdego przykładu w Basicu podana jest odpowiednia struktura w **Action!**. W przykładach w Basicu nie są podane numery wierszy, chyba że są one niezbędne dla zrozumienia przykładu. W poniższych przykładach w **Action!** zakładamy, że zostały wcześniej zadeklarowane następujące zmienne:

```
INT i,j,k
CARD c,d,e
BYTE a,b
BYTE ARRAY s,t,aa,ba
CARD ARRAY ca,da,ea
INT ARRAY ia,ja,ka
```

## MAPA PAMIĘCI ACTION!

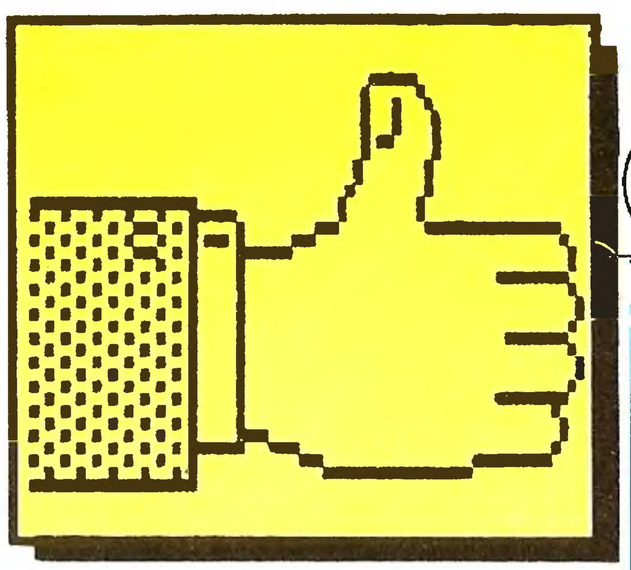
Poniższa mapa pamięci systemu **Action!** dotyczy wyłącznie wersji oryginalnej, umieszczonej w modelu ROM (taleba 3).

UWAGA: Obszar kodu kompilatora rozpoczyna się w miejscu, w którym kończy się bufor edytora. Czyni to dynamicznym położenie w pamięci zarówno edytora, jak i kompilatora.

*Wojciech Zientara*

## TABELA 3

| | |
|---|---|
| $00 | Zmienne OS i Action! |
| $CA | Wolny obszar |
| $CE | Zmienne Action! |
| $D4 | Rejestry liczb rzeczywistych |
| $100 | System Operacyjny |
| $480 | Zmienne Action! |
| $580 | Bufor liczb rzeczywistych |
| $600 | System operacyjny |
| MEMLO | Stos kompilatora Action! |
| LO+$200 | Bufor wejściowy Action! |
| LO+$300 | Tablica danych Action! |
| LO+$750 | Bufor edytora Action! |
| | Obszar kodu kompilatora Action! |
| TOP-$800 | Tablica symboli kompilatora |
| MEMTOP | Pamięć obrazu |
| $A000 | Moduł Action! |
| $C000 | System operacyjny |
| $FFFF | |



# OCENA

## MATEMATYKA

***Mam 13 lat. Przesyłam swój program, który napisałem na podstawie książki „Wakacje z gramolami" J. Tatarkiewicza i A. Witkowskiego. Proszę o ocenienie programu, ujawnienie błędów i napisanie, jak ich uniknąć.***

***Paweł Tutka***
***Gliwice***

Nadesłany program pod tytułem „Matematyka" jest pokazany na pierwszym wydruku. Służy on tylko do obliczania pierwiastków liczb, więc nazwa została mu nadana nieco na wyrost. Należało go zatytułować „Pierwiastki", co dokładnie oddaje treść realizowanego zadania.

Jak działa ten program? Po uruchomieniu wyświetla tytuł, imię i nazwisko autora oraz informację o realizowanym zadaniu. Następnie pyta o liczbę, a po jej wprowadzeniu wyświetla wynik pierwiastkowania. Jego działanie nie kończy się na tym, gdyż użytkownik musi zdecydować, czy chce jeszcze coś obliczać.

Działanie programu jest formalnie poprawne, ale — jak zwykle — „diabeł siedzi w szczegółach". Widać to już w chwili uruchomienia programu: dotychczasowa zawartość ekranu nie jest usuwana, a wyświetlane napisy są umieszczone bezpośrednio jeden pod drugim. Dla uzyskania estetycznego wyglądu zawsze należy rozpoczynać program instrukcją kasującą zawartość ekranu. Ponieważ miejsce na ekranie nie jest niczym ograniczone, to znacznie lepszy efekt wizualny uzyska się wyświetlając napisy w co drugim wierszu. Wynik obliczeń jest natomiast przedstawiany w czterech wierszach. Tu dla uzyskania lepszego efektu powinno się umieścić liczbę razem z jej opisem, a nie pod nim.

Drugą fazą programu jest wprowadzenie przez użytkownika danej i sprawdzenie jej poprawności. Zastosowana tu instrukcja INPUT nie ma żadnego zabezpieczenia. Wystarczy nacisnąć klawisz ‹RETURN› bez wpisywania liczby lub podać błędną wartość (np. 1O zamiast 10), aby program został przerwany komunikatem o błędzie. Niezbędne jest tu użycie instrukcji TRAP, która pozwoli na uniknięcie podobnych niespodzianek.

Autor ograniczył dozwolone wartości wprowadzanej liczby do zakresu od 0.0000000001 do 99999999999. W zapisie komputerowym będzie to zakres od 1E—10 do 9.99999999E+10. Nie wnikając w to, czy takie ograniczenie jest zasadne, czy nie, gdyż zależy to od woli autora, należy je jednak nieco poprawić. Dolną granicą może być po prostu zero, ponieważ dopiero liczby ujemne powodują błąd w funkcji SQR. Ze względu na zakres liczb rzeczywistych dostępnych w Atari górna granica może być znacznie podwyższona. Postawmy ją na ustalonym poziomie wprowadzając jednak niewieką poprawkę, czyli zwiększenie o jeden. Efektem tych zmian jest uzyskanie zakresu od 0 do 1E+11, co nie ma znaczenia dla programu, a jest prostsze do wpisania.

Podanie przez użytkownika wartości spoza wyznaczonego zakresu powoduje wyświetlenie komunikatu i przejście do końcowej części programu. Zostało to zrealizowane przy użyciu dwóch instrukcji IF i trzech GOTO (wiersze 50, 60, 120 i 130). Jednakże dzięki umieszczeniu całości w jednym wierszu programu można ograniczyć liczbę instrukcji do jednego IF i jednego GOTO. Wymaga to tylko użycia operatora logicznego OR (lub) do połączenia dwóch sprawdzanych warunków.

Kolejny etap programu jest bardzo krótki — następuje w nim tylko wyświetlenie danej i wyniku. Jednak nawet tu można wprowadzić poprawki. O jednej z nich (mającej znaczenie estetyczne) już napisałem. Druga zmiana ma znaczenie formalne. Wynik działania jest najpierw przypisywany zmiennej X (wiersz 70), a dopiero później wyświetlany. Lecz nowa zmienna X nie jest nigdzie więcej wykorzystywana, a więc jest zbędna. Wystarczy przecież obliczenie wykonać podczas wyświetlania zapisując wynik bezpośrednio na ekranie.

Najwięcej poprawek można dokonać w końcowej części programu. Kilkakrotnie zwracałem już uwagę na możliwość zastąpienia instrukcji INPUT przez GET w operacjach, których zadaniem jest tylko wybór jednego z kilku wariantów. Atari Basic umożliwia zastosowanie w takim przypadku jeszcze innej konstrukcji, która wykorzystuje kod ostatnio naciśniętego klawisza przechowywany w rejestrze KBCODES (adres 764). Dzięki temu eliminuje się konieczność deklarowania dodatkowej zmiennej Z$. Inaczej należy też rozwiązać sprawdzanie odpowiedzi. W oryginalnym programie występują kolejne porównania z obiema dopuszczalnymi wartościami (wiersze 160—180), które następnie powodują odpowiednie skoki. Zamiast równości trzeba w drugim porównaniu wprowadzić nierówność i zmienić kolejność sprawdzania, dzięki czemu eliminuje się jeden skok.

Po wyborze program się kończy lub jest kontynuowany. W zakończeniu również znalazł się błąd. Instrukcja NEW powoduje przerwanie realizacji programu i usunięcie go z pamięci. Z tego powodu następująca po niej instrukcja END nigdy nie zostanie wykonana.

Po wniesieniu opisanych wyżej poprawek program przedstawia się tak jak na drugim wydruku. Wyraźnie widoczne jest jego skrócenie. Nie jest to jednak wszystko. W programach nadsyłanych do oceny widać wyraźną niechęć autorów do pisania kilku instrukcji w jednym wierszu. Bardzo niechętnie jest również używana instrukcja ON. Zastąpienie IF przez ON (tam gdzie to możliwe) i umieszczanie kilku instrukcji w jednym wierszu wpływa najbardziej na długość programu, a ponadto zwiększa szybkość jego realizacji. Wystarczy porównać dwie pierwsze wersje z trzecią (wydruk 3), która pokazuje sposób maksymalnego „upakowania". Teraz program, który miał w pierwszej wersji 24 wiersze, a w drugiej 15, został zmniejszony do 7 wierszy.

*Wojciech Zientara*

***P.S. Czytelnicy zainteresowani nauką programowania powinni również samodzielnie analizować inne programy publikowane w „Moim Atari". Są one drukowane w postaci nadesłanej przez autorów, co nie oznacza, że nic nie można w nich poprawić. Uważny czytelnik w każdym z tych programów znajdzie kilka błędów.***

## LISTING 1

```
1 DIM Z$(1)
5 SETCOLOR 2,0,0
10 ? "    * * * M A T E M A T Y K A *
* *"
20 ? "       P A W E L       T U T K A
   "
25 ? "  PROGRAM OBLICZA PIERWIASTKI LI
CZB "
30 ? "Czekam na liczbe !"
40 INPUT Y
50 IF Y<1E-10 THEN GOTO 130
60 IF Y>9.99999999E+10 THEN GOTO 130
70 X=SQR(Y)
80 ? "Pierwiastkiem liczby"
90 ? Y
100 ? "jest liczba"
110 ? X
120 GOTO 140
130 ? "NIE BAW SIE !!!"
140 ? "Czy konczysz TAK-T, NIE-N  "
150 INPUT Z$
160 IF Z$="T" THEN GOTO 190
170 IF Z$="N" THEN GOTO 30
180 GOTO 130
190 ? CHR$(125)
200 NEW
210 END
```

## LISTING 2

```
10 GRAPHICS 0:SETCOLOR 2,0,0
20 ? "    * * * M A T E M A T Y K A *
* *":?
30 ? "       P A W E L       T U T K A
":?
40 ? "  PROGRAM OBLICZA PIERWIASTKI LI
CZB":?
50 ? "Czekam na liczbe !"
60 TRAP 60:INPUT Y
70 IF Y<0 OR Y>1E+11 THEN ? "NIE BAW S
IE !!!":GOTO 90
80 ? "Pierwiastkiem liczby ";Y:? "jest
 ";SQR(Y):?
90 POKE 764,255
100 ? "Czy konczysz TAK-T, NIE-N":?
110 Z=PEEK(764):IF Z=255 THEN 110
120 IF Z=35 THEN 50
130 IF Z<>45 THEN 110
140 ? CHR$(125)
150 NEW
```

## LISTING 3

```
10 GRAPHICS 0:SETCOLOR 2,0,0:? "    *
* * M A T E M A T Y K A * * *":? :? "
     P A W E L       T U T K A":?
20 ? "  PROGRAM OBLICZA PIERWIASTKI LI
CZB":?
30 ? "Czekam na liczbe !"
40 TRAP 40:INPUT Y:IF Y<0 OR Y>1E+11 T
HEN ? "NIE BAW SIE !!!":GOTO 60
50 ? "Pierwiastkiem liczby ";Y:? "jest
 ";SQR(Y):?
60 POKE 764,255:? "Czy konczysz TAK-T,
 NIE-N":?
70 Z=PEEK(764):ON Z=255 GOTO 70:ON Z=3
5 GOTO 30:ON Z<>45 GOTO 70:? CHR$(125)
:NEW
```


KAŻDE PAŃSTWA ZAMÓWIENIE ZREALIZUJEMY

LHK

**LHK Electronics Service S.A**
**81-736 Sopot**
**ul. Architektów 1A**
**✳ tlx.512742 BSP PL**
**PC/XT/AT/386/LAP**

w dowolnych konfiguracjach, po atrakcyjnych cenach proponujemy: HD 20,30,40,80 MB, monitors 14":amber, paper-white, EGA
Bogaty wybór akcesoriów komputerowych:
a) filtry
b) obudowy
c) ruchome ramiona
d) dyskietki itp.

LHK

**TWOIM TRAFNYM WYBOREM**

Uwaga! Odbiorcom hurtowym **udzielamy do 12% rabatu**

**NOWOŚĆ!**

DYSKIETKA CZYSZCZĄCA NAPĘDY GŁOWIC KOMPUTEROWYCH

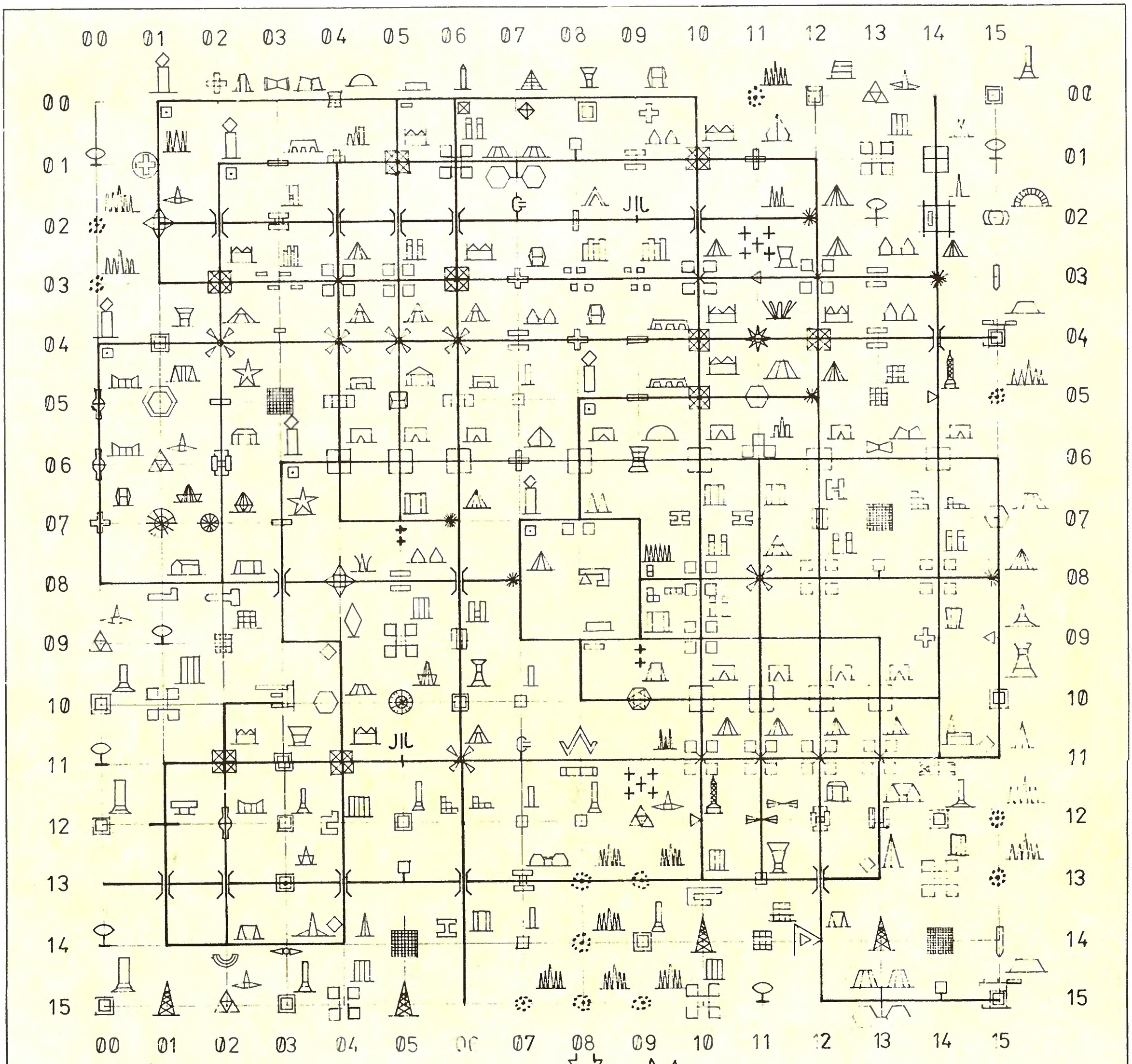
Dyskietka czyszcząca zapobiega zużywaniu się głowicy, przedłuża trwałość komputera, eliminuje wszelkie zanieczyszczenia bez ryzyka uszkodzenia komputera. Dyskietka może być używana wielokrotnie, posiada wymienne wkłady i wystarcza na kilka miesięcy pracy komputera.
Cena promocyjna 25000 zł.

**Zamówienie**

**zamawiam ..... sztuk ✳ 25000 zł**
**Nazwisko/firma .....**
**adres .....**
**wysyłka za zaliczeniem pocztowym**

**LHK prowadzi regenerację taśm do wszystkich typów drukarek**
**Regenerując taśmę — oszczędzasz pieniądze**
**Usługę realizujemy w 48 godzin — wysyłka za zaliczeniem pocztowym.**

GRY




**Zgodnie z obietnicą prezentujemy mapę drugiego miasta z gry „Mercenary". Przypominamy jednocześnie, że plan pierwszego miasta był opublikowany w „Moim Atari 2", a krótki opis gry w „Moim Atari 1".**

# MERCENARY

## — THE SECOND CITY

# KRUCZKI i SZTUCZKI

## Aby otrzymać płynny przesuw w trybie GRAPHICS 0 należy użyć poniższych instrukcji:

**POKE 622,2:POKE 700,2:GRAPHICS 0**

* * * * *

Znaki kontrolne (oprócz RETURN) można uzyskać na ekranie czterema sposobami:

1. **POKE 672,128** zawsze wyświetla następny znak (tylko jeden);
2. identyczne działanie ma **? CHR$(27);** (także tylko dla jednego znaku);
3. **POKE 766,1** powoduje wyświetlanie wszystkich znaków, aż do instrukcji **POKE 766,0;**
4. **?#6;CHR$(N)** gdzie N jest kodem ASCII wyświetlanego znaku, przedtem musi być koniecznie wykonana instrukcja **GRAPHICS 0**, a jeśli na końcu **PRINT** nie ma średnika lub przecinka, to pozostaje ślad kursora.

* * * * *

Przerwanie pracy na komputerze na kilka minut powoduje jego przejście do trybu przyciągania uwagi, a więc zmianę kolorów ekranu. Normalne kolory przywracane są po naciśnięciu dowolnego klawisza. Jeśli korzystamy z jakiegoś programu użytkowego, to można w ten sposób narobić sobie szkód. Należy w takim przypadku nacisnąć klawisze **‹CONTROL›+‹SHIFT›+‹A›.** Komputer odbiera to jako naciśnięcie klawisza, lecz taka kombinacja ma kod 255, czyli sygnalizuje, że nie został naciśnięty żaden klawisz!

* * * * *

Jan Musielak z Poznania przekazał nam sposób wykonania w Kyan Pascalu instrukcji POKE, której dane (adres komórki i wartość do niej wpisywana) są znane podczas pisania programu. Realizuje to następujący fragment programu:

```
#A
LDA #wartość
STA adres
#
```

* * * * *

Natomiast Michał Kaczmarczyk z Nowego Sącza napisał, jak uzyskać tryb **GRAPHICS 0** z oknem tekstowym. Należy w tym celu wpisać:

**GRAPHICS 0:POKE 703,4:POKE 87,12**

Wówczas cały ekran będzie dostępny przez **? #6**, a samo okno tekstowe przez **?.** Znaczenie kolejnych instrukcji w podanej sekwencji jest następujące: otwarcie kanału 6 dla ekranu, utworzenie okna tekstowego, „przekonanie" komputera, że ma do czynienia z trybem 12 (taki sam przydział pamięci jak tryb 0).

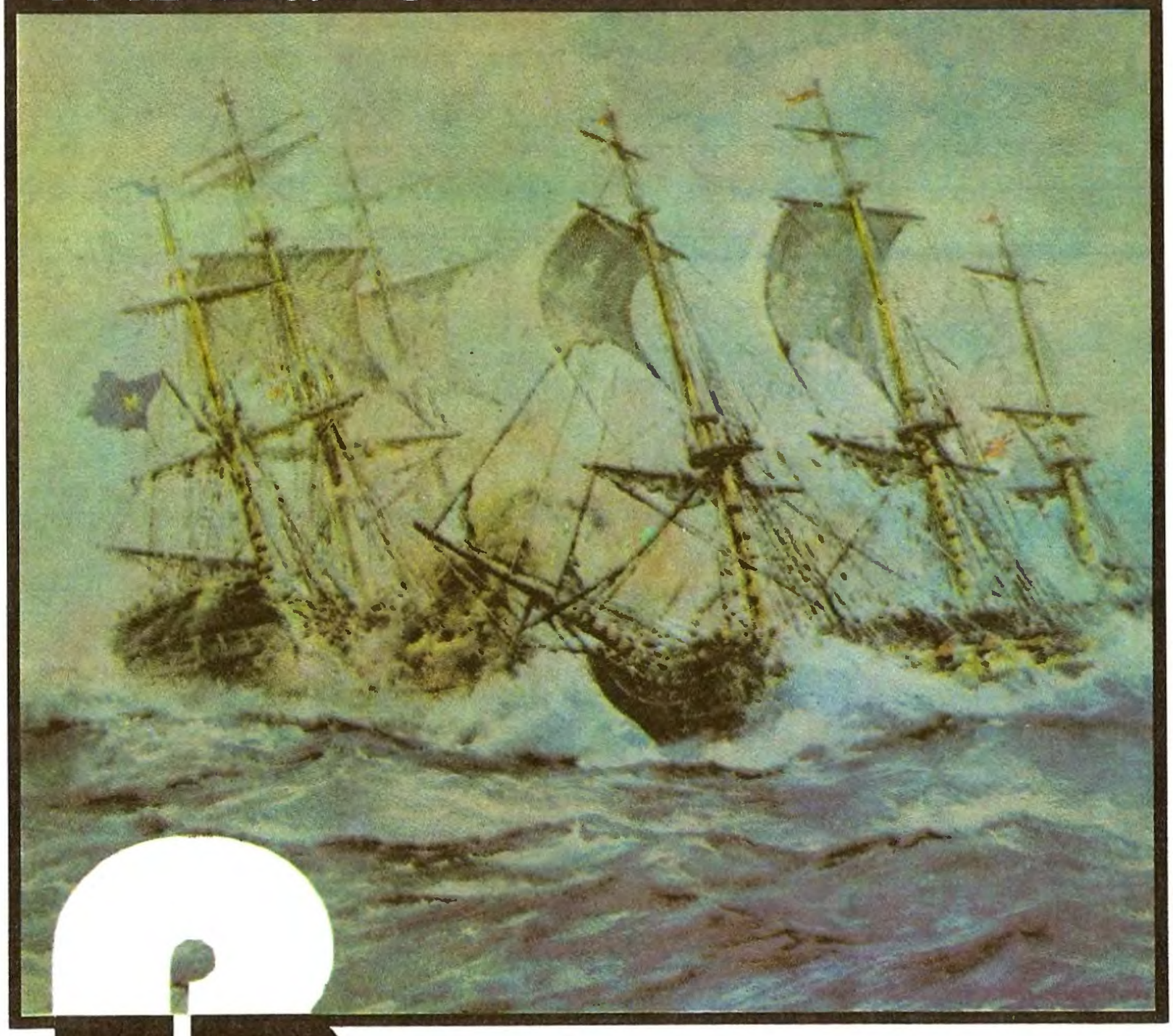
* * * * *

Na koniec coś dla użytkowników SpartaDOS X. Instrukcję **MOVE** zrealizuje następujący plik poleceń (najlepiej zapisać go pod nazwą **MOVE.BAT** i wywoływać jako — **MOVE źródło cel**):

```
COPY %1 %2
DEL %1
```

Przy wywołaniu **MOVE** możliwe jest pominięcie drugiego parametru, lecz pominięcie pierwszego spowoduje błąd. Przy przenoszeniu wszystkich plików zastosowanie specyfikacji *.* wymaga zatwierdzenia przez naciśnięcie klawisza ‹Y›. Można to ominąć podając specyfikację **?*.*.**

# VADEMECUM ATARI



Rys. Adam Werka

# BROADSIDES

**„Broadsides" jest symulacją bitwy pomiędzy okrętami żaglowymi z przełomu XVII i XVIII wieku przeznaczoną dla jednej lub dwóch osób.**

## GŁÓWNE MENU

‹1› — sterowanie: JOYSTICK — joystick
KEYBOARD — klawiatura
‹2› — liczba graczy: SOLITAIRE — jeden
TWO PLAYERS — dwóch
‹3› — rodzaj gry: ARCADE — zręcznościowa
TACTICAL — taktyczna
‹4› — rodzaj walki: SAILING — żegluga i abordaż
BOARDING ONLY — tylko abordaż
›5‹ — poziom gry: LEVEL — poziom (1—3)
PLAYER OPTION — dodatkowe pary (A—L)
‹6› — okręty: NO CHANGE — bez zmiany
CHANGE VARIABLES — zmiana parametrów
‹START› — rozpoczęcie gry (gdy 6 = NO CHANGE) lub parametry okrętów (gdy 6 = CHANGE VARIABLES); przed grą wybór prędkości (0 — najszybciej, 9 — najwolniej).

## PARAMETRY OKRĘTÓW

Wariant CHANGE VARIABLES umożliwia zmianę istniejących okrętów lub utworzenie własnych oraz zapisanie dokonanych zmian na dyskietce. Parametry są przedstawione w trzech obrazach.

‹START› — rozpoczęcie gry
‹OPTION› — następny obraz

### Obraz 1

‹1› — nazwa okrętu 1
‹6› — nazwa okrętu 2
‹2› — grupa dział okrętu 1
‹7› — grupa dział okrętu 2
‹3› — liczba dział w grupie dla okrętu 1
‹8› — liczba dział w grupie dla okrętu 2
‹4› — rodzaj dział w grupie dla okrętu 1
‹9› — rodzaj dział w grupie dla okrętu 2
‹5› — typowe parametry dla okrętu 1
‹0› — typowe parametry dla okrętu 2

W punktach 4 i 9 można wybrać następujące działa: zwykłe o wagomiarze 6, 8, 9, 12, 18, 24, 32, 36 i 42 funty oraz karonady (z literą „C") o wagomiarze 12, 18, 24, 32, 42 lub 68 funtów. Karonady mają mniejszy zasięg.

### Obraz 2

‹1› — czas ładowania dział dla okrętu 1
‹2› — czas ładowania dział dla okrętu 2
‹3› — czas wykonywania zwrotu dla okrętu 1
‹4› — czas wykonywania zwrotu dla okrętu 2
‹5› — maksymalna prędkość okrętu 1
‹6› — maksymalna prędkość okrętu 2
‹7› — siła niszcząca pocisków okrętu 1
‹8› — siła niszcząca pocisków okrętu 2
‹9› — wielkość kadłuba okrętu 1
‹0› — wielkość kadłuba okrętu 2

### Obraz 3

‹1› — celność strzelców okrętu 1
‹2› — celność strzelców okrętu 2
‹3› — liczebność załogi okrętu 1
‹4› — liczebność załogi okrętu 2
‹5› — straty podczas abordażu dla okrętu 1

# INSTRUKCJE

‹6› — straty podczas abordażu dla okrętu 2
‹7› — narodowość okrętu 1
‹8› — narodowość okrętu 1
‹CTRL›‹k› — zapis ustawienia na dyskietce (k = A-L)

Możliwe jest nadanie okrętom narodowości brytyjskiej (BRITISH), francuskiej (FRENCH), amerykańskiej (U.S.) lub innej (OTHER). Ustawienie zapisane na dyskietce odczytuje się wybierając z głównego menu odpowiedni wariant PI AYER OPTION.

# GRA

### Klawisze sterujące przebiegiem gry:

| | |
|---|---|
| przerwa w grze | ‹ESC› |
| nowa gra | ‹CTRL›‹R› |
| włączenie/wyłączenie dźwięku | ‹CTRL›‹S› |
| sterowanie z klawiatury | ‹CTRL›‹K› |
| sterowanie joystickami | ‹CTRL›‹J› |

## Żegluga

### Informacje obok mapy (od góry):

— nazwa okrętu,
— komenda lub meldunek,
— aktualne uzbrojenie,
— aktualna prędkość (SPEED),
— prędkość maksymalna (MX SPEED),
— rodzaj niesionych żagli (F = pełne),
— liczba punktów kadłuba (HULL),
— liczebność załogi (CREW).

### Informacje pod mapą:

— skala mapy w jardach (600 lub 2400),
— liczba punktów obu graczy (POINTS),
— siła i kierunek wiatru (WIND),
— aktualny czas symulowany.

UWAGI: Skala mapy zmienia się automatycznie zależnie od odległości między okrętami. Rysunek okrętu pokazuje odniesione uszkodzenia. Prędkość okrętu zależy od jego możliwości oraz od kierunku i siły wiatru. Po ustawieniu pod wiatr lub prawie pod wiatr prędkość spada do zera. Gra toczy się od godziny 6:00 rano do 6:00 po południu (w czasie symulowanym).

### Komendy dostępne podczas żeglugi:

| | |
|---|---|
| TURN TO PORT | — skręt w lewo |
| FASTER SPEED | — zwiększyć prędkość |
| STEADY SPEED | — utrzymać prędkość |
| SLOWER SPEED | — zmniejszyć prędkość |
| BACK SAIL | — żagle zredukowane |
| BATTLE SAIL | — żagle bojowe |
| FULL SAIL | — żagle pełne |
| SOLID SHOT | — kule pełne |
| CHAIN SHOT | — kule łańcuchowe |
| GRAPE SHOT | — kartacze |
| AIM AT SAILS | — celować w żagle |
| AIM AT HULL | — celować w kadłub |
| FIRE xxx YARDS | — ognia! na xxx jardów |
| TURN TO STRB | — skręt w prawo |

### Meldunki wyświetlane podczas żeglugi:

| | |
|---|---|
| AYE, AYE, SIR | — tak jest, sir |
| TURNING, SIR | — skręcamy, sir |
| STILL LOADING | — ładujemy |

### Klawisze sterujące okrętami:

| | okręt 1 | okręt 2 |
|---|---|---|
| poprzednia komenda | ‹D› | ‹+› |
| następna komenda | ‹F› | ‹*› |
| wykonanie komendy | ›S‹ | ‹;› |

## Abordaż

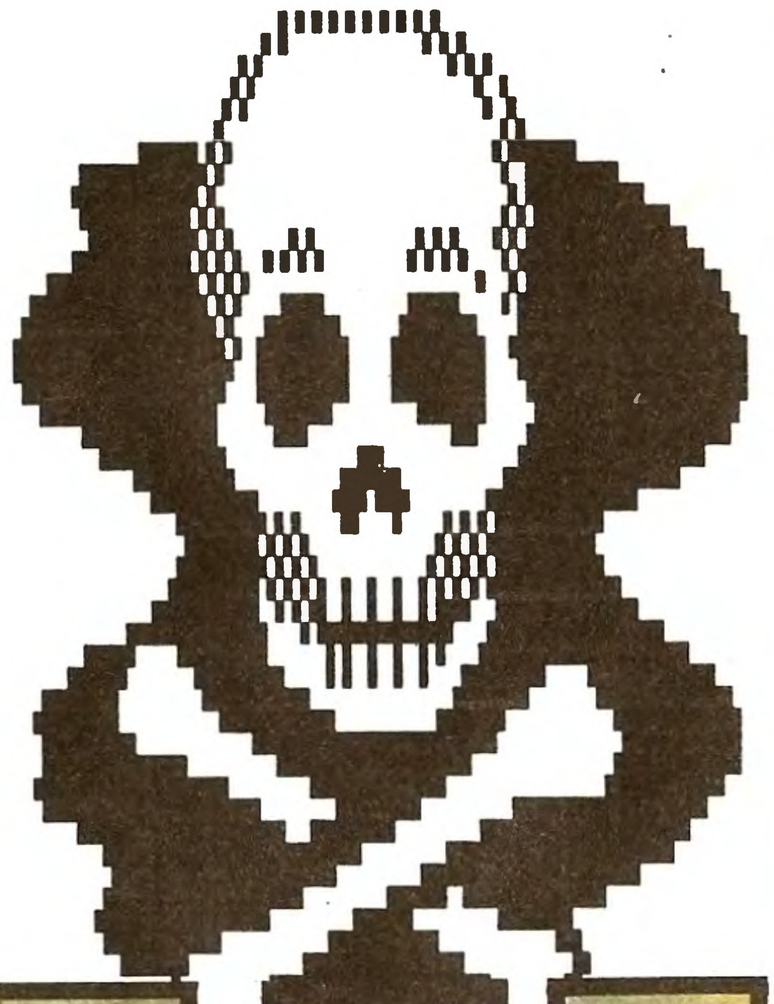
Abordaż trwa od chwili zetknięcia się okrętów do chwili przecięcia wszystkich lin łączących walczące okręty. W czasie abordażu następują kolejno po sobie fazy walki wręcz i ostrzeliwania przeciwnika.

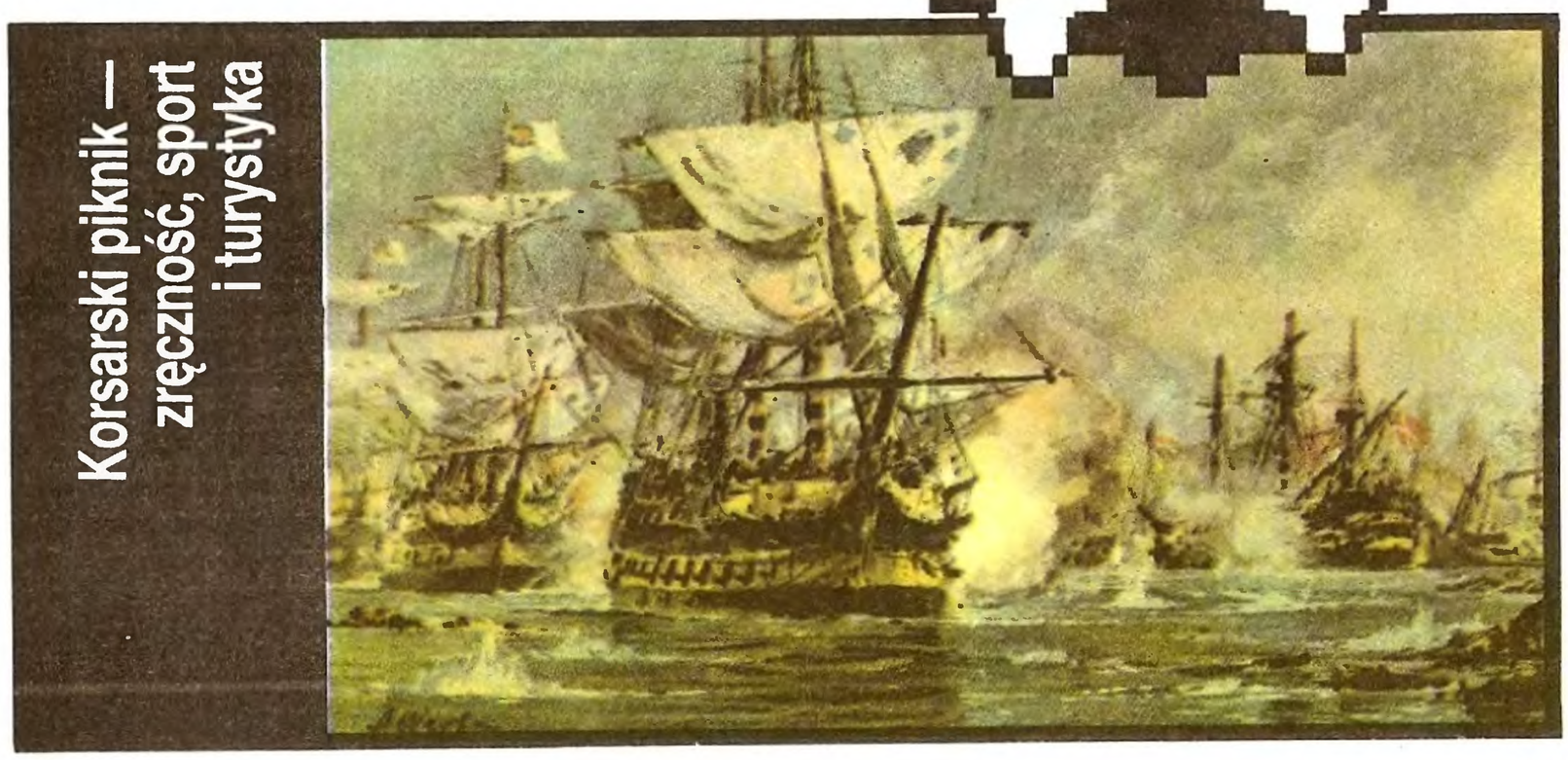
### Klawisze sterujące szermierzami:

| | okręt 1 | okręt 2 |
|---|---|---|
| ruch w prawo | ‹F› | ‹*› |
| ruch w lewo | ‹D› | ‹+› |
| krótkie pchnięcie | ‹1› | ‹8› |
| pchnięcie | ‹2› | ‹9› |
| cięcie | ‹3› | ‹0› |

### Klawisze sterujące strzelcami:

| | okręt1 | okręt 2 |
|---|---|---|
| celowanie do strzelców | ‹1› | ‹8› |
| celowanie na pokład wroga | ‹2› | ‹9› |
| celowanie na swój pokład | ‹3› | ‹0› |





Korsarski piknik — zręczność, sport i turystyka

# RECENZJA

**W tym roku przybyło w Polsce kilkadziesiąt tysięcy nowych właścicieli osmiobitowego Atari 65 lub 130 XE. Wielu z nich w pierwszych dniach będzie się interesować przede wszystkim grami. Ale gry prędzej czy później mogą się znudzić. Proponuję więc dać chwilę wypoczynku joystickowi.**

**Spróbujmy w inny sposób wykorzystać możliwości Atari. Najwięcej satysfakcji przyniesie nam własny program. Ułożenie poprawnie działającego programu nie jest wcale takie skomplikowane. Najtrudniejszą sprawą jest (niektórzy nie będą chcieli w to uwierzyć) dobry pomysł. Przemyślmy najpierw, co chcielibyśmy zaprogramować. Warto poświęcić na to trochę czasu, jeśli chcemy zrobić wrażenie na znajomych. Potem zaczniemy się martwić, jak przełożyć pomysł na język zrozumiały dla komputera.**

**Pierwsze kroki pomoże nam stawiać instrukcja obsługi oraz literatura fachowa. Nie warto oszczędzać wydatków na książki, jeśli chcemy szybko opanować zasady programowania. Dzięki nim unikniemy wielu błędów.**

**Jak okiełznać komputer radzi wszystkim nowicjuszom Wojciech Zientara w podręczniku „Podstawy programowania w Atari Basic", który niedawno wydało SOETO. Autora nie trzeba chyba bliżej przedstawiać. Jest on szefem klanu Atari i od niedawna redaktorem naczelnym naszego nowego pisma „Moje Atari". Przy takich rekomendacjach nie musimy się obawiać o fachowość książki.**

**„Podstawy programowania w Atari Basic" rozpoczynają oczywiście rozdziały, w których podane są wszystkie elementarne informacje: budowa i działanie komputera, system dwójkowy i szesnastkowy, jak zapisywać i odczytywać programy, układanie algorytmów, krótki opis funkcji i najważniejszych instrukcji języka Basic. Dowiadujemy się również, co to są procedury, do czego służą pętle, i jak możemy nauczyć komputer myślenia, czyli wyboru jednego z kilku wariantów programu. To już właściwie wystarczy, by poradzić sobie z najprostszymi problemami programowania.**

**Prawdziwe zadowolenie przyniesie nam dopiero umiejętność wykorzystania graficznych i dźwiękowych możliwości komputera. Temu zagadnieniu poświęcona jest dalsza część książki. Oprócz opisu instrukcji odpowiedzialnych za grafikę i dźwięk znajdziemy tu wiele gotowych procedur, które urozmaicą nasz program. Bardzo zainteresowały mnie przykłady efektów dźwiękowych. Osobiście sprawdziłem na moim 800 XL, jak brzmią odgłosy startującego helikoptera, działa laserowego, rakiety, wybuchu torpedy czy syreny wozu policyjnego. Podręcznik podaje również gotowy przepis na zaprogramowanie własnych znaków graficznych. Ci, którzy myślą o uruchomieniu własnej gry, muszą koniecznie zapoznać się z rozdziałem o grafice graczy i pocisków. Duszki mogą także urozmaicić niejeden program użytkowy.**

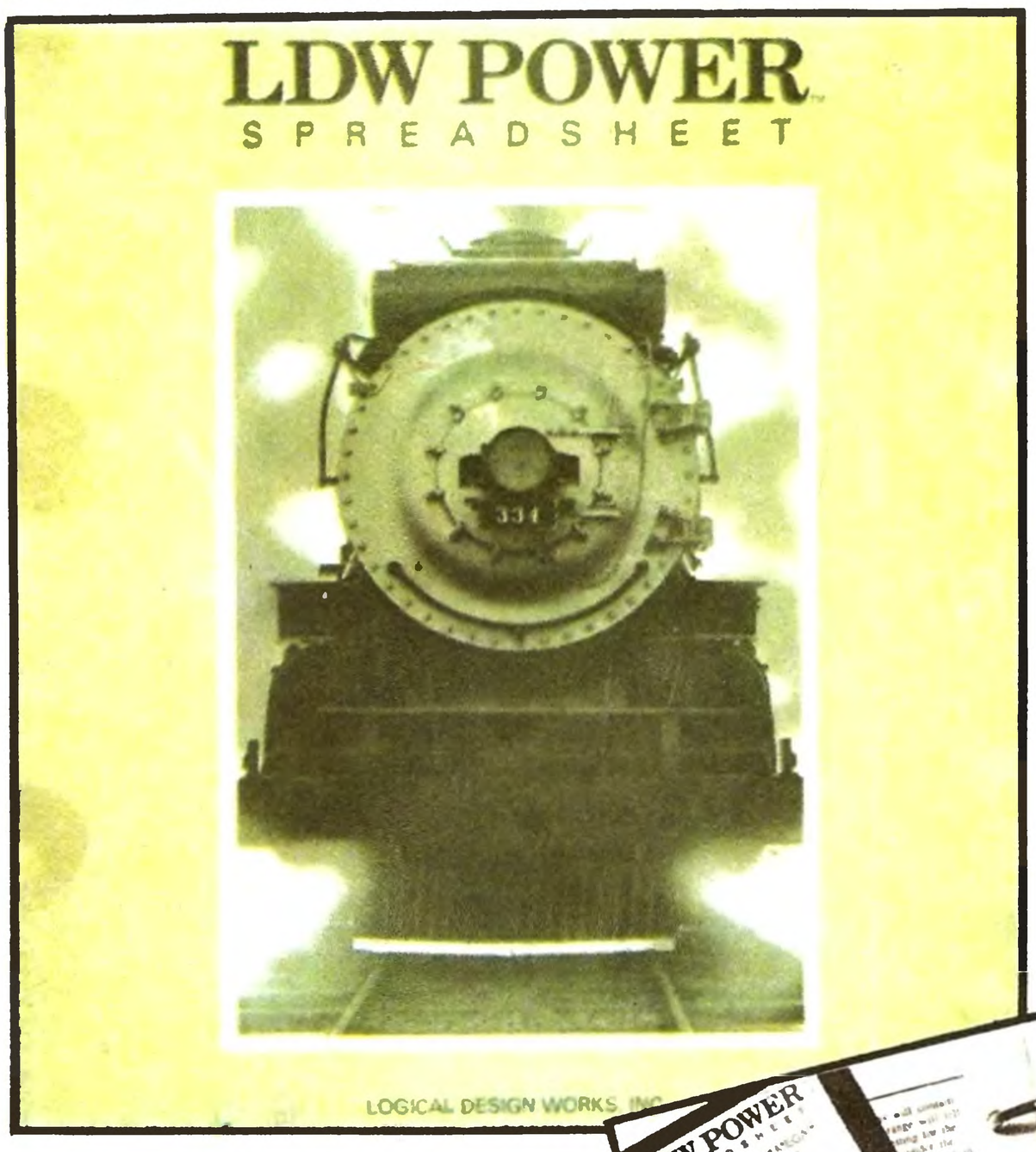
**Ważnym zagadnieniem jest ingerencja w pamięć komputera, czyli inaczej mówiąc wykorzystanie instrukcji PEEK i POKE. Przydają się one nie tylko do uzyskania nieśmiertelności w grach przygodowych. Nie zaszkodzi również, jeśli przyszły programista przeczyta rozdziały o manipulatorach — joysticku i paddle — oraz urządzeniach zewnętrznych. Bez opanowania umiejętności zaprogramowania podstawowych operacji odczytu i zapisu danych nie uda się ułożyć żadnego przyzwoitego programu użytkowego. Nie zaszkodzi więc uczciwie posiedzieć przy tej lekturze.**

**(J.J)**

„Podstawy programowania w Atari Basic", Wojciech Zientara, Stołeczny Ośrodek Elektronicznej Techniki Obliczeniowej, Warszawa 1990, Wyd. I, 170 stron

**Użytkowników 8-bitowych komputerów Atari mogą zainteresować również inne książki. Myślę, że każdy znajdzie w nich coś ciekawego dla siebie.**


1. Atari Basic, praca zbiorowa pod redakcją Wiesława Miguta, KAW 1987.
2. Assembler 6502, Jan Ruszczyc, SOETO 1987.
3. Poznajemy FORTH, Jan Ruszczyc, SOETO 1987.
4. Mapa pamięci Atari XL/XE — Podstawowe procedury systemu operacyjnego, Wojciech Zientara, SOETO 1988
5. Mapa pamięci Atari XL/XE — Procedury wejścia/wyjścia, Wojciech Zientara, SOETO 1988
6. Mapa pamięci Atari XL/XE — Procedury Interpretera Basica, Wojciech Zientara, SOETO 1989
7. Języki Atari XL/XE część 1–2, Wojciech Zientara, SOETO 1989
8. Poradnik programisty Atari, Wojciech Zientara, SOETO 1989
9. Poradnik użytkownika Atari XL/XE, Wojciech Zientara, SOETO 1990

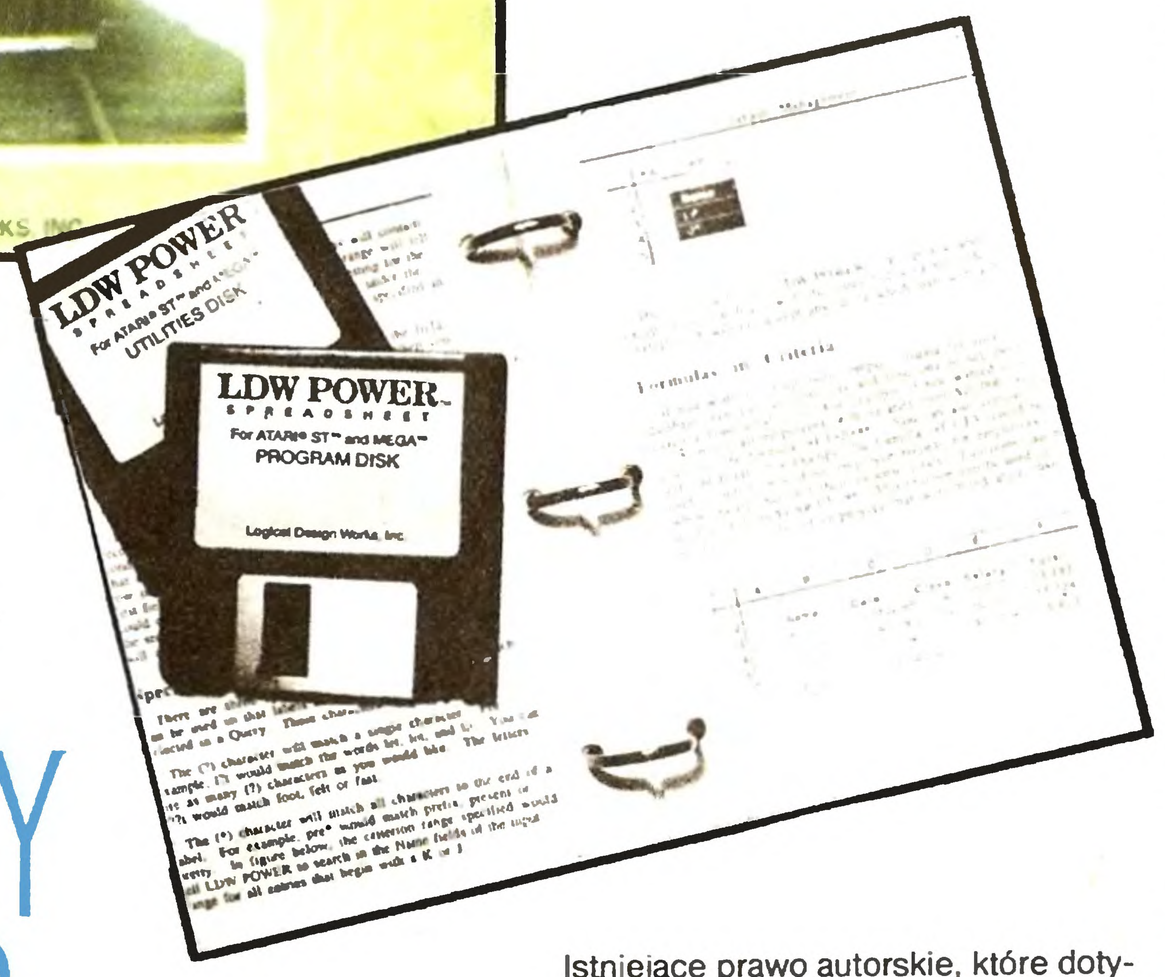



# JAK WYGLĄDA PRAWDZIWY PROGRAM?

**W odpowiedzi na takie pytanie przeciętny użytkownik komputera zaopatrujący się w programy na giełdzie po prostu wzruszy ramionami — „program nie wygląda, bo jest to coś zapisane na dyskietce".**

Rzeczywiście, tak się to przedstawia w przypadku kradzionych programów kupowanych od złodziei na giełdzie. Kopiują oni tylko na dyskietkę kilka plików i to wszystko. Niekiedy za dodatkową (słoną) opłatą można nabyć lichą trudno czytelną odbitkę kserograficzną instrukcji, na domiar złego napisaną w dziwnym języku będącym połączeniem polskiego, angielskiego, gwary komputerowej, młodzieżowej itd.

Taka sytuacja jest spowodowana brakiem w Polsce prawa autorskiego chroniącego programy komputerowe.




Istniejące prawo autorskie, które dotyczy ochrony publikacji, jest natomiast nagminne łamane. Niestety nikt w Polsce nie zwraca na to uwagi, żadna zaś szanująca się firma zagraniczna nie chce sprzedawać u nas swoich produktów.

Jak naprawdę wygląda program komputerowy? W jakiej postaci jest on sprzedawany w cywilizowanych krajach?

Na zamieszczonych obok zdjęciach pokazany jest taki „prawdziwy" program — arkusz kalkulacyjny „LDW Power" przeznaczony dla komputerów Atari ST i produkowany przez Logical Design Works.

Program zapakowany jest w pudełko z twardej tektury, oprawione w szare płótno i dodatkowo umieszczone w papierowej, lakierowanej obwolucie. Pudełko to ma 23,7 cm wysokości, 20,5 cm szerokości i 5,2 cm grubości. W pudełku znajduje się identycznie oprawiona okładka. W niej z kolei umieszczona jest właściwa zawartość.

Co wchodzi w skład programu? Przede wszystkim jest to szczegółowa, licząca 246 stron instrukcja, która napisana jest w sposób zrozumiały nawet dla zupełnego laika. Do tego dołączona jest tzw. Quick Reference Card, która opisuje w telegraficznym skrócie wszystkie polecenia programu. Na samym końcu instrukcji znajduje się koperta z przezroczystej folii, w której umieszczone są dwie dyskietki. Na jednej z nich zapisany jest właściwy program „LDW Power". Druga zawiera natomiast przykładowe arkusze kalkulacyjne, programy pomocnicze, ich opisy i przykłady oraz kroje liter programy umożliwiające druk na drukarkach Star i Atari (standardowo program współpracuje z drukarkami Epson).

Ponadto w pudełku jest jeszcze karta pocztowa, którą należy wypełnić i wysłać. Podaje się na niej swój adres, numer kopii programu i inne informacje. Każda taka karta jest rejestrowana u producenta, co daje możliwość bezpłatnego korzystania z porad w przypadku kłopotów. Rejestracja taka umożliwia również nabycie w przyszłości po znacznie obniżonej cenie nowej wersji programu.

Być może kiedyś, gdy i u nas zaczną obowiązywać cywilizowane przepisy, dorobimy się własnych programów w takiej właśnie postaci. Skończą się wtedy kłopoty, które powodują, że wielu użytkownikom komputer utrudnia pracę zamiast ją ułatwiać.

*Wojciech Zientara*

# Errata

**Pomimo naszych starań zdarzają się w „Moim Atari" błędy. Ich powodem jest najczęściej niestaranność składu w drukarni. Nie zmniejsza to jednak naszej winy, gdyż czytelnicy mają prawo żądać od redakcji dostarczania produktu najwyższej jakości. Przepraszamy więc za zaistniałe pomyłki i publikujemy sprostowania.**

## Moje Atari 1

### Okna raz jeszcze

W pierwszym listingu podwójnie zostały wydrukowane wiersze 190 i 200. Wiersz 190 jest wydrukowany jednakowo w obu miejscach, lecz w pierwszym wierszu 200 jest błąd — jego druga linia pochodzi z wiersza 180. Należy więc wpisać tylko drugi wiersz 200.

Niestety znacznie gorzej przedstawia się sprawa drugiego listingu. Jego części zostały w drukarni pomieszane (podobnie jak „MultiDOS"). Podanie poprawek jest niemożliwe. Jeżeli jednak czytelnicy wyrażą takie życzenie, to prawidłowa wersja zostanie wydrukowana w jednym z następnych numerów naszego pisma.

## Moje Atari 2

### MultiDOS

Podczas składania w drukarni poszczególne części listingu zostały pomieszane. Każdy czytelnik z pewnością zwrócił na to uwagę. Po dokładnej analizie można odczytać poprawną wersję programu, trzeba tylko zwrócić uwagę na to, że wiersze programu zostały podzielone. W związku z tym w miejscach, gdzie nastąpiło pomieszanie części, druga linia nie należy do wiersza poprzedniego, lecz do wiersza o numerze mniejszym o jeden niż wiersz następny. Błędy te występują w następujących wierszach: 146, 173, 254, 281, 308, 362, 389 i 416. Ostatni wiersz programu ma numer 428.

## Dyskietki

Okazuje się, że nawet posiadanie przez redakcję poprawnej, działającej wersji programu nie gwarantuje, że wersja ta zostanie wiernie wydrukowana. Najlepszym sposobem uniknięcia kłopotów jest nabycie w redakcji dyskietki zawierającej listingi **WSZYSTKICH** programów opublikowanych w danym numerze. Wychodząc naprzeciw zainteresowaniu czytelników wykonaliśmy również takie dyskietki zawierające programy, które były zamieszczone w „Moim Atari 1" oraz „Bajtkach": „Tylko o Atari 1" i „Tylko o Atari 2".

**Programów rysujących wykresy funkcji opublikowano w ciągu ostatnich lat kilka. Nie spotkałem jednak programu, który omijałby przedziały nieokreśloności funkcji oraz rysował wykresy funkcji wielokrotnie złożonych, np. y=f(x)=tan(3cosh(2arcsin(x))). Prezentowany program w Turbo Basicu jest pozbawiony tych wad.**

Na początku określamy przedział, dla którego będzie rysowany wykres. Jeżeli wartość funkcji w danym punkcie jest nieokreślona lub jeżeli wykracza ona poza dostępny w Atari przedział liczb, to punkt nie zostanie narysowany. Pierwszy rysowany wykres jest wykresem zadanej funkcji, drugi zaś wykresem jej pochodnej. Skalowanie osi jest orientacyjne i odnosi się tylko do wykresu funkcji. Wykres pochodnej ma wyłącznie znaczenie jakościowe.

Wciskając ‹RETURN› powracamy do podstawowej części programu, gdzie możemy zmienić parametry wykresu, a następnie — po pytaniu o funkcję — możemy:

— wpisać nową funkcję,
— nacisnąć ‹RETURN› bez wpisywania funkcji, co spowoduje rysowanie funkcji poprzednio wprowadzonej, z nowymi parametrami,
— nacisnąć ‹HELP› i ‹RETURN›, co umożliwia podawanie argumentów funkcji w celu otrzymania jej dokładnych wartości.

W trakcie obliczeń program wyłącza ekran.

W programie zostały użyte następujące skróty nazw funkcji:

| | | | |
|---|---|---|---|
| **LN** | — ln(x)=log_(x) | **ACOS** | — arccos(x) |
| **LOG** | — log10(x) | **ATN** | — arctg(x) |
| **SIN** | — sin(x) | **ACT** | — arcctg(x) |
| **COS** | — cos(x) | **SINH** | — sinh(x) |
| **TAN** | — tg(x) | **COSH** | — cosh(x) |
| **CTN** | — ctg(x) | **TNH** | — tgh(x) |
| **ASIN** | — arcsin(x) | **CTH** | — ctgh(x) |

*Jacek Tarasiuk*

```
UQ  1 REM WYKRESY FUNKCJI
KZ  2 REM Jacek Tarasiuk
IA  3 REM (c) 1990, Bajtek
NJ  4 REM
GB  5 REM INICJALIZACJA TABLIC, ZMIENNYCH
    I ADRESOW
JS  6 DIM Y(300),YP(300),S$(120),A$(80),B$
    (80),ZM$(4),TXT$(9),K$(1):S$="SIN(X)"
NI  7 NR=10:LN=878:LOGA=882:TN=886:CTN=890
    :ASIN=894:ACOS=898:ACT=902:SINH=906:MI
    N=0:MAX=0
ZX  8 XP=-1:XK=1:DX=(XK-XP)/300:ZEROY=180:
    ZEROX=10:GORA=1000000:YG=1000000:YD=-1
    000000
HN  9 COSH=910:TNH=914:CTNH=920:SIN=925:CO
    S=927:ATN=929:PUNKT=1:SIAT=0:EXP=931:P
    I=3.14159264:WAR=0
AV  10 GOSUB 950:GOSUB 2000:OSIE=0:POCH=1
YQ  20 IF WAR=0 THEN GOSUB 800
NC  30 GOTO 500
KB  499 REM OBLICZANIE PUNKTOW WYKRESU
GU  500 POKE 559,0:I=0
QH  501 I=I+1
NI  502 TRAP 532
OF  504 X=XP+I*DX:IF OSIE=0:GOSUB 1011:ELS
    E :Y=YP(I):ENDIF
EU  506 IF ABS(Y)>=GORA THEN Y=0.1234567
UR  508 IF Y>YG THEN Y=0.1234567
RW  510 IF Y<YD THEN Y=0.1234567
AM  512 Y(I)=Y
JG  514 IF Y>=MAX THEN MAX=Y
FM  516 IF Y<=MIN THEN MIN=Y
AY  518 IF I<300 THEN 501
YU  519 REM PARAMETRY WYKRESU. RYSUNEK.
NL  520 DY=(MAX-MIN)/180:IF YG<>1000000 OR
     YD<>-1000000 THEN DY=(YG-YD)/180
WH  521 IF POCH=1 THEN GOSUB 2100
KD  522 POKE 559,34:IF OSIE=0 THEN GRAPHIC
    S 8+16:COLOR 1:GOSUB 536:GOSUB 550:OSI
    E=1
PO  523 PCZ=0:FOR I=1 TO 300
OM  524 Y=INT((Y(I)-MIN)/DY):IF Y(I)<>0.12
    34567 AND PCZ=0 THEN PLOT I+10,180-Y:P
    CZ=1
SJ  525 IF Y(I)=0.1234567 THEN PCZ=0
UQ  526 IF Y(I)<>0.1234567 AND PUNKT=1 THE
    N PLOT I+10,180-Y
JP  527 IF Y(I)<>0.1234567 AND PUNKT=0 THE
    N DRAWTO I+10,180-Y
ZZ  528 NEXT I:IF POCH=1 THEN FOR J=1 TO 3
    00:Y(J)=YP(J):NEXT J:POCH=0:GOTO 500
JG  529 TEXT 0,185,"RETURN-powrot do pocza
    tku,ESC-KONIEC."
KR  530 GOSUB 1000:IF X=155 THEN 7
QY  531 IF ASC(K$)=27:END :ELSE :GOTO 530:
    ENDIF
IN  532 IF ERR=11 OR ERR=3 THEN Y=0.123456
    7:GOTO 512
FS  535 REM RYSOWANIE OSI
IH  536 ZEROX=10:ZEROY=180:IF XP*XK<0 THEN
     ZEROX=10-XP/DX
OD  538 IF MIN*MAX<0 THEN ZEROY=180+MIN/DY
LB  540 PLOT 10,ZEROY:DRAWTO 315,ZEROY:PLO
    T ZEROX,180:DRAWTO ZEROX,5
FQ  542 PLOT ZEROX-2,8:PLOT ZEROX-1,8:PLOT
     ZEROX+1,8:PLOT ZEROX+2,8:PLOT ZEROX-1
    ,7:PLOT ZEROX+1,7
MG  544 PLOT 313,ZEROY-2:PLOT 313,ZEROYZM1
    91:PLOT 313,ZEROY+1:PLOT 313,ZEROY+2:P
    LOT 314,ZEROY-1:PLOT 314,ZEROY+1
MS  546 TXT$=STR$(XP):TEXT 5,ZEROY+2,TXT$
IX  547 TXT$=STR$(XK):TEXT 300,ZEROY+2,TXT
    $
JJ  548 TXT$=STR$(MAX):TEXT ZEROX+3,2,TXT$
    :TEXT ZEROX+2,ZEROY+2,"0"
KG  549 TXT$=STR$(MIN):TEXT ZEROX+3,175,TX
    T$:RETURN
WX  550 FOR I=10 TO 310 STEP 20
PQ  552 IF SIAT=1 THEN FOR J=5 TO 180 STEP
     4:PLOT I,J:NEXT J
HM  554 FOR J=-2 TO 2:PLOT I,J+ZEROY:NEXT
    J:NEXT I
BA  555 FOR I=10 TO 180 STEP 20
YI  556 IF SIAT=1 THEN FOR J=10 TO 310 STE
    P 4:PLOT J,I:NEXT J
YU  557 FOR J=-2 TO 2:PLOT J+ZEROX,I:NEXT
    J:NEXT I
AJ  558 RETURN
VP  799 REM ANALIZA WPROWADZONEJ FUNKCJI
AC  800 POKE 559,0:REPEAT
OO  802 POWTORZ=0
KN  804 FOR I=1 TO LEN(A$)
FT  806 IF POWTORZ=1 THEN 812
KZ  808 IF A$(I,I)="(" THEN POZ1=I
IO  810 IF A$(I,I)=")" THEN POZ2=I:GOSUB 8
    20:POWTORZ=1
GG  812 NEXT I
GD  814 UNTIL POWTORZ=0
ON  816 CLS :POSITION 2,2:? 1001+NR;"Y=";A
    $:? "CONT":GOSUB 862:CLS :LIST 1000,10
    50
AE  818 RETURN
FP  820 NR=NR+1:ZM$(1,2)="ZM":ZM$(3,4)=STR
    $(NR)
ZY  822 CLS :POSITION 2,2:? 1000+NR;ZM$;"=
    ";A$(POZ1+1,POZ2-1)
SU  824 ? "CONT"
UW  826 POKE 842,13:POSITION 2,0
UQ  828 STOP
MN  830 POKE 842,12
OF  832 FUNK=0:X=3:IF A$(POZ1-2,POZ1-1)="L
    N" THEN GOSUB LN
FI  834 X=4:IF A$(POZ1-3,POZ1-1)="TAN" THE
    N GOSUB TN
RU  836 IF A$(POZ1-3,POZ1-1)="CTN" THEN GO
    SUB CTN
MQ  838 IF A$(POZ1-3,POZ1-1)="ATN" THEN GO
    SUB ATN
SQ  840 IF A$(POZ1-3,POZ1-1)="LOG" THEN GO
    SUB LOGA
DM  842 IF A$(POZ1-3,POZ1-1)="TNH" THEN GO
    SUB TNH
IG  844 IF A$(POZ1-3,POZ1-1)="ACT" THEN GO
    SUB ACT
NX  845 IF A$(POZ1-3,POZ1-1)="EXP" THEN GO
    SUB EXP
CW  846 X=5:IF A$(POZ1-4,POZ1-1)="ACOS" TH
    EN GOSUB ACOS
HG  848 IF A$(POZ1-4,POZ1-1)="ASIN" THEN G
    OSUB ASIN
KM  850 X=5:IF A$(POZ1-4,POZ1-1)="SINH" TH
    EN GOSUB SINH
LY  852 IF A$(POZ1-4,POZ1-1)="COSH" THEN G
    OSUB COSH
SF  853 X=4:IF A$(POZ1-3,POZ1-1)="COS" THE
    N GOSUB COS
GU  854 X=4:IF A$(POZ1-3,POZ1-1)="CTH" THE
    N GOSUB CTNH
UF  855 X=4:IF A$(POZ1-3,POZ1-1)="SIN" THE
    N GOSUB SIN
JJ  856 IF FUNK=1 THEN 860
UB  858 GOSUB 870:B$(1,POZ1-1)=A$(1,POZ1-1
    ):B$(POZ1,POZ1+3)=ZM$:B$(POZ1+4,POZ1+4
    +LEN(A$)-POZ2)=A$(POZ2+1):A$=B$
ZQ  860 RETURN
US  862 POKE 842,13:POSITION 2,0
UM  864 STOP
TN  866 POKE 842,12:RETURN
UH  868 NR=NR+1:ZM$(1,2)="ZM":ZM$(3,4)=STR
    $(NR):RETURN
RJ  870 B$="
                                     "
ZY  872 RETURN
SA  874 GOSUB 870:B$(1,POZ1-X)=A$(1,POZ1-X
    ):B$(POZ1-X+1,POZ1-X+4)=ZM$
NT  876 B$(POZ1+5-X,POZ1+5-X+LEN(A$)-POZ2)
    =A$(POZ2+1):A$=B$:FUNK=1:RETURN
KJ  878 GOSUB 868:POSITION 2,2:? 1000+NR;Z
    M$;"=";"LOG(";"ZM";NR-1;")":GOSUB 862
QG  880 GOSUB 874:RETURN
OP  882 GOSUB 868:POSITION 2,2:? 1000+NR;Z
    M$;"=";"CLOG(";"ZM";NR-1;")":GOSUB 862
QS  884 GOSUB 874:RETURN
XM  886 GOSUB 868:POSITION 2,2:? 1000+NR;Z
    M$;"=";"SIN(";"ZM";NR-1;")/COS(ZM";NR-
    1;")":GOSUB 862
RE  888 GOSUB 874:RETURN
NN  890 GOSUB 868:POSITION 2,2:? 1000+NR;Z
    M$;"=";"COS(";"ZM";NR-1;")/SIN(ZM";NR-
    1;")":GOSUB 862
QO  892 GOSUB 874:RETURN
EK  894 GOSUB 868:POSITION 2,2:? 1000+NR;Z
    M$;"=";"ATN(";"ZM";NR-1;"/SQR(1-(ZM";N
    R-1;"*ZM";NR-1;")))":GOSUB 862
RA  896 GOSUB 874:RETURN
IF  898 GOSUB 868:POSITION 2,2:? 1000+NR;Z
    M$;"=";"0.5*PI-ATN(";"ZM";NR-1;"/SQR(1
    -(ZM";NR-1;"*ZM";NR-1;")))":GOSUB 862
PR  900 GOSUB 874:RETURN
XC  902 GOSUB 868:POSITION 2,2:? 1000+NR;Z
    M$;"=";"0.5*PI-ATN(";"ZM";NR-1;")":GOS
    UB 862
QD  904 GOSUB 874:RETURN
OF  906 GOSUB 868:POSITION 2,2:? 1000+NR;Z
    M$;"=";"(EXP(";"ZM";NR-1;")-EXP(-ZM";N
    R-1;"))*0.5":GOSUB 862
QP  908 GOSUB 874:RETURN
IN  910 GOSUB 868:POSITION 2,2:? 1000+NR;Z
    M$;"=";"(EXP(";"ZM";NR-1;")+EXP(-ZM";N
    R-1;"))*0.5":GOSUB 862
PZ  912 GOSUB 874:RETURN
MG  914 GOSUB 868:POSITION 2,2:? 1000+NR;Z
    M$;"=";"(EXP(";"ZM";NR-1;")-EXP(-ZM";N
    R-1;"))/";
HO  916 ? "(EXP(";"ZM";NR-1;")+EXP(-ZM";NR
    -1;"))":GOSUB 862
QR  918 GOSUB 874:RETURN
GU  920 GOSUB 868:POSITION 2,2:? 1000+NR;Z
    M$;"=";"(EXP(";"ZM";NR-1;")+EXP(-ZM";N
    R-1;"))/";
JG  922 ? "(EXP(";"ZM";NR-1;")-EXP(-ZM";NR
    -1;"))":GOSUB 862
QH  924 GOSUB 874:RETURN
YL  925 GOSUB 868:POSITION 2,2:? 1000+NR;Z
    M$;"=";"SIN(";"ZM";NR-1;")":GOSUB 862
QN  926 GOSUB 874:RETURN
QG  927 GOSUB 868:POSITION 2,2:? 1000+NR;Z
    M$;"=";"COS(";"ZM";NR-1;")":GOSUB 862
QT  928 GOSUB 874:RETURN
MR  929 GOSUB 868:POSITION 2,2:? 1000+NR;Z
    M$;"=";"ATN(";"ZM";NR-1;")":GOSUB 862
PX  930 GOSUB 874:RETURN
EF  931 GOSUB 868:POSITION 2,2:? 1000+NR;Z
    M$;"=";"EXP(";"ZM";NR-1;")":GOSUB 862
QD  932 GOSUB 874:RETURN
PW  949 REM PLANSZA
PN  950 GRAPHICS 0:POKE 752,3:POKE 82,0
PX  951 POSITION 0,0:? "┌──────────┬──────────
    ┬────┬──────┐"
XJ  952 POSITION 0,1:? "│xp=          │xk=
          │punkty│ciagly │"
BF  953 POSITION 0,2:? "├──────────┼──────────
    ┼─┬──┴──────┤"
VM  954 POSITION 0,3:? "│yp=          │yk=
          │<│automatycznie│"
HH  955 POSITION 0,4:? "├──────────┴──────────
    ┴─┴─────────┤"
UQ  956 POSITION 0,5:? "│Jezeli chcesz obl
    iczyc wartosc                  │"
RF  957 POSITION 0,6:? "│funkcji w okreslo
    nym punkcie                    │"
YY  958 POSITION 0,7:? "│przed podaniem fu
    nkcji wcisnij Help             │"
WE  959 POSITION 0,8:? "├─────────┬───────────
    ┬──────────────┤"
EA  960 POSITION 0,9:? "│z siatka│bez siat
    ki│wartosci funkcji│"
AT  961 POSITION 0,10:? "└─────────┴──────────
    ┴──────────────┘"
JQ  962 POSITION 0,14:? "┌─────────────────────
    ──────────────┐"
CE  963 FOR I=15 TO 22:POSITION 0,I:? "│
                                   │";:
    NEXT I
OD  964 POSITION 0,23:? "└─────────────────────
    ──────────────┘";:RETURN
WO  999 REM ODCZYT KLAWIATURY
JT  1000 OPEN #1,4,0,"K:":GET #1,X:K$=CHR$
    (X):CLOSE #1:RETURN
DE  1011 ZM11=X
FA  1012 ZM12=SIN(ZM11)
NZ  1013 ZM13=ZM12
FX  1014 ZM14=COS(ZM13)
BS  1015 Y=ZM14
AO  1050 RETURN
JZ  1999 REM WPROWADZANIE DANYCH
JH  2000 POSITION 4,1:? "          ":POSITION
     15,1:? "          ":POSITION 1,15:? "Wpr
    owadz poczatek xp i koniec xk        "
CM  2001 POSITION 1,16:? "przedzialu x-ow.
                                     "
SK  2002 POSITION 1,17:? "Jesli wcisniesz
    Return przyjety zo-                  "
RE  2003 POSITION 1,18:? "stanie przedzial
    [-1,1].                              "
NZ  2004 TRAP 2006:POSITION 2,11:INPUT "xp
    =",XP:POSITION 2,13:INPUT "xk=",XK:GOT
    O 2007
YL  2006 XP=-1:XK=1
TR  2007 POSITION 4,1:? XP:POSITION 15,1:?
     XK:POSITION 2,13:? "          ":POSIT
    ION 2,11:? "          ":GOSUB 962
WR  2008 POSITION 4,3:? "          ":POSITION
     15,3:? "          ":POSITION 1,15:? "Wpr
    owadz poczatek yp i koniec yk        "
LU  2009 POSITION 1,16:? "przedzialu y-kow
    .                                    "
FD  2010 POSITION 1,17:? "Gdy wcisniesz Re
    turn, przeciwdziedzi-"
TZ  2011 POSITION 1,18:? "na zostanie wyzn
    aczona automatycznie,"
QR  2012 POSITION 1,19:? "w zaleznosci od
    przebiegu wykresu.                   "
VL  2013 TRAP 2017:POSITION 2,11:INPUT "yp
    =",YD:POSITION 2,13:INPUT "yk=",YG
JL  2015 POSITION 4,3:? YD:POSITION 15,3:?
     YG:POSITION 2,13:? "          ":POSIT
    ION 2,11:? "          ":GOSUB 962
IN  2016 POSITION 23,3:? "K│automatycznie"
    :GOTO 2018
VG  2017 POSITION 2,11:? "          ":YD=-
    1000000:YG=1000000:GOSUB 962
JG  2018 REM
SZ  2030 POSITION 1,15:? "Jesli chcesz wyk
    res punktowy wcisnij "
TC  2031 POSITION 1,16:? "klawisz -P-, jes
    li chcesz ciagly -C-."
HW  2032 GOSUB 1000:IF K$="P" THEN PUNKT=1
    :POSITION 23,1:? "punkty│ciagly "
LC  2033 IF K$="C" THEN PUNKT=0:POSITION 2
    3,1:? "punkty│ciagly "
DL  2034 GOSUB 962
LR  2035 POSITION 1,15:? "Jesli chcesz wyk
    res z siatka wcisnij "
AJ  2036 POSITION 1,16:? "klawisz -s-,jesl
    i bez Return.                        "
GE  2037 GOSUB 1000:IF K$="S" THEN SIAT=1:
    POSITION 1,9:? "z siatka│bez siatki":G
    OSUB 962:GOTO 2039
SU  2038 SIAT=0:POSITION 1,9:? "z siatka│b
    ez siatki":GOSUB 962
QO  2039 POSITION 1,15:? "^-potega   SIN  A
    SIN   SINH "
GK  2040 POSITION 1,16:? "LN         COS  A
    COS   COSH "
ZA  2041 POSITION 1,17:? "LOG        TAN  A
    TN    TNH  "
ER  2042 POSITION 1,18:? "           CTN  A
    CT    CTH  "
HG  2043 POSITION 1,19:? "RETURN -funkcja
    poprzednia"
LA  2049 POSITION 0,11:? "f(x)=";:INPUT B$
    :IF B$="" THEN B$=S$
VA  2050 S$=B$:A$(1,4)="    ":A$(5)=B$:A$
    (LEN(B$)+5)="  ":? B$:GOSUB 870:DEL 10
    05,1049:NR=10
RL  2052 IF PEEK(732)=17 THEN POKE 732,0:G
    OSUB 2200
ZB  2055 DX=(XK-XP)/300:NR=10:RETURN
CW  2099 REM OBLICZANIE POCHODNEJ
ZE  2100 FOR J=3 TO 297
DE  2105 IF Y(J-2)=0.1234567 OR Y(J+2)=0.1
    234567 THEN YP(J)=0:GOTO 2110
BP  2108 YP(J)=2*Y(J+2)+Y(J+1)-Y(J-1)-2*Y(
    J-2)/(10*DX)
GO  2110 NEXT J:RETURN
TT  2199 REM OBLICZANIE WARTOSCI FUNKCJI
FN  2200 GOSUB 800:GRAPHICS 0:GOSUB 950:WA
    R=1
RV  2201 POSITION 1,16:? "Wprowadz argumen
    t. Argumenty dla                     "
NM  2202 POSITION 1,17:? "ktorych, funkcja
     jest nieokreslona                   "
MU  2203 POSITION 1,18:? "zostana zignorow
    ane. Wykres - RETURN "
VQ  2205 POSITION 0,11:? "
                                     ";
TJ  2206 POSITION 0,11:? "X=";:TRAP 2210:I
    NPUT X:TRAP 2205:GOSUB 1011:POSITION 1
    5,11:? "f(x)=":Y:GOSUB 1000:GOTO 2205
AH  2210 RETURN
```

URATUJ ŚWIAT
PRZED OBCYMI NAJEŹDŹCAMI

XL, XE, ST

# HAWKQUEST

W WIEK XXI
LUDZKOŚĆ WKRACZAŁA
Z NADZIEJĄ.
PLON OSTATNIEGO
DZIESIĘCIOLECIA:
OTWARCIE GRANIC.
ROZWIĄZANIE PROBLEMÓW
GŁODU W TRZECIM ŚWIECIE,
WYKORZYSTANIE ENERGII
ATOMOWEJ W CELU
ROZWIJANIA TRANSPORTU
I PRZEMYSŁU NA CAŁYM
GLOBIE, ALE NADE WSZYSTKO
POWSZECHNE
I NIEODWOŁALNE ROZBROJENIE —
NAPAWAŁ UFNOŚCIĄ
I POZWALAŁ WIERZYĆ
W RYCHŁE SPEŁNIENIE
MARZEŃ POPRZEDNICH
POKOLEŃ O POKOJOWEJ
WSPÓŁPRACY WSZYSTKICH
PAŃSTW W CELU ZAPEWNIENIA
MIESZKAŃCOM ZIEMI
SPOKOJU I DOBROBYTU
NA WIELE, WIELE LAT.

Zagospodarowano Antarktydę — kontynent dotychczas wielce niegościnny dla kolonizatorów, bliska była chwila zwycięskiego zakończenia walki z wirusami atakującymi ludzi, znaczne obszary świata były już zdemilitaryzowane, trwała likwidacja ostatnich baz wojskowych na terenie USA.

Jednak największym osiągnięciem schyłku lat dziewięćdziesiątych było zapanowanie nad zjawiskiem od stuleci bezlitośnie i bezkompromisowo spędzającym sen z oczu mieszkańcom wszystkich kontynentów.
W roku 1998 na odludnym płaskowyżu w głębi Australii zakończono budowę Ośrodka Zapobiegania Trzęsieniom Ziemi. Stało się to możliwe dzięki skonstruowaniu superkomputera o olbrzymiej mocy obliczeniowej, który analizował i przetwarzał dane pochodzące ze specjalnych czujników porozmieszczanych na powierzchni całej planety. System ten oparty był na specjalnym 128-bitowym procesorze, którego newralgiczny punkt stanowił kryształ krzemu, będący produktem wieloletniego realizowania skomplikowanej technologii produkcji i praktycznie niemożliwy do odtworzenia. W chwili umieszczenia go w jednym z zespołów komputera był jedynym egzemplarzem na Ziemi...

* * *

Ośrodek pracował z górą trzy lata i koszmar dręczący 6 miliardów ludzi stał się już tylko wyeliminowanym fragmentem minionej przeszłości. Lecz to, co nastąpiło pewnej letniej nocy, było zburzeniem całego ludzkiego pojęcia o bezpieczeństwie ich rodzimej planety. Oddział kosmicznych najeźdźców w kolorze blue najbezczelniej w świecie zaatakował centrum ośrodka i po upływie godziny na ekranach radarów można było jedynie obserwować oddalający się krążownik galaktyczny z kryształem na pokładzie.

W siedzibie ONZ w Genewie zawrzało. Wyłączenie ośrodka z pracy groziło, w razie trzęsienia ziemi, uszkodzeniami licznych reaktorów atomowych, działających we wszystkich zakątkach kuli ziemskiej. W perspektywie dawało to wielotysięczne straty w ludziach i nieodwracalne skażenie środowiska na niespotykaną dotąd skalę.

W tym samym mniej więcej czasie, kiedy wielcy tego świata radzili nad możliwymi sposobami ratunku, a mali ukryci w domach oczekiwali apokalipsy, w jednym z podziemnych bunkrów w Moskwie ktoś wytrwale mocował się z zardzewiałą pokrywą włazu. Przez jego twarz przemknął wyraz zadowolenia, kiedy huk otwieranej pokrywy rozszedł się zwielokrotniony stutysięcznym echem po nieskończonych, niskich korytarzach. Zapalił latarkę. Wąziutki snop światła ukazał niewielkie pomieszczenie, które w połowie wypełniał potężny sejf. Pomimo iż przewidywano jego wytrzymałość na przetrwanie wybuchu jądrowego, po z górą pięćdziesięciu latach otwarcie go nie nastręczało trudności.

Po blisko 20 minutach poszukiwań tajemniczy intruz opuścił bunkier niosąc ze sobą starą, pokrytą grubą warstwą kurzu teczkę. Mimo niepozornego wyglądu zawierała ona materiał, który z pewnością nie należał do kategorii opowiadań dla dzieci. Był to ostatni projekt Sikorskiego, stworzony przed wyjazdem ze Związku Radzieckiego — superśmigłowiec bojowy, przy którym Airwolf i Błękitny Grom nadają się co najwyżej do opylania pól w celu ochrony przed stonką ziemniaczaną. Maszyna ta została wyposażona w najnowocześniejsze systemy gwarantujące jej przetrwanie najzacieklejszych ataków wroga. Nieograniczony zasięg śmigłowca zapewnia atomowy napęd „nie do zdarcia", a jedyną broń stanowi absolutnie pewne i skuteczne działko 76 mm. Słowem, jest to w pełni profesjonalny i niezawodny sprzęt służący tylko jednemu: sianiu kompletnego zniszczenia. Sprzęt, który poprowadzony przez odpowiedniego pilota, może uratować Ziemię od zagłady.

Zapewne domyślasz się, Czytelniku, że dokładnie w tym miejscu rozpoczyna się epopeja Twoja i stalowego jastrzębia, którego musisz poprowadzić do zwycięstwa. Przed tobą wiele godzin walki na śmierć i życie o przetrwanie rasy ludzkiej. Na planetach systemu Kiross — bazy wypadowej wstrętnych najeźdźców — rozrzucone zostały fragmenty skradzionego kryształu, które powinieneś zebrać, by móc powrócić do domu. Nie ukrywam, że jest to zadanie z gatunku bardzo trudnych, możliwe do zrealizowania jedynie przy najwyższym poświęceniu. W boju, Twoim sojusznikiem będzie tylko zestaw podstawowych wiadomości o każdej z wrogich planet i jastrząb — poszukiwacz, z którym stanowić będziesz jedność w śmiertelnym starciu. Niech moc będzie z Tobą, bo nasze życie zależy od Ciebie!

System Kiross — opracowany wg. raportu wywiadu nr 234/II — 22

**KIROSS** — karłowata gwiazda typu G1

**SOREA** — masa 1.1, średnica 9980 km, doba 30 h, klimat nieokreślony, 16 premii, 145 dział naziemnych, poziom 5.

**MYCEA** — masa 0.9, średnica 8460 km, doba 23 h, klimat umiarkowany, 41 premii, 160 dział naziemnych, poziom 1.

**ANCOONA** — masa 1.6, średnica 39000 km, doba 52 h, klimat wodny, 48 premii, 120 dział naziemnych, poziom 2.

**CYTRON** — masa 0.6, średnica 1500 km, doba 12 h, klimat sztuczny, 29 premii, 130 dział naziemnych, poziom 4.

**XAWIER** — masa 2.1, średnia 51080 km, doba 70 h, klimat tropikalny, 34 premie, 120 dział naziemnych, poziom 3.

*Master*

Producent: Red Rat Software

**Ocena (w skali 1-10):**

| | |
|---|---|
| **GRAFIKA** | **8** |
| **DŹWIĘK** | **7** |
| **SENS GRY** | **8** |

# DRUID

XL, XE

DO KRÓLESTWA BELORN
PRZYBYLI CZTEREJ KSIĄŻĘTA
CIEMNOŚCI.
ICH PRZYBYCIE ZAKŁÓCIŁO
RÓWNOWAGĘ MIĘDZY
SIŁAMI DOBRA I ZŁA.

●

W królestwie zapanował strach i zniszczenie. Teraz ty, jako ostatni z żyjących druidów, masz zmierzyć się ze złem.

Musisz zniszczyć wszystkich książąt ciemności. Dokonasz tego wędrując przez osiem poziomów gry, gdzie przyjdzie ci jeszcze zmierzyć się ze złymi duchami, żukami, wężami i innymi potworami, których dotknięcie kosztuje cię stratę cennej energii. Możesz ją uzupełniać wchodząc na jeden z pentagramów życia. Do obrony masz czary proste i specjalne. Uzupełnianie czarów odbywa się w specjalnych skrzynkach. Aby otworzyć skrzynkę, należy na nią wejść i następnie wybrać joystickiem odpowiednie zaklęcie i nacisnąć FIRE. Z każdej skrzyni można korzystać tylko raz.

Zaklęcia proste to: WATER, FIRE i ELECTRICITY. Czary specjalne to: INVISIBILITY, GOLEM i CHAOS. W zależności od potwora, z jakim walczysz, dobieraj odpowiednich czarów i zaklęć.

Informacje wyświetlane na ekranie to:
— sznur po napisie DRUID oznaczający ilość energii,
— sznur z napisem GOLEM ukazuje się po użyciu odpowiedniego czaru i przedstawia energię Golema,
— sznur po napisie RATING przedstawia wtajemniczenie Druida (wynik),
— liczby pod symbolem czaru oznaczają ilość zaklęć.

Jak już wspomniałem, gra odbywa się na ośmiu poziomach połączonych ze sobą schodami.

Zaklęcia proste rzuca się używając FIRE w joysticku. Do czarów specjalnych służą klawisze:

2- INVISIBILITY (niewidzialność) — unieruchamia wszystkie potwory z wyjątkiem śliny książąt ciemności,

3- GOLEM — na ekranie pojawia się stwór niszczący potwory kosztem swojej energii. Golemem steruje się za pomocą joysticka i klawisza „A". Można wybrać jedną z trzech możliwości:

— **WAIT** — Golem stoi w miejscu i niszczy tylko te potwory, które na niego wejdą,
— **FOLLOW** — Golem porusza się w rownolegle do Druida i służy za jego osłonę,
— **SEND** — Golem porusza się w kierunku, w którym zwrócony jest Druid (Druidax nie można zatrzymać).

Wyboru wariantu dokonujemy klawiszem „A"

4- CHAOS — Działanie tego czaru to:
— odtworzenie początkowego stanu energii
— stworzenie osłony, ktora pozwala Druidowi na poruszanie się w bliskości książąt ciemności bez straty energii
— umożliwienie zakończenia gry

**Klawisze używane w grze:**
1 — otwieranie drzwi
2 — zaklęcie INVISIBILITY
3 — zaklęcie GOLEM
4 — zaklęcie CHAOS
A — sterowanie Golemem
H — przerwa w grze
P — zmiana aktualnego zaklęcia prostego.

*Maczek*

Producent: Firebird
Autor: John Croudy

| Ocena (w skali 1–10): | |
|---|---|
| GRAFIKA: | 9 |
| DŹWIĘK: | 5 |
| SENS: | 7 |

# TRAILBLAZER

XL, XE

PROGRAM TEN DOBRZE ZNAJĄ POSIADACZE COMMODORE C-64 POD TYTUŁEM SOUT. ZABAWA POLEGA NA STEROWANIU PIŁKĄ NA TRASIE ZAWIESZONEJ GDZIEŚ W KOSMOSIE.

●

W grze może uczestniczyć jeden lub dwóch graczy, ściągając się wzajemnie lub na czas.

Można wybrać trzy dowolne z 21 dostępnych tras (od A do V). Piłką steruje się joystickiem, ale gracz pierwszy zawsze może się zdecydować na korzystanie z klawiatury: „z" w lewo, „X" w prawo, „SPACE BAR" skok.

Trasa zbudowana jest z różnokolorowych prostokątów. Postawienie piłki na każdym z nich wywołuje różne efekty. I tak stanięcie na:
czerwonym — zmniejsza prędkość piłki,
czarnym — to pułapka, którą należy omijać,
zielonym — zwiększa prędkość piłki,
purpurowym — zmienia kierunek sterowania (wychylenie joysticka w lewo powoduje skręcenie piłki w prawo i odwrotnie),
polu migającym — na kilka sekund nadaje piłce superprędkość.

Po wczytaniu gry należy wcisnąć FIRE. Ukaże się plansza. W górnej części ekranu, jest pokazana przykładowa trasa. W środkowej, po lewej stronie zapisane są najlepsze wyniki, a obok najlepszy czas przejazdu dla każdej z tras. W dolnej części ekranu ukazuje się pięć różnych wariantów gry do wybrania joystickiem:
1. wyścig dla jednego gracza (ARCADE 1 player),
2. wyścig dla dwóch graczy (ARCADE 2 players),
3. wyścig na czas dla gracza I (TRIAL 1 player),
4. wyścig na czas dla dwóch graczy (TRIAL 2 players),
5. wyścig z komputerem (player and robot).

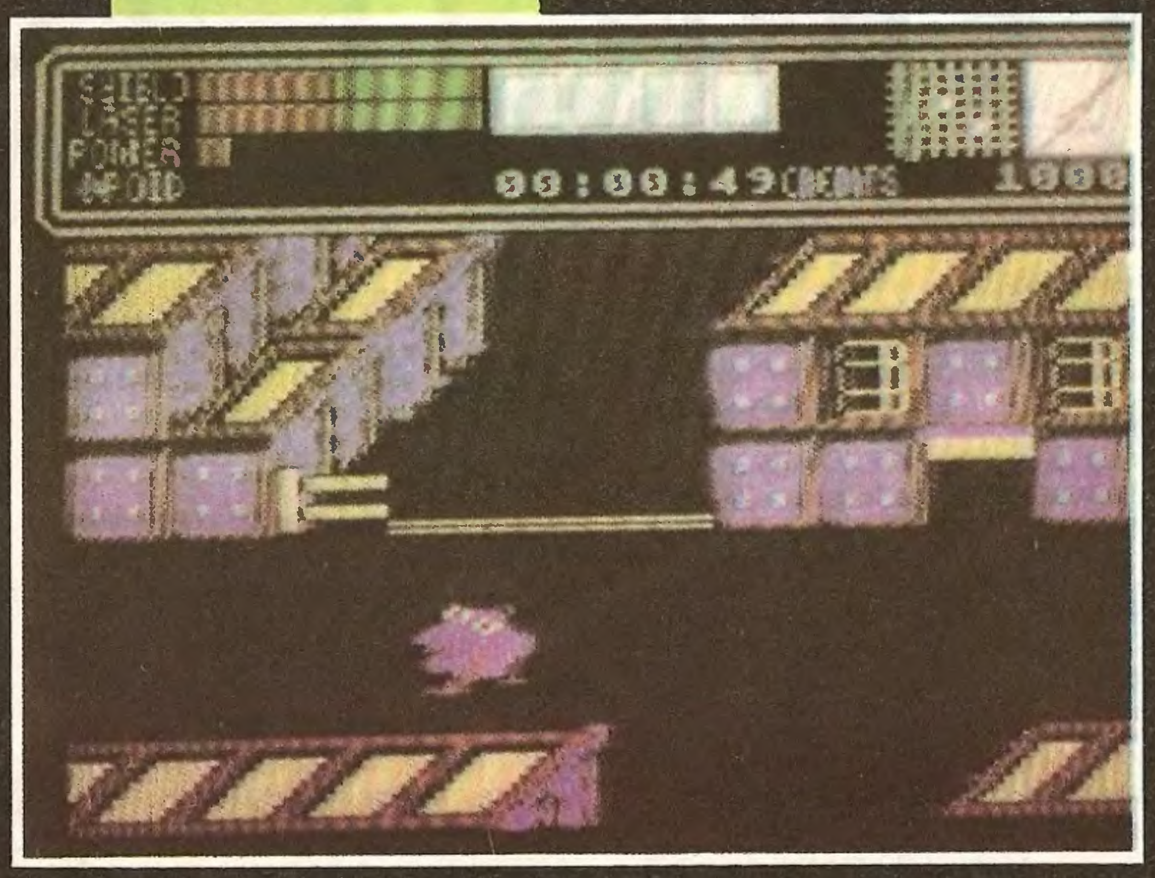
Podczas gry w wariantach 1 i 2 można oddać tylko 7 skoków na każdej trasie, a każda z nich ograniczona jest limitem czasowym. Trzy ostatnie warianty umożliwiają wybranie trzech z dwudziestu jeden tras. Można oddać dowolną ilość skoków, nie ma też limitu czasowego.

*Maczek*

Producent: Gremlin Graphics
Autor: A. Hoque

| Ocena (w skali 1–10): | |
|---|---|
| GRAFIKA: | 6 |
| DŹWIĘK: | 5 |
| SENS: | 7 |

## XL, XE




JESTEŚ OSADNIKIEM NA ODLEGŁEJ PLANECIE. TWOJA KOLONIA JEST OGRODZONA, E NIE PRZESZKADZA TO NIESFORNYM ŻYJĄTKOM, KTÓRE FORSUJĄ OGRODZENIE I ZJADAJĄ ROŚLINY.

# COLONY

Twoje zadanie zatem polega na uratowaniu ogrodzenia i plonów.

Kolonię otaczają trzy rodzaje ogrodzeń:
— najsilniejsze to siatka stalowa (steel fence)
— średnie to drewniany płot (wooden fence)
— najsłabsze to drut kolczasty (barbed wire)

Uszkodzone ogrodzenie trzeba zanieść do odpowiedniego magazynu, oddać (DROP) i pobrać nowe (PICK UP).

Gdy będziesz potrzebował jakiejś rzeczy, musisz udać się do centrum kontroli i dokonać zamówienia. Pojawi się informacja, że statek już wyruszył (SHIP EN ROUTE). Na przybycie statku (SHIP ARRIVED) musisz zapalić światła (ACTIVE BEACON), w przeciwnym razie transportowiec nie wyląduje.

Odwiedzając magazyny możesz zostawić popsute ogrodzenie i pobrać nowe. W magazynie przedmiotów organicznych możesz pozostawić w depozycie plony (DEPOSIT MUSHROOMS). W centrum kontroli zapalasz i gasisz oświetlenie lądowiska (ACTIVITE i DEACTIVITE). Zamawiając poszczególne przedmioty musisz pamiętać, że jesteś ograniczony finansowo. Twój kredyt to 1000 sztuk złota. Można zamówić drugiego robota (SPARE DROID BODY), ale kosztuje on 1000 sztuk złota, a potrzebne jest jeszcze drugie tyle złota na system sterowania.

**EKRAN**

**W górnej części ekranu znajduje się tablica rozdzielcza, a na niej, wskaźnik:**
**— SHIELD — poziom pola siłowego twojego robota**
**— LASER — energia lasera**
**Gdy któryś ze wskaźników znajdzie się blisko zera, musisz się udać do pomieszczenia ładowania robota.**
**— POWER — wskaźnik energii głównego generatora,**
**— cztery wskaźniki pokazujące co niesie robot,**
**— zegar i wielkość kredytu,**
**— wskaźnik ETA sygnalizujący obecność statku transportowego oraz stan radiolatarni.**
**Dwa duże kwadraty to radar wskazujący położenie robota na mapie kolonii oraz wskaźnik trybu pracy robota (symbol lasera oznacza walkę, ręki z próbówką — pracę). Zmiany trybu pracy dokonuje się dowolnym klawiszem. W dolnej części ekranu znajduje się linia komunikatów.**

Producent: Mastertronic
Autor: Icon Design

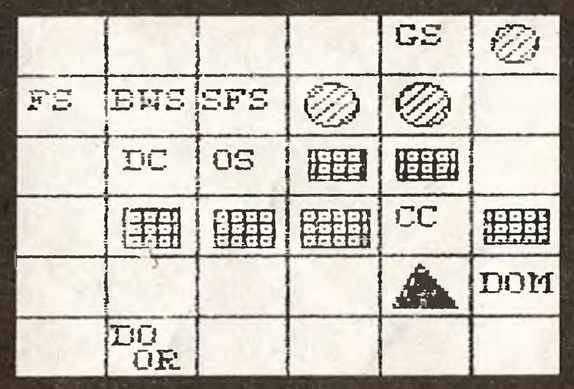
**Ocena (w skali 1–10):**

| | |
|---|---|
| **GRAFIKA:** | **8** |
| **DŹWIĘK:** | **5** |
| **SENS GRY:** | **9** |

**STEROWANIE**

**FIRE — rozpoczęcie gry**
**BREAK — pauza, FIRE — wznowienie**
**RESET — restart gry**
**Dowolny klawisz — zmiana trybu pracy robota**
**Joystick — sterowanie pracą robota (FIRE powoduje w trybie pracy przemieszczanie przedmiotów w zasobniku, a naciśnięcie i ruch do dołu powoduje pozostawienie pierwszego z prawej przedmiotu z zasobnika). FIRE w trybie walki to strzelanie. Wewnątrz budynku joystick porusza kursorem, FIRE uaktywnia opcję.**

*Maczek*




**— baterie słoneczne (zasilają generator)**
**— pole uprawne**
**— miejsce lądowania transportowca**

**DC — DROID CHARGER (punkt ładowania robota)**
**OS — ORGANIC SUPPLIES (magazyn produktów organicznych)**
**CC — CONTROL CENTRE (Centrum Kontroli)**
**GS — GENERAL SUPPLIES (magazyn ogólny)**
**FS — FENCING SUPPLIES (magazyn ogrodzeń)**
**SFS — STEEL FENCE STORE (magazyn siatki stalowej)**
**BWS — BARBED WIRE STORE (magazyn drutu kolczastego)**
**DOOR — główna brama kolonii**
**DOM — miejsce rozpoczęcia gry**


# TOMS

*Informujemy szanownych klientów, że wprowadziliśmy nową wersję systemu TOMS MULTI dla stacji LDW2000 i CA2001, zawierającą nowe oprogramowanie w ROMie stacji, oraz charakteryzującą się udoskonaloną współpracą ze Spartą X i systemem QMEG.*

*Użytkownicy naszych systemów mogą na korzystnych warunkach rozszerzyć możliwości swoich stacji, o ile posiadają one nienaruszone plomby gwarancyjne.*

*Ponadto polecamy nasze usprawnienia stacji i komputerów tym wszystkim, którzy jeszcze nie korzystali z naszych usług. Nie zawiodą się!!!*

*Tylko ćwierć megabajta pamięci RAM w komputerze oraz system TOMS MULTI DRIVE pozwolą w pełni wykorzystać możliwości drzemiące w 8-bitowym ATARI.*

*Pełne informacje i zamówienia:*

*mgr inż. Tadeusz Okoń, Warszawa–Ursynów, ul. Lachmana 4/44, tel. (0—2) 641-54-29,*
*mgr inż. Marek Stolarski, Warszawa–Natolin, ul. Kazury 13/26, tel. (0—22) 46-01-02 (godz. 8—16)*

A-3

Sklepy firmowe

# „ATARES"

polecają:

NAJTAŃSZE komputery COMMODORE, AMIGA, ATARI, instrumenty muzyczne (organy) firmy HOHNER, oprogramowanie, BLIZZARD TURBO, FLASH SYSTEM do LDW 2000, CA 2001 (format dyskietki 500 KB, transmisja 130 KB), cartridge systemowe i z grami, monitory, drukarki, joysticki, dyskietki i inne akcesoria. zapewniony serwis gwarancyjny i pogwarancyjny (CHORZÓW, TRUCHANA 35). Również realizacja zamówień hurtowych.

**Zapraszamy:**
**„ATARES"** Chorzów, Truchana 35, tel. 415-791 (hurt i detal)
**„ATARES"** Świętochłowice, A. Czerwonej 20 (detal)
**„MIRAGE"** Rybnik, Sobieskiego 7 tel. 212-42
**„BIT"** Racibórz, Szkolna 34
Racibórz, Browarna 2
**„ABC ELECTRONICS"** Gliwice, Wrocławska 7
**„HOBBIT"** Chorzów, Szczecińska 10
**„KRAM"** Bytom, PPR, tel. 81-65-29
**„HERMES"** Mysłowice, Wyspiańskiego 1

B 108

**DZIAŁ REKLAM MAGAZYNÓW MOJE ATARI TOP SECRET BAJTEK TEL. 21-12-05 codziennie 9.00—15.00**

# EMULA

**O zaletach Atari ST można pisać bardzo dużo. Wśród komputerów ST są maszyny stosunkowo proste, tak jak 520 ST, oraz potężne MEGA ST 4 i uwzględniające standardy ST, superszybkie Atari TT. Tworzą one rodzinę komputerów osobistych i domowych wysokiej klasy.**

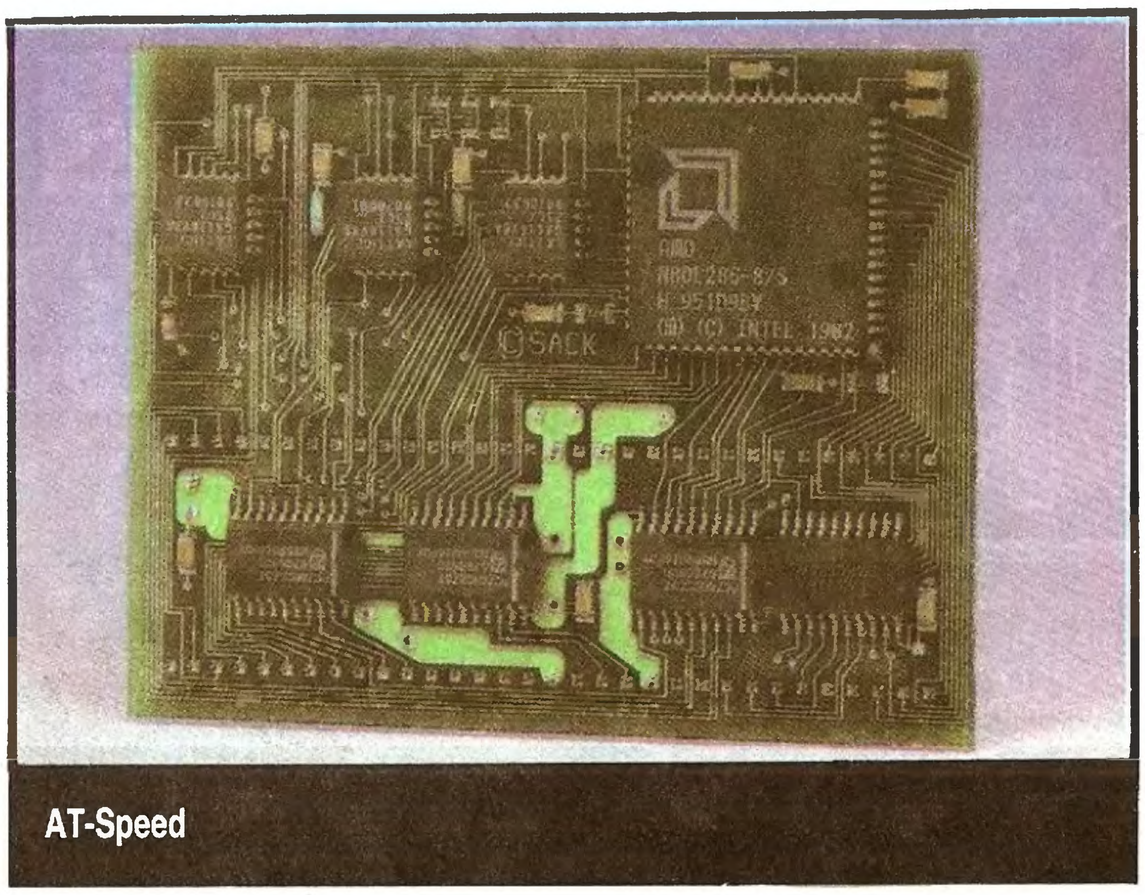
Na ST opracowano wiele znakomitych pakietów oprogramowania użytkowego. Niektóre z nich są znacznie lepsze niż ich odpowiedniki pracujące pod kontrolą MS-DOS. Wiele klasycznych programów dla IBM doczekało się już wersji na ST. Atari ST spełnia nawet najbardziej wyrafinowane wymagania, które można postawić przed komputerem tej klasy. Teoretycznie użytkownicy Atari nie powinni mieć powodów do zazdrości o inne komputery. Ale mimo to

## mit IBM

zbyt silnie oddziaływuje na wyobraźnię. Znak tej firmy ma niemal magiczne znaczenie. Komputery standardu zgodnego z IBM PC/XT/AT podbiły cały świat. PC są produkowane lub montowane prawie wszędzie. Nic dziwnego, że można je spotkać niemal w każdej instytucji oraz w wielu domach. Na potrzeby użytkowników oryginalnych IBM i jego klonów pracuje dziś potężny przemysł elektroniczny, tysiące firm software'owych, wydawnictwa. Znakomite oprogramowanie i bogata literatura fachowa stanowią łakomy kąsek dla użytkowników innych komputerów, w tym także Atari ST.

**W jednym z wydań audycji Radiokomputer słuchacze dopytywali się o możliwość uruchamiania programów dla IBM PC na komputerach Atari ST.**

Odpowiedź na to pytanie brzmi twierdząco, choć sprawa nie jest taka prosta. Najpierw musimy uzbroić się w odpowiednie narzędzia. Bez nich nie warto nawet próbować. ST i PC różnią się między sobą tak bardzo, że samo przełożenie dyskietki z jednego komputera do drugiego z góry skazane jest na niepowodzenie. Początkowo wydaje się, że wszystko sprzysięgło się przeciw amatorowi takiego eksperymentu. Inne procesory, inny system operacyjny, inna rozdzielczość ekranu.




AT-Speed

W ST wbudowana jest stacja dysków 3,5", a użytkownicy IBM przeważnie korzystają z dyskietek 5,25".

Mimo licznych trudności producenci sprzętu i oprogramowania znaleźli już sposób na spełnienie zachcianek użytkowników ST, a przy okazji na zwiększenie obrotów swoich firm.

Urządzenie, które potrafi zasymulować pracę innego komputera. nazywa się emulatorem.

Emulatory możemy podzielić na dwa podstawowe rodzaje: programowe i sprzętowe. Emulator programowy jest to najprościej mówiąc program, który pozwoli uruchomić na naszym komputerze programy napisane dla zupełnie innego komputera. Właśnie do tej pory zaliczamy program „PC Ditto", emulator IBM na ST. Nie będę się nim bliżej zajmował, ponieważ zamierzamy w najbliższych numerach opublikować na ten temat oddzielny artykuł.

Drugi rodzaj urządzeń, które umożliwiają uruchamianie programów IBM na ST, nazwałem emulatorami sprzętowymi, ponieważ składają się one z pewnego układu elektronicznego oraz oprogramowania. Za chwilę przekonamy się, że część sprzętowa takiego emulatora jest bardziej skomplikowana niż niejeden 8-bitowy komputer.

## PC Speed

Pierwszym emulatorem sprzętowym, który umożliwiał pracę ST pod kontrolą systemu operacyjnego MS-DOS, był PC-Speed. Jeszcze w lutym 1990 roku PC-Speed kosztował w RFN prawie 550 DM. W połowie roku na rynku pojawiły się emulatory PC/AT. Spowodowało to znaczną obniżkę cen PC-Speed, który w grudniu 1990 można było nabyć za około 350 DM. Emulator ten jest produkowany w dwóch wersjach: do ST i Mega ST oraz do STE.

W sierpniu 1990 oszacowano, że stosuje go już ponad 20 000 użytkowników różnych komputerów Atari ST różnych typów.

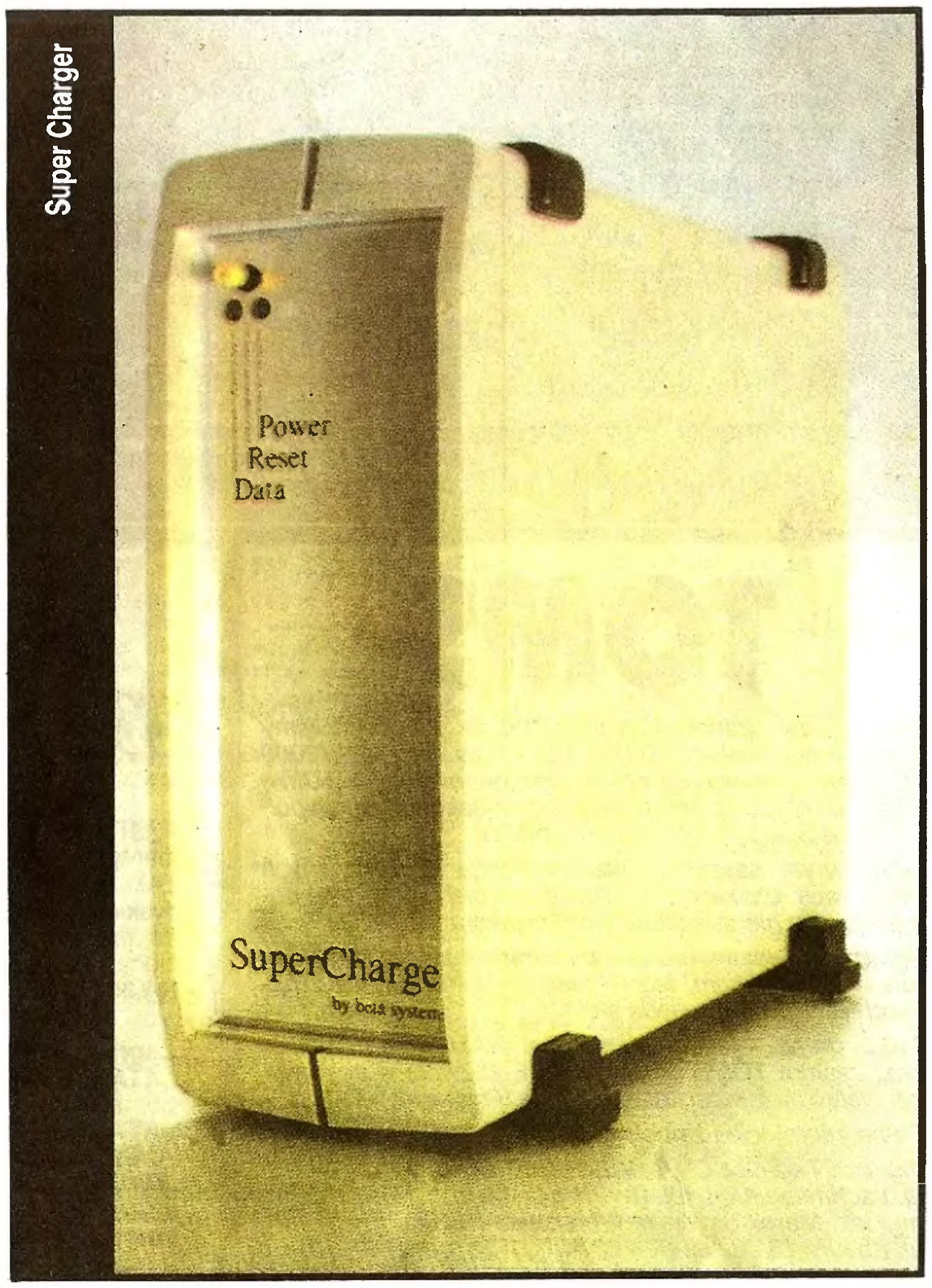
Następcami PC-Speed są znacznie szybsze emulatory AT: ATonce i AT-Speed. Szczegółowe omawianie obu znużyłoby naszych czytelników, ponieważ mają one bardzo zbliżone parametry. Podstawowe dane AT-Speed są zamieszczone na końcu artykułu.

## ATonce

Emulator ATonce zaprezentowano na ubiegłorocznych targach CeBIT'90 w Hanowerze. Już wkrótce po zakończeniu wystawy oferowało go wiele specjalistycznych sklepów. W skład zestawu wchodzi płytka, dyskietki z oprogramowaniem instalacyjnym i systemowym oraz podręcznik. ATonce bez problemów pracuje pod kontrolą MS-DOS 3.30 lub 4.01.

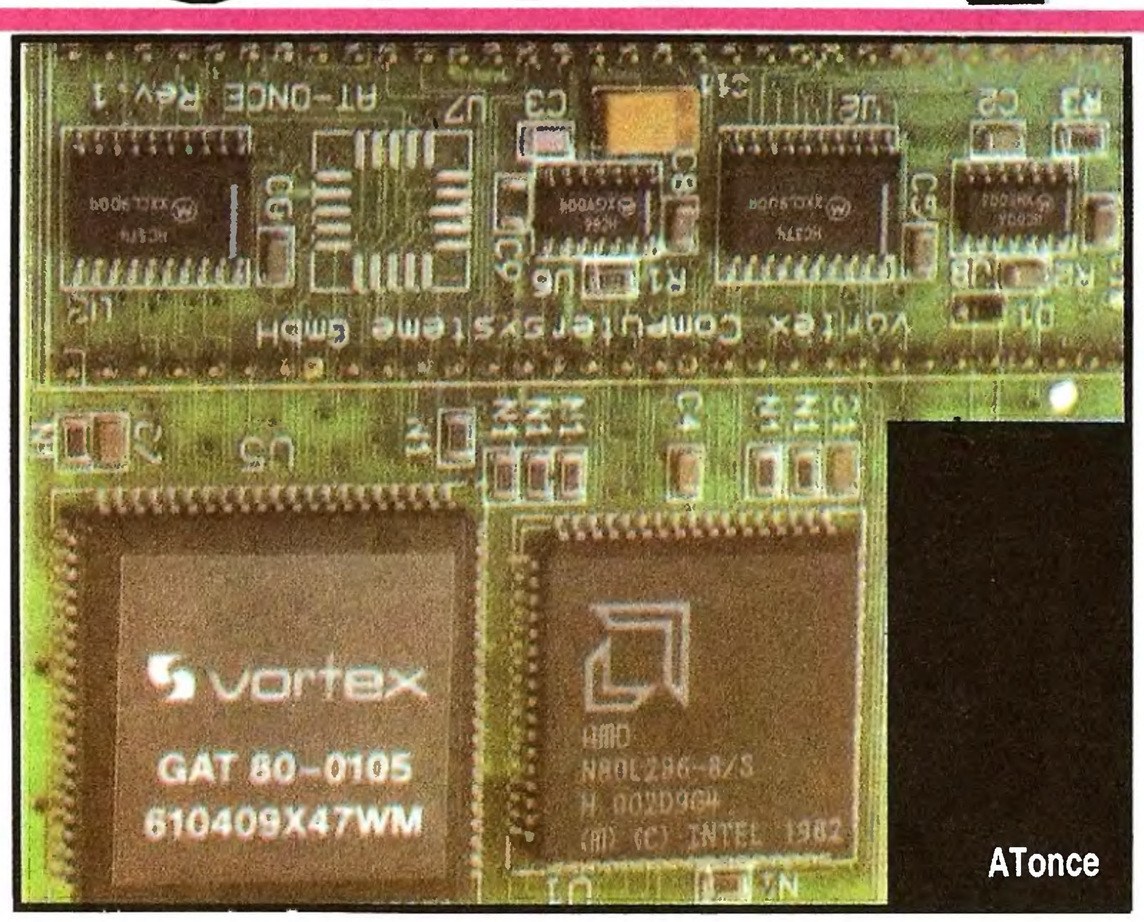
Urządzenie jest zbudowane na bazie mikroprocesora 80L286. Instrukcja dokładnie opisuje sposób postępowania podczas montażu układu w komputerze. W starszych modelach ST należy płytę emulatora wlutować, łącząc przy tym odpowiednio jej wyprowadzenia z mikroprocesorem 68000.

Właściciele Mega ST oraz 1040 STE mają o wiele łatwiejsze zadanie. Tu




Super Charger

# TORY PC




wystarczy specjalny adapter, który zakłada się bezpośrednio na podstawę mikroprocesora, a z montażem ATonce może poradzić sobie nawet laik, który nie ma zielonego pojęcia o lutowaniu obwodów elektrycznych.

Oprogramowanie zawarte na dyskietce systemowej dba o właściwe dopasowanie emulatora do konfiguracji posiadanego przez nas komputera. ATonce gwarantuje poprawną współpracę z dwiema stacjami dysków elastycznych oznaczanymi literami A i B oraz z twardym dyskiem. Pamięć twardego dysku można podzielić na 24 oddzielne obszary, tak zwane partycje, które są oznaczane literami C-Z. Każda partycja może mieć maksymalny rozmiar do 32 MB. Wskazane jest utworzenie dwóch partycji jednej dla programów MS-DOS, a drugiej dla TOS — systemu operacyjnego Atari. Unikniemy wtedy bałaganu, który utrudnia sprawne gospodarowanie posiadanymi zbiorami.

ATonce jest w stanie symulować działanie kilku różnych sterowników monitora. W zależności od potrzeb użytkownika może wybrać emulację karty CGA, Herkules, Toshiba T3100 lub Olivetti. W trybie Hercules uzyskuje się rozdzielczość 720×348 punktów, pod warunkiem, że w komputerze zainstalowano uprzednio układ OverScan. Urządzenie to pozwala podwyższyć rozdzielczość monitorów stosowanych w komputerach Atari, która w trybie monochromatycznym wynosi standardowo tylko 640 × 400 punktów. Właściciele kolorowych monitorów muszą się zadowolić tylko emulacją karty CGA.

System operacyjny można uruchomić zarówno z dyskietki, jak i z twardego dysku. Dane dla TOS i MS-DOS są zapisywane na twardym dysku i dyskietkach zawsze w tym samym formacie. Oznacza to, że informacje umieszczone w partycji MS-DOS mogą być również wykorzystywane przez programy uruchamiane pod systemem TOS.

Szybkość działania emulatora określa tzw. współczynnik Nortona, który w przypadku ATonce ma wartość 6,7. Wyznacza się go przy pomocy specjalistycznych testów. Programy testujące porównują badany komputer (lub emulator) z oryginalnym IBM XT, dla którego przyjęto współczynnik Nortona 1,0.

Użytkownik, który zdecydował się na zainstalowanie emulatora odnosi wrażenie, że pracuje z prawdziwym komputerem klasy PC. Teraz może korzystać nie tylko z biblioteki programów ST. Ma także do dyspozycji klasyczne pakiety oprogramowania pracującego pod kontrolą MS-DOS, takie jak Wordstar, Word, Lotus 1-2-3 czy dBase. Do zakupu ATonce zachęca niska cena, która wynosi niecałe 500 DM, a więc znacznie mniej niż kosztowałby nawet najprostszy komputer AT.

## SuperCharger

SuperCharger jest dotychczas jedynym emulatorem MS-DOS, który można przyłączyć do wyjść Atari ST. Podczas jego instalacji odpada konieczność ingerencji w wewnętrzną strukturę komputera, nie trzeba rozkręcać obudowy ani lutować połączeń. Emulator mieści się w eleganckiej, oddzielnej skrzyneczce, którą stawiamy obok ST Przed uruchomieniem wystarczy go połączyć kablem ze złączem DMA komputera (złącze do przyłączenia drukarki laserowej lub dodatkowego twardego dysku).

Wśród układów scalonych zamontowanych na płycie głównej SuperChargera na próżno szukać mikroprocesora Intel 820286, typowego dla innych emulatorów oraz komputerów AT. Sercem urządzenia jest mikroprocesor NEC V30.

SuperCharger jest nieco wolniejszy niż AT-Speed i ATonce. Ma on współczynnik Nortona 4,0. Za to inne zalety całkowicie rekompensują ten mankament. Najistotniejszą z nich jest własna pamięć RAM, która w zależności od konfiguracji ma pojemność 512 KB lub 1 MB. Dodatkowa pamięć znacznie zwiększa możliwości ST, gdyż jest dostępna nie tylko z poziomu MS-DOS, ale i TOS. Tak więc SuperCharger można przy okazji traktować jako rozszerzenie pamięci komputera. Pakiet procedur użytkowych o nazwie Toolbox pozwala zorganizować RAM-dysk w pamięci emulatora. Jest to bardzo wygodna metoda przechowywania zbiorów roboczych. Dane zapisane w RAM-dysku są odporne na resetowanie komputera pod warunkiem, że nie wyłączymy zasilania samego SuperChargera. W podręczniku i na dyskietce podane są również inne przykłady procedur użytkowych, przeznaczonych przede wszystkim do wykorzystania przez zaawansowanych programistów.

Procesor, zainstalowany w emulatorze, pracuje cały czas, niezależnie od mikroprocesora na płycie głównej ST. Każdy z nich może realizować osobne zadanie. Mamy więc do dyspozycji system pracujący wielozadaniowo. Teraz użytkownik nie musi się nudzić, kiedy wykonywane są jakieś żmudne i długotrwałe operacje. Wystarczy jedno zadanie, na przykład sortowanie bazy danych, zlecić programowi uruchamianemu pod MS-DOS. W naszym przykładzie będzie to dBASE. Oczekiwanie na wynik pracy wykonywanej przez emulator skrócimy sobie rundką ulubionej gry, którą rozegramy z ST. Powrót do trybu ST nie wymaga skomplikowanych czynności. Wystarczy nacisnąć jednocześnie klawisze Alt-Control-Backspace.

Emulatory zdobyły sobie w RFN bardzo wielu zwolenników. Sprzedaje je większość dystrybutorów komputerów i urządzeń peryferyjnych Atari. Ostatnio oferowane są nawet Mega ST 1 i Mega ST 2 w wersji z zainstalowanym PC-Speed lub AT-Speed. Firma Vortex uatrakcyjnia swoją ofertę łącząc ATonce w zestawy z twardym dyskiem 40 MB.

Nie zpominajmy jednak, że żaden emulator nie zastępuje prawdziwego komputera osobistego klasy IBM PC. Użytkownik zawsze musi liczyć się z wystąpieniem pewnych nieoczekiwanych problemów. W skrajnych przypadkach niektóre programy mogą po prostu nie chodzić.

Istnieją również emulatory innych komputerów na ST. Niedawno natknąłem się na wzmiankę o dwóch emulatorach Apple-Macintosch. Niestety, poza wymienieniem nazw, ALADIN i SPECTRE 128, nie potrafię powiedzieć nic więcej na ten temat. Może pomogą nam czytelnicy „Bajtka" i „Mojego Atari". Czekamy na listy związane z tym tematem.

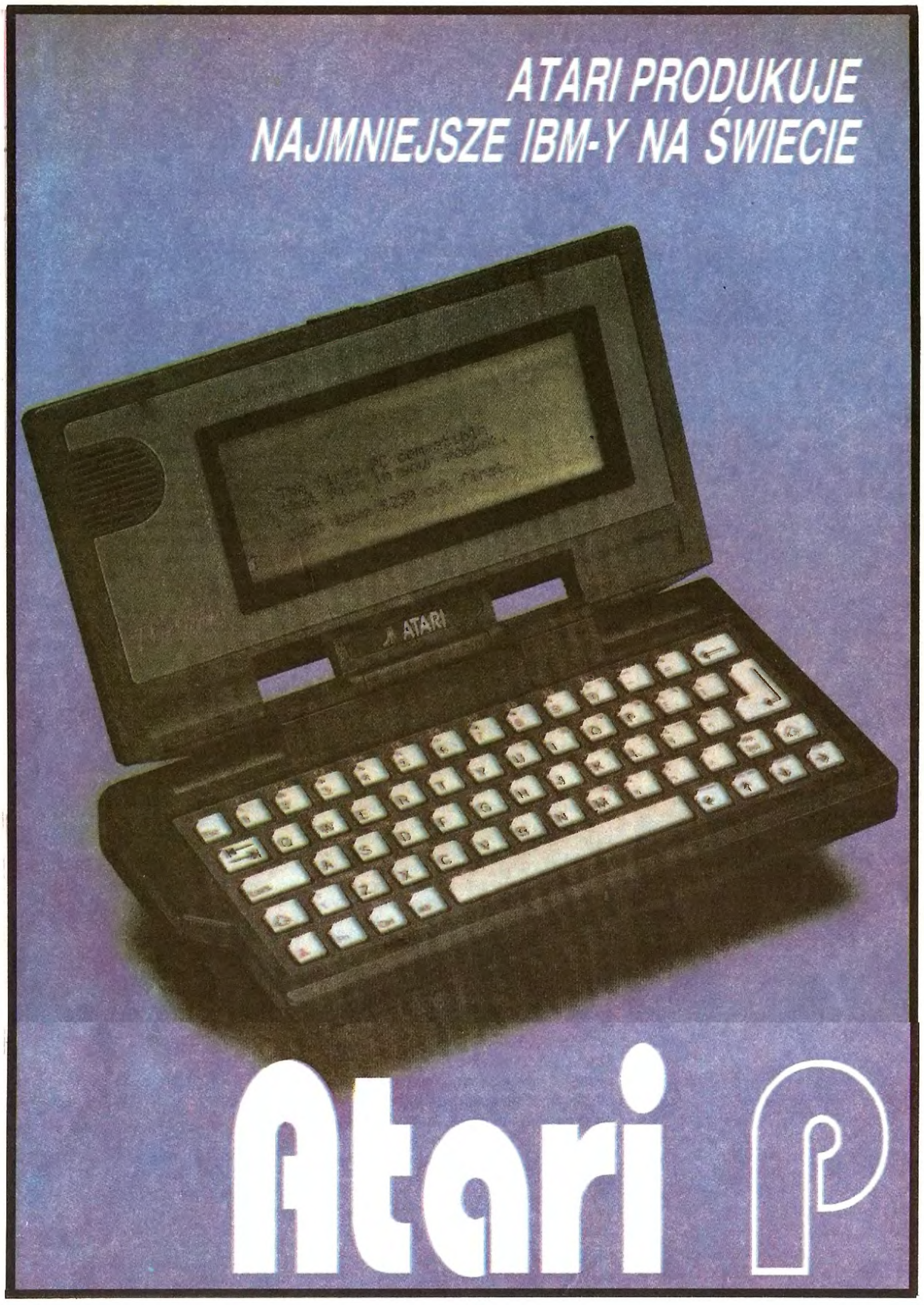
*Janusz Jarmoch*

**Dane techniczne emulatora AT- Speed**

* Mikroprocesor Intel 80286
* Współczynnik Nortona 6.7
* System operacyjny MS-DOS
* Cała pamięć RAM w Atari pod kontrolą mikroprocesora 80286
* Wykorzystanie pamięci podstawowej DOS do 702 KB
* Wykorzystanie pamięci rozszerzonej/dodatkowej max 3 MB
* Uruchamianie systemu z wewnętrznej lub zewnętrznej stacji dysków
* Współpraca z wewnętrzną stacją dysków 360/720 KB
* Współpraca z zewnętrzną stacją dysków 3.5"/5.25", 360/720 KB
* Współpraca ze stacją dysków 1,4 MB
* Współpraca z twardym dyskiem
* Możliwość wydzielenia na twardym dysku 24 partycji po 32 MB
* Wykorzystanie myszy Atari jako zgodnej z Mictosoft Maus
* Emulacja kart graficznych CGA, HERCULES, OLIVETTI, ATT, TANDY
* Od wersji 2.21 emulacja karty graficznej EGA i VGA
* Komunikacja poprzez interfejs równoległy i szeregowy

**Ceny detaliczne emulatorów w RFN — grudzień 1990**

| | |
|---|---|
| * PC Speed | 349– 398 DM |
| * AT Speed | 488– 528 DM |
| * AT Once | 448– 488 DM |
| * Wbudowanie emulatora | 49– 65 DM |
| * SuperCharger 1 MB | 698– 798 DM |
| * Mega ST2 + AT Speed | 2279–2448 DM |
| * Mega ST2 + PC Speed | 2159 DM |




**Portfolio jest jednym z najmniejszych komputerów osobistych, które pracują pod kontrolą systemu operacyjnego MS-DOS. Określenie „osobisty" można w tym przypadku potraktować zupełnie dosłownie, ponieważ całe urządzenie z powodzeniem mieści się w kieszeni kurtki, nie wspominając już o torbie lub neseserze. Przeznaczenie komputera znakomicie oddaje jego nazwa. Słowo Portfolio oznacza po angielsku teczkę lub aktówkę, czyli przedmiot, z którym wybieramy się w podróż. Kto nie wierzy, niech sprawdzi w słowniku!**

## 640 KB w kieszeni

Atari Portfolio ma wymiary zaledwie 200 × 100 × 25 mm, czyli nieznacznie więcej niż kaseta video systemu VHS, i waży tylko 495 gramów. Jego mózgiem jest szesnastobitowy mikroprocesor INTEL 80C88, taktowany z częstotliwością 4.92 MHz. Pamięć wewnętrzna, 128 KB RAM i 125 KB ROM, nie zaimponuje użytkownikom dużych komputerów AT i 386, którzy już zdążyli oswoić się z cyframi sięgającymi kilku megabajtów. Wypowiadając się na ten temat nie zapominajmy jednak, że Portfolio nie zajmuje powierzchni całego biurka, tak jak jego więksi bracia. Jego pamięć jest i tak większa niż Spectrum, Atari 800 XL i Commodore 64 razem wzięte. Bardziej wymagający użytkownicy mogą w razie potrzeby dokupić moduł rozszerzający RAM do 640 KB. To odpowiada już standardom przyjętym w normalnych, biurowych komputerach PC.

## Karty RAM i RAM-dysk

Wprawdzie Portfolio nie ma wbudowanego napędu dysków elastycznych, ale do zapisu i przechowywania danych przewidziano inne urządzenia. Podstawowym nośnikiem informacji są karty RAM, które funkcjonalnie zastępują zwykłe dyskietki. Ich wielkość odpowiada rozmiarom karty kredytowej. Wewnątrz znajduje się bateria, która podtrzymuje dane przez około 12 miesięcy. Pojemność kart RAM wynosi 32, 64 lub 128 KB. Dane i programy są zapisywane w postaci zbiorów, tak jak to robi typowa stacja dysków. Jako wyposażenie dodatkowe można zamówić urządzenie odczytujące i zapisujące karty RAM, czyli po prostu coś w rodzaju stacji dysków. Stację podłącza się bezpośrednio do komputera PC. Umożliwia ona wymianę danych z PC bez udziału Portfolio. Nowością, która pojawiła się w tym roku, są karty OTP-ROM (One Time Programmable ROM Board). Dane można zapisać na nich tylko raz. Przewiduje się, że OTP-ROM przyjmie się przede wszystkim jako nośnik pakietów oprogramowania użytkowego przeznaczonego dla Portfolio. Już teraz na kartach OTP oferowanych jest wiele interesujących pozycji software.

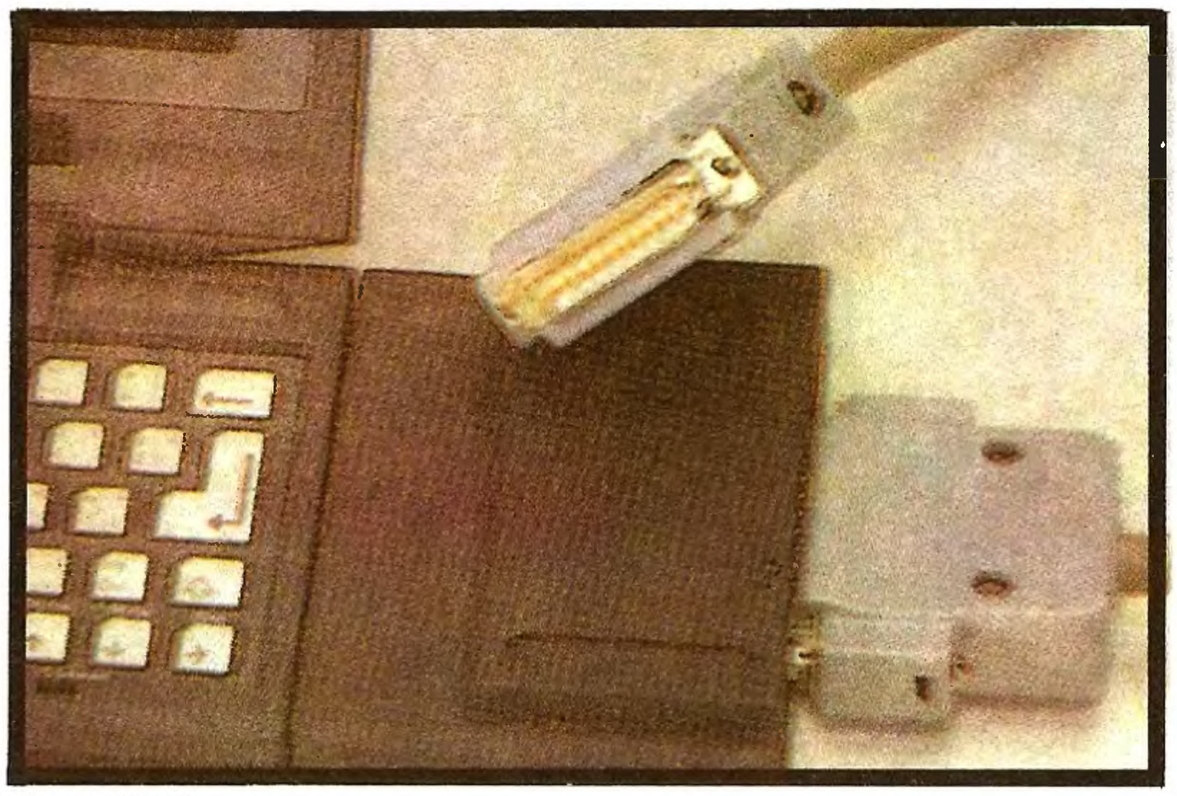
Podręczne informacje, nad którymi aktualnie pracuje użytkownik, najlepiej zapisywać na RAM-dysku. System oznaczą go literą C. Zapisane na nim dane nie giną nawet po wyłączeniu zasilania. Jest to zasługa nowoczesnych układów pamięci RAM.

## A gdzie wyłącznik?

Kieszonkowy komputer oczywiście nie może być zasilany z sieci elektrycznej. Podstawowym źródłem energii dla Portfolio są trzy baterie 1,5 V. Producent zaleca stosowanie baterii alkalicznych, które są znacznie trwalsze i nie wyciekają tak jak baterie cynkowo-węglowe. Powinny one wystarczyć na około 6 tygodni użytkowania komputera.

Co się dzieje podczas wymiany baterii? Niektórzy mogą się obawiać o utratę informacji pracowicie zmagazynowanych w RAM-dysku. Tymczasem okazuje się, że jest to zupełnie bezpieczna czynność. Użytkownik ma na włożenie nowych baterii aż 10 minut. W komputerze zastosowano bowiem specjalny rodzaj układów pamięci RAM, tzw. SRAM, które zużywają bardzo mało energii i zachowują dane przez pewien czas po odłączeniu zasilania.

Oczywiście istnieje również możliwość podłączenia komputera do sieci elektrycznej. Niestety zasilacz należy dokupić osobno, ponieważ nie należy do zestawu wyposażeń standardowych. Włączenie zasilania następuję po naciśnięciu dowolnego klawisza. Rolę wyłącznika pełni kombinacja klawiszy FN (w lewym dolnym rogu klawiatury) i „O", który dodatkowo oznaczono symbolem OFF. Komputer jest wyłączany także poleceniem OFF, podanym z poziomu DOSu. Kto o tym nie wie, będzie niepotrzebnie tracić czas na szukanie tradycyjnego wyłącznika sieciowego.



*Złącze RS232*

## Ekran i klawiatura

W Portfolio zastosowano rozwiązanie monitora typowe dla komputerów przenośnych. Monochromatyczny ekran ciekłokrystaliczny jest wbudowany w uchylną pokrywę, która po złożeniu całkowicie zasłania klawiaturę.
Jego rozdzielczość wynosi 240 × 64 pixeli, co daje w trybie tekstowym 8 wierszy po 40 znaków. Czytelność ekranu jest dobra tylko przy dostatecznym oświetleniu. Nie pogarsza jej jaskrawe światło słoneczne. W zbyt ciemnym pomieszczeniu tekst przestaje być widoczny.

Klawiatura typu QWERTY zawiera 63 klawisze, a wśród nich 4 klawisze sterujące ruchem kursora. Klawisze numeryczne 0–9 spełniają jednocześnie rolę klawiszy funkcyjnych.

Niektórzy zarzucają Portfolio, że ekran jest zbyt mały i mieści tylko 40 znaków w wierszu. Za słabą stronę uznawana jest również zbyt uboga klawiatura.
Ale nie zapominajmy, że Portfolio ma mieścić się w kieszeni. Te niedogodności są ceną, jaką użytkownik musi zapłacić za zmiejszenie komputera do wymiarów niewiele większych od kieszonkowego kalkulatora. Zachowania standardów obowiązujących w typowym PC możemy domagać się od urządzenia, które można postawić na normalnym biurku.

## Wyposażenia specjalne

Do mankamentów Portfolio zalicza się również brak wbudowanych na stałe interfejsów. Na szczęście można je dokupić jako dodatkowe wyposażenie. Interfejs równoległy, przy pomocy którego przyłącza się drukarkę, kosztuje niecałe 100 DM. Za jego pośrednictwem odbywa się również wymiana informacji z normalnym PC. Użytkownik otrzymuje wraz z interfejsem dwie dyskietki w formacie 3,5'' i 5,25'', na których znajduje się odpowiedni program komunikacyjny. Dodatkowym wyposażeniem jest także interfejs szeregowy. Na szczęście jest on potrzebny tylko tym, którzy zamierzają korzystać z medomu.

Oddzielenie interfejsów wynika z samej idei zminiaturyzowanego komputera. Wbudowanie ich na stałe nie stanowiłoby najmniejszego problemu technicznego, a przecież nie zawsze są potrzebne. Użytkownik musiałby dźwigać dodatkowy ciężar, no i oczywiście znacznie więcej zapłacić za sam komputer. Oddzielenie pewnych elementów pozwoliło skonstruować maleńkie, przenośne urządzenie, którego zalety są znacznie bardziej widoczne, niż słabe strony. Interfejs równoległy może spokojnie czekać w szufladzie biurka, ponieważ nikt i tak nie będzie dźwigać ze sobą ani drukarki, ani dużego PC. Wyciągniemy go dopiero wtedy, gdy będziemy chcieli coś wydrukować.

Wśród dodatkowych wyposażeń znajduje się także modem akustyczny.
Może on przesyłać informacje z szybkością 1.200 bodów.

## Oprogramowanie

Portfolio pracuje pod kontrolą systemu operacyjnego DIP Operating System 2.11. System ten, to po prostu MS-DOS wersja 2.11, który adaptowano do pracy w specyficznym środowisku komputera przenośnego.

Użytkownik, który decyduje się na zakup Portfolio, nie musi martwić się o poszukiwanie podstawowego oprogramowania. W pamięci ROM, oprócz MS-DOS, znajduje się kilka programów użytkowych. Dobrano je tak, by komputer jak najlepiej spełniał oczekiwania biznesmena, dziennikarza czy urzędnika wybierającego się w podróż. Informacje dotyczące sposobu posługiwania się programami uzyskujemy nie tylko z instrukcji obsługi. Najwłaściwszy sposób postępowania podpowie nam w razie potrzeby sam Portfolio. Służy do tego, podobnie jak w wielu innych komputerach osobistych, klawisz F1. Podczas instalacji systemu możemy wybrać język, w którym będą wyświetlane teksty informacyjne (Help). Do dyspozycji mamy trzy języki europejskie: angielski, niemiecki i francuski.

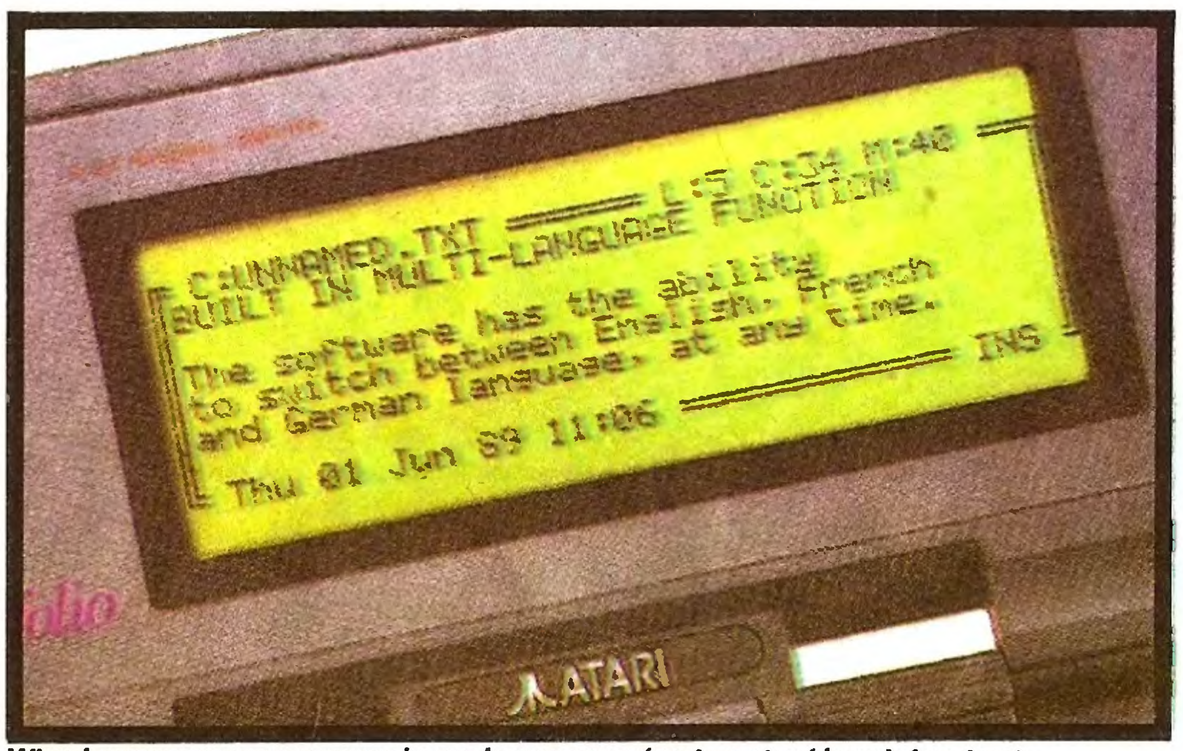
Do redagowania tekstów służy edytor. Maksymalna długość wiersza wynosi 250 znaków. Teksty są zapisywane w formacie ASCII, a więc można je bez kłopotu przetwarzać na innych komputerach. Edytor realizuje podstawowe funkcje niezbędne podczas pracy nad dokumentem: poszukiwanie i zastępowanie zadanego ciągu znaków innym tekstem, przenoszenie słów, operacje kopiowania i kasowania bloków tekstu. Jego możliwości są bardzo skromne w porównaniu z najnowocześniejszymi, profesjonalnymi edytorami. Najważniejsze jednak, że można na nim bardzo szybko sporządzić podręczne notatki, które potem spokojnie opracujemy za pomocą programów zainstalowanych w dużym, biurowym komputerze.

Drugim ważnym narzędziem jest spreadsheet, czyli program do opracowywania danych w formie zestawień i tabelek. Arkusz roboczy Portfolio jest zapisywany w formacie zrozumiałym dla programu Lotus 1-2-3. Maksymalna wielkość opracowywanej tabeli wynosi 255 wierszy i 127 kolumn. Nie zawsze potrzebne jest tak mądre narzędzie do prowadzenia obliczeń, jakim jest spreadsheet. Czasami wystarcza kieszonkowy kalkulator. Kolejny program pozwoli nam wykorzystywać komputer jako zwykłą, inteligentną maszynę liczącą.

Obok edytora tekstu i arkusza kalkulacyjnego zainstalowano w Portfolio prostą bazę danych. Jest ona zorganizowana w formie książki adresowej. O terminach umówionych spotkań będzie nam przypominać program typu Time Manager, który zastępuje kieszonkowy kalendarzyk. Jego zalety szczególnie docenią zapominalscy.

Oprogramowanie Portfolio nie zastąpi profesjonalnych pakietów produkowanych przez czołowe firmy. Realizuje ono tylko podstawowe funkcje dostępne w typowych programach stosowanych w biurach. Mimo wszystko kieszonkowy komputer, który dysponuje takimi możliwościami, robi imponujące wrażenie.
Wszystkie opracowane przez użytkownika dane można przesłać do innego komputera pracującego pod kontrolą systemu MS-DOS.

# Folio




***Wbudowane oprogramowanie może pracować w trzech różnych językach***

## DANE TECHNICZNE ATARI PORTFOLIO

**Procesor:**

* Mikroprocesor INTEL 80C88
* Częstotliwość zegara 4,92 MHz

**Pamięć wewnętrzna:**

* Pamięć RAM 128 KB
* Pamięć ROM 256 KB
* Możliwość rozszerzenia pamięci RAM do 640 KB
* RAM-dysk

**Klawiatura:**

* Klawiatura typu QWERTY
* 63 klawisze
* 4 klawisze sterujące ruchem kursora
* Klawisze funkcyjne razem z klawiszami numerycznymi

**Monitor:**

* Zintegrowany ekran ciekłokrystaliczny
* Rozdzielczość 240 × 64 punkty
* 8 wierszy tekstu po 40 znaków

**Zasilanie:**

* 3 baterie 1,5 V
* Gniazdo zasilacza elektrycznego

**Wymiary i waga:**

* Długość 200 mm
* Szerokość 100 mm
* Wysokość 25 mm
* Waga 495 g

**Wyposażenia specjalne:**

* Interfejs szeregowy
* Interfejs równoległy
* Moduł rozszerzający pamięć do 640 KB
* Karty pamięci RAM 32 KB, 64 KB, 128 KB
* Karty OTP-ROM
* Zewnętrzna stacja dysków do kart RAM
* Modem akustyczny
* Zasilacz

**Oprogramowanie:**

* System operacyjny MS-DOS 2.11
* Arkusz kalkulacyjny kompatybilny z Lotus 1-2-3
* Edytor tekstu
* Kalkulator
* Książka adresowa
* Kalendarz
* Program komunikacyjny

Niewielkie rozmiary, własne źródło zasilania i łatwość transportu zainteresowały producentów różnych urządzeń elektronicznych.
Jedna z niemieckich firm wyprodukowała ostatnio miernik napięcia i natężenia prądu, którego jednostką centralną jest właśnie Portfolio. Myślę, że takich przykładów będziemy mogli wkrótce podać więcej.

*Janusz Jarmoch*

Najtańsze w Polsce!

**SV 119**
**Junior**
2 Fire
6 Blaszanych styków
Prosty mechanizm

**SV 120**
**Junior–Stick**
2 Fire
6 Blaszanych styków
Uchwyt pistoletowy

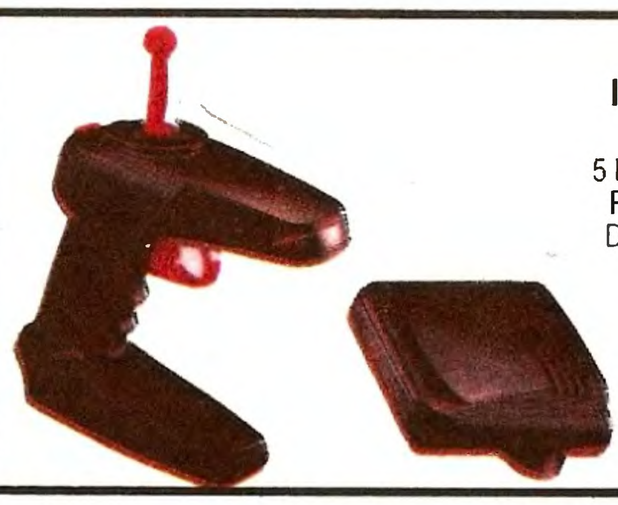
**SV 122**
**Quickjoy II**
2 Fire
6 Blaszanych styków
AutoFire
Drążek lotniczy

**SV 124**
**Turbo**
6 Mikrostyków
AutoFire
Drążek lotniczy

**SV 123**
**Supercharger**
2 Fire
6 Mikrostyków
Ergonomiczna budowa
Precyzyjny mechanizm

**SV 126**
**Jet Fighter**
2 Fire
6 Mikrostyków
AutoFire
ACS-Regulator
szybkości AUTO
Obsługa pod kciuk"
Drążek lotniczy

**SV 125**
**Superboard**
6 Fire
10 Mikrostyków
AutoFire
Cyfrowy wyświetlacz
czasu
Sygnał dźwiękowy
Przełącznik dla
leworęcznych
Drążek lotniczy



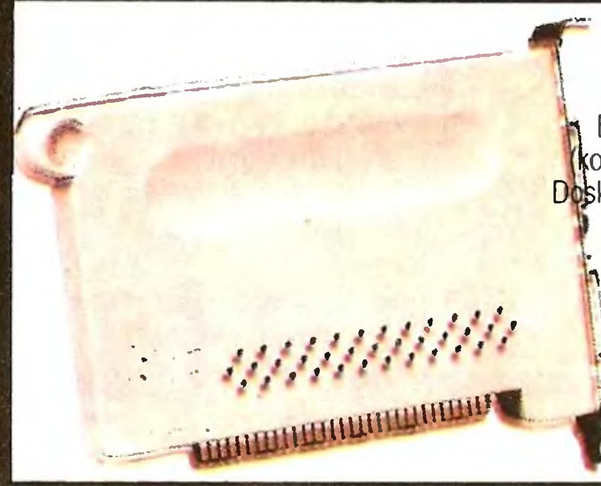
**SV 130**
**IR Infrared**
1 Fire
5 Mikrostyków
Podczerwień
Daleki zasięg
Odbiornik

**SV 128**
**Megaboard**
4 Fire
10 Mikrostyków
AutoFire
6 cyfrowy stoper
ATM — Anti Tilt Mechanism
Fire Pad

**SV 140**
**Enterprice**
2 Fire
6 Mikrostyków
AutoFire
ACS — Regulator
szybkości AUTO
Drążek lotniczy
„kierownica"
Kabel 4 m

**SV 201**
**Quickjoy M 5**
Do IBM XT/AT
(kompatybilnych)
Współpracuje z Game-Card
lub I/O Card
2 Fire
2 AutoFire
6 Mikrostyków
Wybór AUTO
PSC — Regulator XY
Sygnalizacja Świetlna
Fire
ASC — Regulator szybkości
AUTO

**SV 202**
**M 6 analog**
Analogowy
DO IBM XT/AT
(kompatybilnych)
Współpracuje z Game-Card
lub I/O Card
2 Fire



**SV 210**
**Game Card**
Do IBM XT/AT
(kompatybilnych)
Doskonale pracuje
z M 5 i M 6

**SV 127**
**Top Star**
2 Fire
6 Mikrostyków
AutoFire
Przeźroczysta obudowa
SAS — Shock Absorbing
System
Platynowane części

**SV 500**
**Van 3**
Pudełka na dyskietki
80 sztuk 3½"
Zamknięcie na klucz

**SV 510**
**Van 5**
Pudełko na dyskietki
80 sztuk 5¼"
Zamknięcie na klucz

| | | | |
|---|---|---|---|
| SV 119 — 64 900 | SV 123 — 119 900 | SV 128 — 319 000 | SV 210 — 219 000 |
| SV 120 — 74 900 | SV 125 — 239 900 | SV 130 — 339 000 | SV 500 — 119 000 |
| SV 122 — 89 900 | SV 126 — 169 900 | SV 201 — 249 000 | SV 510 — 119 000 |
| SV 124 — 109 900 | SV 127 — 249 900 | SV 202 — 229 000 | |

**interwencje**
**tel. 23-92-21**

SV 201 + SV 210 — 439 000
SV 202 + SV 210 — 399 000

Należy przesłać pieniądze przekazem zwykłym (czerwony blankiet) na adres:

TAL Quickjoy

***Uwaga!** W miejscu na korespondencję prosimy podać po raz drugi: adres domowy, kod i symbol zamówionej pozycji. W przypadku braku adresu zwrotnego nie realizujemy zamówienia.*

„TAL" — QWERTY
ul. Mikowa 45
02-411 Warszawa
fax: 659-12-35

**Czas realizacji 14 dni**