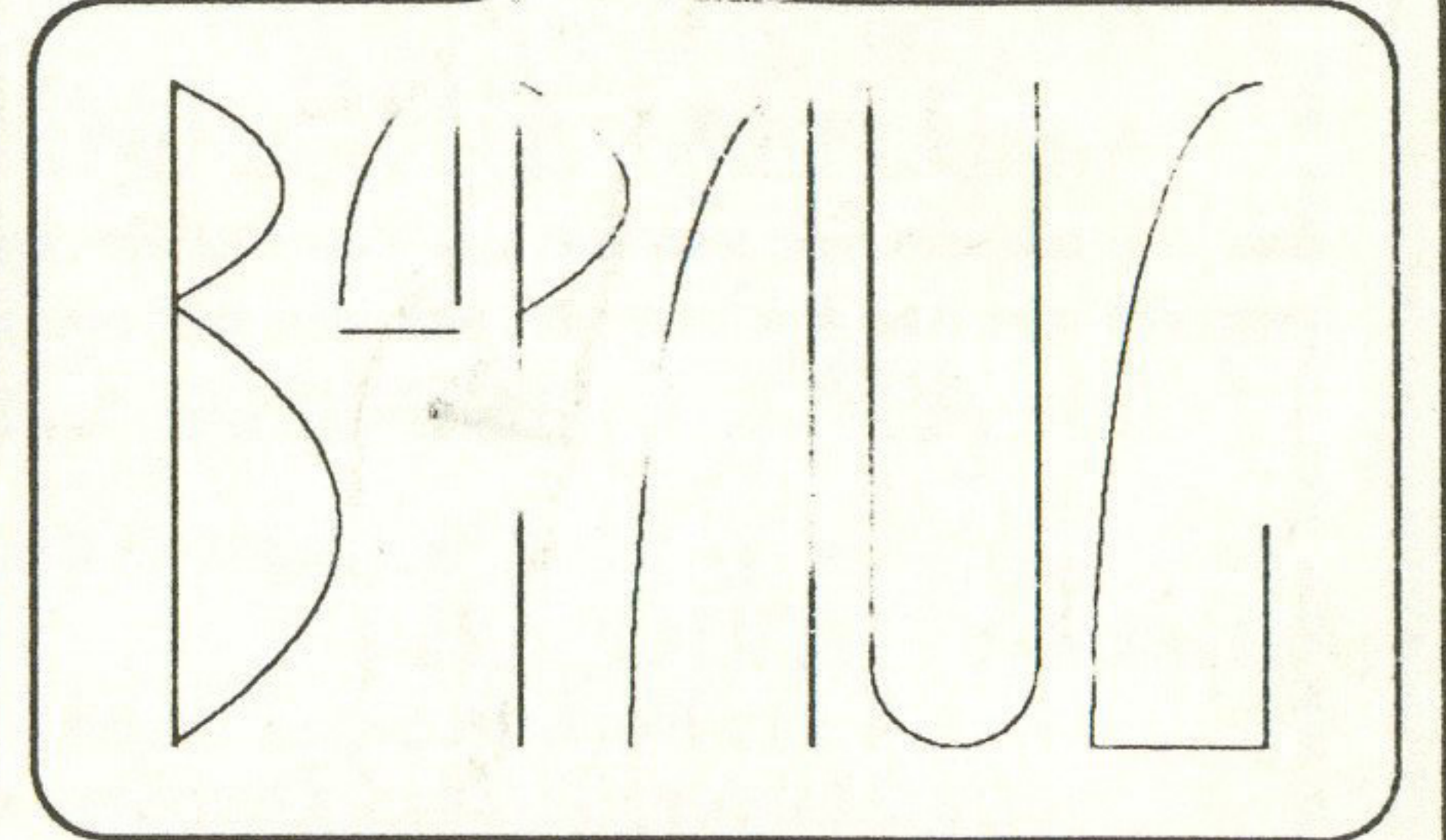
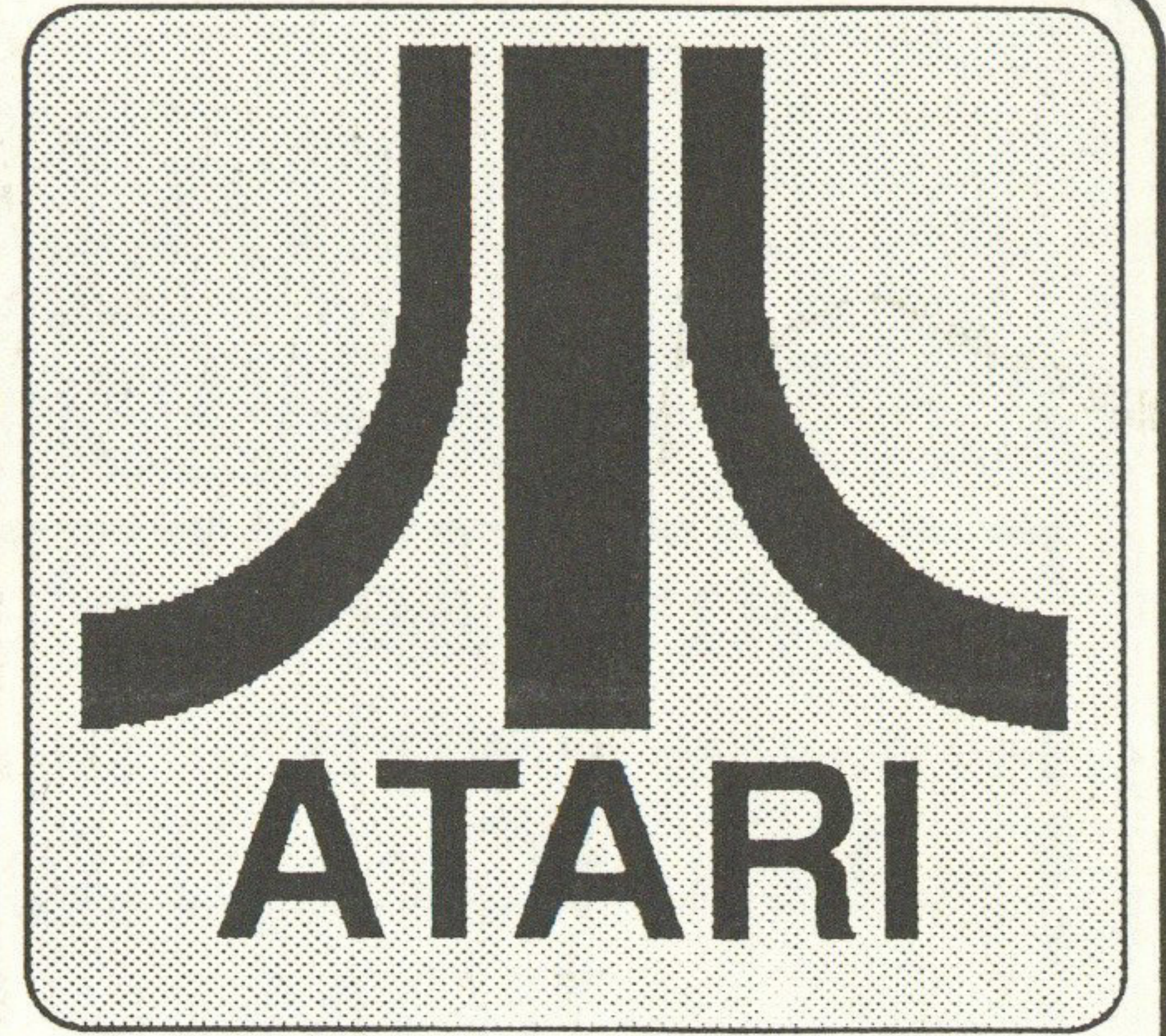


The Alternative Atari Newsletter



£1.25

May/June 1991

Issue 12

Deskjet 500 Reviewed

8:16 Competition
8bit and ST prizes to win.

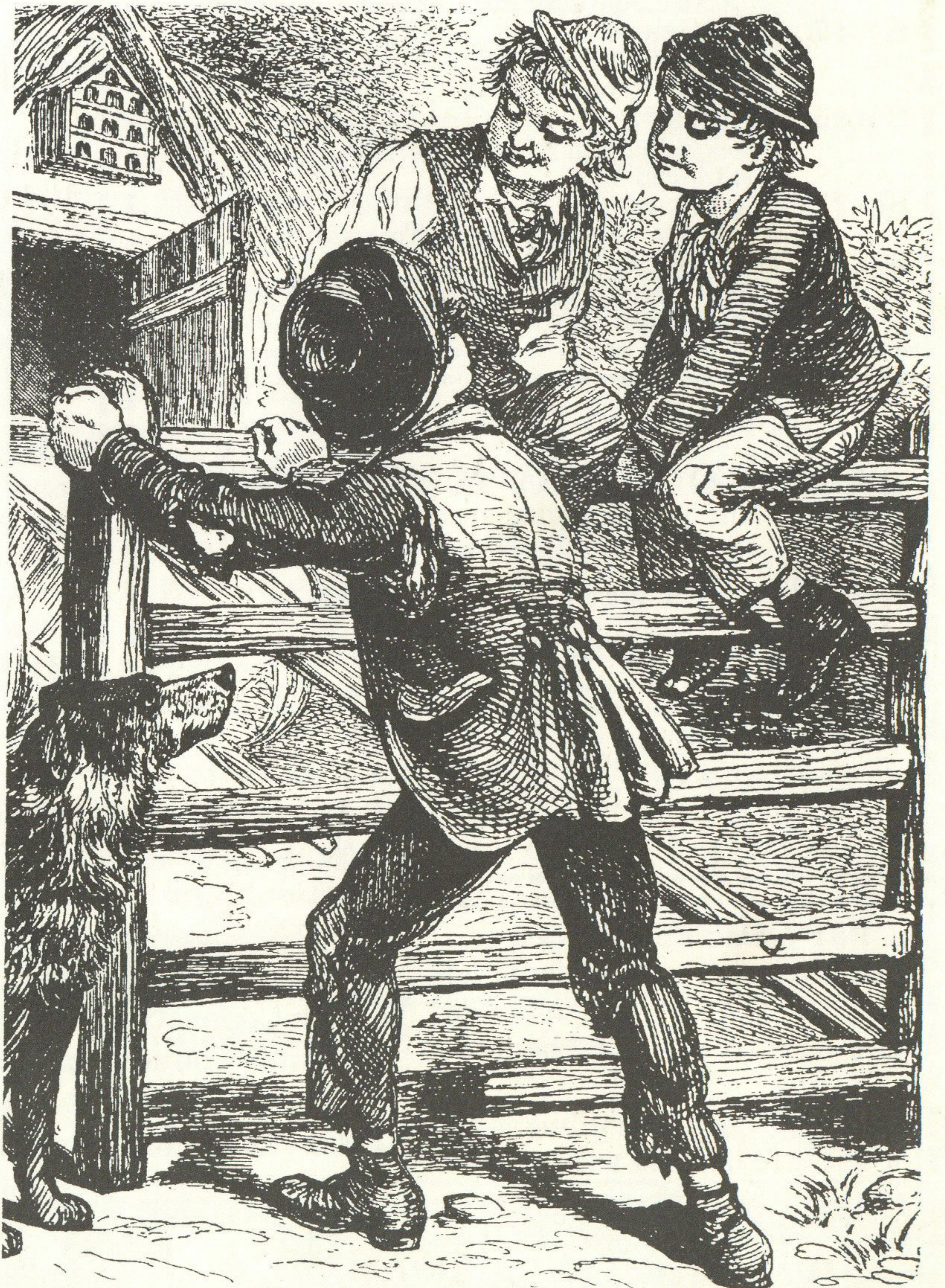
PL65
A look at this rare 8bit
language.

Disk Drives & the DCB
How to talk to your disk drive at
the machine level!

AtariWriter Plus
Turbo 816
Continuation of our tutorials.

Ever wanted to switch
between MAC/65 and
BASIC?
Well, we tell you how.

Plus the regular columns:
Notice Board, User Group
File, Dear 8:16



GRALIN INTERNATIONAL

11 Shillito Road, Parkstone, Poole, Dorset

Atari Light Pen
with
Atari Graphics
ROM
Only £25.00

Atari XE Games Console
(Boxed with PSU)
Built in Missile Command
Joystick and 2 extra ROMs
Limited Quantity- £40 + P&P

APX Cassette Collections

£4.00 per set

- Set #1: **Utilities #1**
Text Formatter (Forms); Sound Editor; Decision Maker; T: Text Display; Chameleon CRT Terminal Emulator
- Set #2: **Games #1**
Mugwump; Babel; Block 'em; Phobos
- Set #3: **Educational Software**
Typo Attack; Counter; Memory Match; Morse Code Tutor
- Set #4: **Games #2**
Quarxon; Solitaire; Pro Bowling; Codecracker
- Set #5: **Utilities #2**
Extended WSN; Banner Generator; Instedit; Ultimate Renumber Utility; Dsembler
- Set #6: **Games #3**
Blackjack Casino; Dice Poker; Mankala; Lookahead
- Set #7: **Games #4**
Avalanche; Attank!; Space Trek; Salmon Run
- Set #8: **Mixed Bag**
Player Piano; Terry; Space Chase; Magic Melody Box

ANTIC Software

We have aquired a number of ANTIC software titles, along with back issue magazines and disks. List available with catalog, or send wants list.

- Software£ 6.99 ea
Magazines£ 0.99 ea
Magazine with disk£ 2.49 ea
Magazine Issue Disk£ 1.59 ea
New software titles available.

- 850 Interface£49.99
Multi-Viewterm+Interface .£29.99
Super Data Base 1-2-3£59.95
Fooblitzky£14.95
Video Title Shop£19.95

Come and meet us at the
34th Longleat Radio Rally
30th June
2 miles west of Warminster
Other attraction: Craft Fair

Cassette Software

£2.49 per set

- Set 1: Star Raiders, Chess, Asteroids, Missile Command, Centipede, Airstrike 2, Realsport Tennis, Chop Suey, Realsport Football, Basketball
- Set 2: Typo Attack, Missile Command, Star Raiders, Centipede, Realsport Tennis, An Invitation To Programming

Parrot II
Sound Sampler
Only £49.95

P&P / Handling Charges

- Software: £1.50 for orders
under £50
Hardware: £2.50 per order

There is no P&P or handling charge for software ordered with printers, computers or hardware.
Overseas orders extra. Please ask for quote.
Cheque/Postal Orders only.

ICD Products

- Action! language cartridge with Toolkit disk£41.99
Action! Toolkit disk£15.75
Action! Runtime disk£15.75
MAC/65 assembler cartridge with Toolkit disk£41.99
MAC/65 Toolkit disk£15.75
P:R: Connection£52.49
Printer Connection£31.49
R-Time 8 cartridge£31.49
Rambo XL memory upgrade board (no RAM)£20.99
SpartaDOS Toolkit disk£15.75
SpartaDOS X cartridge£41.99
US Doubler 1050 enhancement£15.75
US Doubler with SpartaDOS Construction set£31.49

The Flight Crew!

Colin Hunt Starfleet Controller
Graham Broomfield Ground Controller
Thomas Holzer VCS Controller

All contributors are credited with their articles entry within the contents list. Which just happens to be on this page!

People who regularly help/contribute include Paul Brookes, Ian Brooker, Thomas Holzer, Graham Broomfield, Simon Trew and David Watson. Sorry if I missed anyone out.

8:16 is produced by BaPAUG Publishing.

All original articles published within 8:16 are the copyright of the author, but can be used by other publications provided the author and 8:16 are acknowledged as the original source and a copy of the publication is sent to 8:16. Articles re-printed from other publications should be credited to the original source.

8:16 and the BaPAUG cannot be held liable for any errors or claims made by advertisers.

Letters and articles (and we urgently need them) should be sent to:

8:16
c/o 248 Wimborne Road
Oakdale, Poole, Dorset BH15 3EF

We can also be mailed at the X.400 ID: (C:GB, ADMD:TMAILUK, O:SPRINTINTL, FN:Colin, SN:Hunt). If you understand it, you'll probably know how to use it!

This issue of 8:16 was produced using an Atari 130XE, 1050 disk drive, 520STFM, SM124 monitor, Cumana 1M external drive, Apple Macintosh SE and Laserwriter Plus. The software used was AtariWriter Plus on the 8 bit. WordUp, 1st Word Plus, Gem Kermit and ST Xformer II on the ST. Microsoft Word, MacPaint, MacDraw, Mac Kermit and Aldus Pagemaker Version 4 on the Macintosh.

Files are transferred from the 8bit to the ST by directly reading the 8bit disks using a 1050 disk drive connected to the ST printer port and saving the files on a ST disk, with help from the Xformer II software. Standard old Kermit is used to transfer these files to the Mac.

The opinions expressed within 8:16 are those of the authors and are not necessarily held by the BaPAUG.

The BaPAUG is a non profit making organisation.

Copy date for the next issue of 8:16
is 24th June 1991.
Issue date is 29th July 1991

All help welcome and needed!

This issue of 8:16 is dedicated to Aaron and Tracy (again) and to all those people that promise articles and help and then fail to deliver - thanks. It is also dedicated to Phil Lynott (may he sleep in peace), Thin Lizzy and Chelsea FC.






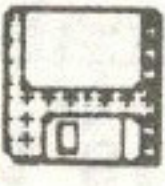
Contents

Page 3

8:16

Issue 12

Features

- Competition Time 5
Our first and maybe our only competition.
Please enter, what have you to lose?
-  PACDEMO Revisited 6
The original 8:16 Turbo BASIC demo re-written in PL65, and a tutorial on how to approach a programming task.
By Simon Trew.
-  Did You Know 13
A short article telling you how to switch between MAC/65 and BASIC without re-booting.
By Paul Bird.
-  Disk Drives & the Device Control Block 14
A quick look at the DCB with specific reference to the disk drives.
By Derryck Croker.
-  AtariWriter Plus Explained for People Who Don't Want to Read the Book 16
Creating and using personal dictionaries.
By Jimmy Boyce.
- ANTIC - The Magazine 17
A quick look at the classic magazine.
By Thomas Holzer.
-  Turbo-Info 19
This issue contains parts 5 and 6 and covers the new 65C816 instructions and how to determine the system configuration.
By Chuck Steinman.
-  Deskjet 500 Review 24
A look at the latest deskjet printer from Hewlett Packard. Article and sample page all printed using the deskjet reviewed.
By Paul Brookes.

Departments

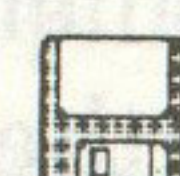
- Notice Board 4
Current Notes Subscription Form 13
Dear 8:16, 22
User Group File 27

Advertisers

- Gralin International 2
Frontier Software 18, 28

 8 Bit Articles

ST Articles



**The Bournemouth and Poole Atari User Group is a member of
The Association of Atari User Groups.**

Notice Board

Compact Storage System

Do you have to get out your computer each time that you wish to use it, and then pack it away again afterwards? Have a cubbyhole some where? Then ECA's **Computer Hideaway** may well be your answer.

Measuring in at only 32" x 27.5" x 18" (HxWxD) when closed, the hideaway looks like any other piece of office/home furniture. It consists of three parts - one for the monitor / TV, one to hold the keyboard and documents and one to hold the printer / paper, with each being independently moveable to create the optimum working conditions.

The Hideaway costs £139 + VAT and is available from ECA, 35 Marston Gardens, Luton, Beds., LU2 7DU. Telephone: 0582 24905.

WANTED

Do you produce Atari related products?
Want some free advertising?
Then send a press release or
information to 8:16 and we will pin it
up on the Notice Board

48 Hour Dispatch - Guaranteed

Frontier Software, who regularly advertise in 8:16, have recently launched a special 48 hour dispatch offer, where they undertake to dispatch all orders for Atari Products within 48 hours of receiving the order. This excludes Saturday and Sunday.

The offer, commencing with immediate effect, could give the customer a further 25% discount if Frontier fail to meet their deadline. Martin Walsh, Frontier Software's Marketing Manager, said "We have a great reputation for offering good service and we now put it to the test. If we fail in this undertaking then the customer will receive a 25% discount.

"Some less responsible suppliers to the Atari market here in Great Britain have brought the Mail Order market place for Atari goods into disrepute. Since our products are designed and manufactured here in Yorkshire we are well placed to provide this speedy dispatch of orders and to offer excellent telephone support. I challenge other suppliers of imported products for the Atari ST range to publicly offer such an undertaking."

New 8bit Titles

Zeppelin Games have just announced the release of two new Atari 8 bit software titles - **Stack Up** and **Jocky Wilson's Compendium Of Darts**.

The first is a mixture of Tetris and Puzznic, while the latter contains 6 dart games.

Zeppelin Games have also stated that they will continue to release Atari titles through the summer - good news for everyone.

GFA-BASIC Supports TT

GFA(UK) have recently released version 3.6 of GFA BASIC. This version supports the 68030 based Atari TT.

GFA-BASIC interpreter code can now be compiled either in the ST or TT ram. The extended graphic modes of the Atari TT, as well as the additional I/O ports can also be accessed using GFA-BASIC 3.6.

GFA Data Media (UK) Ltd.

Box 121, Wokingham, Berkshire, RG11 1FA

(0734) 794941

Portfolio Product Directory

DIP, the original designers of the Atari Portfolio, have released a handy 48 page booklet entitled "The DIP Pocket PC/Atari Portfolio Product Directory". The booklet, which is free, contains information on products currently available for this neat little machine.

Software products include time management applications, databases, terminal emulation / communication systems and very specific applications such as the "Heatcalc" which is a heating system design program.

Hardware products include memory cards/modules, ruggedised Pocket PC (a cool £1200), PC drive card, file transfer cables and peripherals. The most interesting product is the internal memory upgrades which can add another 512K bytes.

The product directory also details some books including one called "The Programmers Guide".

DIP Systems

32 Frederick Sanger Rd., Surrey Research

Park, Guildford, GU2 5XN

Telephone: 0483 301555

8:16 Subscription Rates Annual (4 issues)

U.K.	£4.00
Europe	£8.40
Elsewhere (sea)	£8.40
Elsewhere (air)	£16.00

8:16 Commercial Advertisement Rates

Full page	£45.00
Half page	£25.00
Quarter page	£15.00

Phone (0202) 677895 for details.

T Competition m e

Welcome to 8:16's first competition. Below you will find a list of 40 words, all related to Atari Computing. Beside the list is a huge 20x20 word grid - all you have to do is find all the words within the grid and return it (photocopies are allowed), with your name, address and computer type to 8:16 before August 25th 1991. Easy!

Prizes? Oh yes, I nearly forgot. Below is a list of prizes, kindly denoted by Frontier Software, Gralin International and the BaPAUG.

ST: **Forget Me Clock II**
8:16 Subscription

8 Bit: **US Doubler**
BASIC XL Cartridge
8:16 Subscription

Even if you don't find all the words, still send your entry in. You never know, you may be the only one who does!

Ideas for future competitions also welcome.

- | | |
|---------------|--------------|
| ADVENTURE | LYNX |
| ANALOG | MONITOR |
| ANTIC | MOUSE |
| ARCADE | PADDLES |
| ASSEMBLER | PAGESTREAM |
| ATARI | PASCAL |
| BACKUP | PILOT |
| BAPAUG | POKEY |
| BASIC | PRINTER |
| DESKTOP | PUBLICDOMAIN |
| FLOPPYDISK | SHAREWARE |
| FORGETMECLOCK | SOFTWARE |
| FRAMEGRABBER | SPREADSHEET |
| FRONTIER | STACY |
| GRALIN | STARRAIDERS |
| HARDWARE | STFORMAT |
| JOYSTICK | TOUCHTABLET |
| KEYBOARD | WARGAME |
| LANGUAGE | WIMP |
| LIGHTGUN | ZORK |

Name:

Computer Type: 8-bit / ST

Address:

T	O	O	H	I	L	F	L	O	P	P	Y	D	I	S	K	E	O	N	D
P	O	K	E	Y	B	O	A	R	D	P	Y	E	C	A	E	I	F	F	E
A	A	L	S	Q	Z	Z	N	U	G	T	H	G	I	L	K	R	O	R	S
D	E	L	I	S	T	F	O	R	M	A	T	W	O	K	O	A	T	A	K
D	A	G	Q	P	B	U	E	R	E	O	A	M	C	N	N	T	S	M	T
L	S	J	O	Y	S	T	I	C	K	R	N	O	T	T	I	A	R	E	O
E	S	I	U	L	P	A	R	T	G	R	L	I	E	R	S	E	G	P	
S	E	E	G	A	U	G	N	A	L	C	E	C	T	C	O	H	D	R	U
R	M	E	E	E	K	R	M	B	E	R	O	T	A	O	S	Y	I	A	B
E	B	N	L	R	C	E	I	M	P	D	R	D	N	D	R	T	A	B	L
O	L	E	D	A	A	D	T	P	Y	T	E	E	A	I	E	P	R	B	I
U	E	R	E	W	B	E	D	S	C	G	M	E	I	H	R	U	R	E	C
N	R	A	T	D	G	S	E	W	A	S	R	O	U	U	U	P	A	R	D
J	A	W	A	R	U	A	N	I	T	P	P	C	U	I	T	I	T	T	O
M	W	T	O	A	R	S	I	M	S	Y	A	I	N	S	N	U	S	A	M
B	E	F	G	H	P	G	L	P	A	G	E	S	T	R	E	A	M	R	A
F	R	O	B	C	A	N	A	L	O	G	L	A	T	O	U	I	U	A	I
E	A	S	H	A	B	C	R	N	L	H	O	B	A	M	D	L	E	O	N
I	H	O	D	A	R	Y	G	S	X	C	P	A	S	C	A	L	Y	N	X
N	S	T	U	T	O	U	C	H	T	A	B	L	E	T	O	M	I	B	S

PACDEMO Revisited

Regular readers of 8:16 will have seen Gavan Moran's PACDEMO in Issue 9. This was a small Turbo BASIC program to show the ability of Turbo BASIC above Atari BASIC.

Whilst reasonably well written, considering he had to fit it into a page, I still think it could have been improved. And choosing Turbo BASIC was a bad move really, as it highlights the fact that Turbo BASIC has no built in sprite control.

My PACDEMO accomplishes much the same as Gavan's but I'm writing mine to show you PL65, a compiled language for the Atari that compiles to native machine code. Actually most of this article explains PL65 and Turbo BASIC and very little is actual program, so don't worry!

The first thing that a Turbo BASIC user will note is that PL65 has no built-in library to speak of. There are no flashy TEXT commands, GRAPHICS commands, SIN functions and so on. It is a 'skeleton' compiler: you have to build all the commands yourself, except for the very basics such as memory manipulation, arithmetic and logic. But this makes the compiled code much smaller and there is no need for a large runtime library like Turbo BASIC programs.

The compiler writers, Noahsoft, supply a bundle of libraries for handling I/O, sound, sprites, fonts and so on, but I don't think I need any of them for this exercise. Conceivably I could use GRAPHICS for the graphics calls (GRAPHICS, PLOT, FILL and so on) but I will design my own display list and screen.

The Design

The first thing to do is to decide how I'm going to design the program. Since the objective has already been designed (by Gavan), I'll get on to the modules. First I need all the data that the program is going to need, that is:

- The shapes to put in the player/missiles;
- The text to put on the screen.

Gavan didn't do too well on the first one here. He put the data in cryptic DATA statements and then POKEd them into an area of memory at run time. He also seemed unsure as to how much data he'd be using, as he used a DO/LOOP rather than FOR/NEXT. I can define my data to be much more readable, because the length of the source code bears no relation to the length of the compiled code (some Basic thinking people say I go overboard with my commenting because of this, but this makes the program self-documenting and I hope it is far more legible to those not familiar with PL65).

As the data is a bit pattern, it seems sensible to use binary numbers. What I tend to do is put a 1 in all the spaces that the bit should be set, and leave the others blank. Then I go back and fill in the zeros. The numbers are all byte-sized so I will use an array of type BYTE to hold the data. The array can be given initial values at compile time - this means that there is no need to POKE data at run time.

I have also decided that in future I might want to expand the size of the data, so I am declaring a constant 'height' which is

the height of the player. If I want to change it, all I need do is change the value of 'height' and add the necessary data. The value of 'height' (11) will be substituted when the program is compiled, so using 'height' does not take any more memory than using 11 directly. Constants thus have two uses:

- To enable programs to be modified more easily;
- To enable values to be given meaningful names.

The second use I shall demonstrate later.

And why have one clump of data when there are four shapes? I'll have a separate data entity for each shape. There is no overhead for defining new variables. My definitions are shown in Listing One. There are three points to make about this code:

- My binary values are preceded by the % sign, to indicate that they are binary. Of course, all blanks between the % and the , should be substituted with zeros.
- No line numbers! PL65 code is created on a text editor or word processor. It is called a 'free-form' language.

```
CONST height = 11 ghost_2[height] DATA
% 111111 ,
%11111111 ,
%1 11 1 ,
BYTE pacman_1[height] DATA %1 11 1 ,
% 1111 , %11 111 1 ,
% 111111 , %11111111 ,
%1111 111 , %11111111 ,
% 1111111 , %11111111 ,
% 111111 , %11111111 ,
% 111111 , % 1 1 1 1 ,
% 111111 , % 1 1 1 1 ;
% 11111111 , pill_1[height] DATA
% 1111111 ,
% 111111 ,
% 1111 ; pill_2[height] DATA
% 111111 ,
% 11 11 ,
% 11 11 ,
% 111111 ,
% 1111 ,
% ,
% ,
% ,
pacman_2[height] DATA % 111111 ,
% 111111 ,
%1111 111 , % 111111 ,
%11111111 , % 1111 ,
%11111111 , % ,
% 1111 , % ,
%11111111 , % ,
%11111111 , % ,
%11111111 , % ,
% 111111 , % ,
% 1111 ; ghost_1[height] DATA
% 111111 ,
%11111111 ,
%1 11 1 ,
%1 11 1 ,
%1 111 11 ,
%11111111 ,
%11111111 ,
%11111111 ,
%11111111 ,
% 1 1 1 1 ,
% 1 1 1 1 ;
```

Listing One: PMG Data

By Simon Trew

Anywhere a space can appear you can also place a comment or a carriage return. So I have my data split over several lines to make it line up. It could have been all squashed up on one line if I had wanted. By the way, I always use lower case for my variables. PL65 does distinguish between case, unfortunately, so that 'BERT' and 'Bert' are two different variables. Keeping them in lower case avoids errors and differentiates my code from PL65 language words (such as 'BYTE') and the supplied libraries, which are all in upper case.

- c) The 'BYTE variable[index]' is like a DIM statement. It allocates a block of memory of size index for the variable specified. This is done at compile time. A minor point is that arrays have indices in square brackets, like Pascal. Data items are separated by commas, and the last one is terminated with a semicolon. The comma after the semicolon indicates that another BYTE variable follows. I could have written

```
BYTE pacman_1 DATA x,x,x,x;  
BYTE pacman_2 DATA x,x,x,x;  
BYTE ...
```

just as easily.

The Graphics Screen

Right, now onto the graphics screen. This is a tough one. There is no TEXT command and no libraries contain one. I thus have the three options:

- Use raw graphics data;
- Write an equivalent of the TEXT command;
- Don't use big letters but do something more impressive instead.

I opt for the second one, but it won't be easy - the TEXT command is very powerful. You have to get used to missing

```
BYTE display_list[40] DATA  
blank_8_lines,  
blank_8_lines,  
blank_8_lines,  
load_memory_scan+os_graphics_3,  
screen AND 255, !See below!  
screen / 256, !See below!  
os_graphics_3, os_graphics_3,  
os_graphics_3, os_graphics_3,  
os_graphics_3, os_graphics_3,  
os_graphics_3, os_graphics_3,  
os_graphics_3, os_graphics_3,  
os_graphics_3, os_graphics_3,  
os_graphics_3, os_graphics_3,  
os_graphics_3, os_graphics_3,  
os_graphics_3, os_graphics_3,  
os_graphics_3, os_graphics_3,  
os_graphics_3, os_graphics_3,  
os_graphics_3,  
wait_for_vblank,  
display_list AND 255,  
display_list / 256;
```

Listing Two: The Display List

commands in PL65. But after a while you build up a nice set of commands you have written yourself. You should also choose the language to fit the job and not the other way about. PL65 is hopeless at number-crunching (it has no floating-point built in) so I would use Turbo BASIC for that purpose.

Right, so I have to design a display list and a screen. The screen is easy - just a large array:

```
CONST  
screen_width = 10,  
screen_height = 24  
  
BYTE  
screen[screen_width*screen_height]
```

It would be tedious to fill it all in with zeros at compile time so I will do that with a run time statement.

The display list is also quite easy. I use lots of constants to make it readable. This is the second advantage of using constants: although it isn't useful to change the value of the constant, it makes the purpose of the value much clearer. To write:

```
pause(sixty_milliseconds)
```

is much more meaningful than to write:

```
pause(3)
```

which might mean 3 seconds, 3 tenths, or to wait until something happens at port 3, or whatever. So I can construct a display list - a small program for Antic - in a really readable way. How many programs have you seen in BASIC that use a custom display list? Lots. And how many are just meaningless DATA statements? But I can do:

```
CONST  
load_memory_scan = $40,  
wait_for_vblank = $41,  
os_graphics_3 = $8,  
blank_8_lines = $70,  
display_list_interrupt = $80,  
horizontal_scroll = $10,  
vertical_scroll = $20
```

The last three constants aren't used at present. This is certainly sloppy programming. But they don't take up any more memory, and I can include a comment saying that they are for future expansion. My display list is shown in Listing Two. To explain the !See Below!. The exclamation marks indicate a comment - as I said, comments can be placed anywhere that a space may be. A comment extends from the exclamation mark until another exclamation mark or a carriage return.

Usually, referencing the variables 'screen' and 'display_list' would imply values stored in those variables. However, in a compile-time expression, that is, an expression that is evaluated when the program is compiled rather than when it is run, referencing any variable means 'the address of'. So if I write:

```
INT fred DATA bert;
```



PACDEMO Revisited

then the initial value of the variable 'fred' is the address of 'bert', NOT the value of the variable 'bert'. Compile time expressions are expected:

- As the value of a CONST.
- In DATA elements.
- As the operands to all assembly-language instructions.

If I want to find the address of a variable at run time, I can use the dot operator, which acts like the ADR function in Basic. The address of a variable is known at compile time, because it is allocated a fixed space in memory. So the expression 'screen AND 255' can be calculated at compile time, and the result will be placed in the array data. The 'screen AND 255' and 'screen/256' compute the low and high bytes of the address of the screen array. This is rather a clumsy method, it would be much better if one could write something like '<expr' for the low byte and '>expr' for the high byte.

Player Missile Graphics

Right, now I'm ready to write some code! Let's start off with something simple, setting up PMGs. Oh No! Not PMGs! Have you seen how many POKEs that takes in BASIC? In PL65, I won't use any POKEs. Why? Because there is no POKE statement. BUT, much better, variables can be assigned to absolute memory locations. So, I could assign a variable:

```
BYTE cursor=752
```

This does not make the variable cursor be initialised with the value 752. That would be accomplished with:

```
BYTE cursor DATA 752;
```

Remember, BYTE is like a DIM, except that EVERY variable must be defined in advance, not just arrays. The equals sign means that the compiler does not reserve any memory for the variable, but that it sits in location 752. It is a bit ugly having to use the equals sign, something else such as the word ABS would have been far better, but I'm stuck with it.

I also have to have a player/missile area set aside. This must fall on a 1K boundary, so I have to adjust the compiler location counter to make sure this happens. This is done in much the same way as assembler, except that the @ symbol rather than the * symbol is used for the compiler location counter. So, a procedure to set up PMGs could be written as shown in Listing Three. The procedure is quite long, but its meaning should be obvious: you know what colour is assigned to colour zero. A couple of interesting new features are also present here. First is the idea of local constants and variables. Any variable or constant declared between a PROC and its BEGIN is LOCAL. This means that it cannot be seen outside that procedure. It enables variables to be hidden from the outside world, and more importantly avoids conflict when two procedures try to use the same variable for different purposes. Consider a procedure in Turbo BASIC to find the factorial of a number, for example the factorial of $7=7*6*5*4*3*2*1$. I

could write a procedure:

```
9000 PROC FACTORIAL
9010   FOR LOOP=ARGUMENT-1 TO 2 STEP -1
9020   ARGUMENT=ARGUMENT*LOOP
9030   NEXT LOOP
9040 ENDPROC.
```

This takes a variable 'ARGUMENT' and computes its factorial, storing the result in 'ARGUMENT'. But it also uses the variable LOOP. What if I wanted my calling procedure to print the factorials of all numbers between 1 and 20? I might write:

```
10 FOR LOOP=1 TO 20
20   ARGUMENT=LOOP
30   EXEC FACTORIAL
40   PRINT LOOP,ARGUMENT
50 NEXT LOOP
```

But my 'LOOP' variable would be corrupted. The only ways around this problem are to use lots of different variables or set up a software stack. It also poses another problem: my

```
@=(@+$3FF) AND $FC00
BYTE pm_area[$400]

BYTE
  colour[9]=704

CONST
  yellow   =$F8,
  blue     =$88,
  white    =$0E,
  black    =$00,
  red      =$24,
  green    =$08

PROC set_up_pmg()
  CONST
    allow_pmg=3,
    allow_dma =46
  BYTE
    graphics_control   =$D010,
    memory_control     =$22F,
    graphics_priority  =623,
    player_x_position[4]=$D000,
    pm_base             =$D407
  INT
    dlist               =560
  BEGIN
    !Put the addr of the display_list
    !into the variable dlist
    dlist               =.display_list
    graphics_control=allow_pmg
    memory_control     =allow_dma
    colour[0]=yellow
    colour[1]=blue
    colour[2]=white
    colour[4]=red
    colour[5]=green
    colour[8]=black
    pm_base=.pm_area/256
    FILL(.pm_area,?pm_area,0)
    FILL(.screen,?screen,0)
  END
```

Listing Three: Procedure to setup PMGs

procedure 'FACTORIAL' is really doing the job of a function. I pass it a parameter, and it gives me a result. But I have to explicitly set and retrieve the result in a variable. Why can't I design a PROC like a built-in function such as SIN, so I could type 'PRINT FACTORIAL(12)?' (This facility is available in many BASICS in a limited way).

In PL65 I could design a procedure:

```
FUNC factorial(INT argument)
  INT loop
  BEGIN
    FOR loop=arg-1 DOWNT0 2 DO
      arg=arg*loop
    NEXT
  END argument
```

I could then have another procedure:

```
PROC first_twenty_factorials()
  INT loop
  BEGIN
    FOR loop=1 TO 20 DO
      WRITE(loop)
      WRITE(factorial(loop))
    NEXT
  END
```

```
PROC write_char(BYTE char,x,y,colour)
  CONST
    char_height=8
  BYTE
    character_set[$400]=$E000,
    value
  POINTER
    screen_addr
  INT
    store_value,
    loop
  BEGIN
    ASL colour ASL colour ASL colour
    ASL colour ASL colour ASL colour
    screen_addr=.screen+screen_width*y+x
    FOR loop=0 TO 7 DO
      value=character_set[char*chloop]
      store_value=0
      LDY #7
      :loop2
      ASL value
      BCS notzero
      LDA #0
      BEQ store
      :notzero
      LDA colour
      :store
      ASLA
      ROL store_value ROL store_value+1
      ASLA
      ROL store_value ROL store_value+1
    loop2
    LDY #0 LDA store_value+1
    STA (screen_addr),Y
    INY LDA store_value
    STA (screen_addr),Y
    screen_addr=screen_addr+screen_width
  NEXT
  END
```

Listing Four: Procedure to write text to the screen

The two 'loop' variables would be kept quite distinct. Also, because I had called the factorial procedure a 'FUNC', it returns a value and can be used in an expression. The 'INT argument' in brackets indicates that the function takes one parameter, and when the function is called copies it into a local variable called argument. The 'END argument statement' means that the result of the function is the value in the local variable 'argument'. PROCs and FUNCs can both take parameters, but of course only FUNCs return values. There are still flaws in this approach: The function can only return one value, and it is not immediately determinable which parameters do what in a procedure. Suppose I had written a procedure to position the cursor:

```
PROC position(INT x=$55 BYTE y=$54)
  BEGIN END
```

This is much like the Basic POSITION command. I could call it with:

```
position(7,9)
```

but it would not be immediately obvious whether 7 was the X coordinate or the Y coordinate (or a colour, or a graphics mode...).

Anyway, back to the plot. The other interesting thing about the procedure is the FILL statement. There are some calls such as PUSH, PULL, MOVE and FILL which are built in to the runtime code and which perform essential operations such as moving strings about. Sensibly, the compiler writers decided to allow the user access to these calls and so one can push and pull values to the software stack, and move and fill blocks of memory. Turbo BASIC has a MOVE command, or rather it has two. It is up to the user to decide whether he needs to copy forwards or backwards. This only actually matters when the source and destination blocks of memory overlap. The PL65 MOVE command has direction sensing, so blocks will always be copied correctly. Strangely, Turbo BASIC has no FILL command. This fills an area of memory with a value. Gavan Moran had to do a tedious POKE in a big loop. Actually, if he had been clever he could have used the MOVE command, but even so it would have been a bit messy. For example, if he wanted to fill 2000 bytes starting at \$6000 with the value 67:

```
10 POKE $6000,67
20 MOVE $6000,$6001,2000
```

Recognise the trick? What about the age-old string-filling trick:

```
10 DIM A$(10000)
20 A$(1)=" " : A$(10000)=" " : A$(2)=A$
```

As I said before, FILL is a much more elegant solution to this problem. I wanted to fill the 'pm_area' with zero, so I used the statement:

```
FILL(.pm_area,?pm_area,0)
```

The . operator I have explained before, it means



PACDEMO Revisited

'address of'. The ? operator means 'the number of elements in' and generally it means the number of bytes taken up by a variable. The exception to this rule is if using INT variables. INTs take up two memory locations for every element and so the above statement would only fill half the area if 'pm_area' was an INT.

Next I need a procedure to copy a shape definition into a player stripe at a set of coordinates specified. I could pass the address of the shape position, or a symbolic value which indicated which one to copy. Since both are just as readable I shall choose the former as it's simpler. My procedure will be:

```
PROC copy_shape(INT address BYTE player,x,y)
CONST
  player_start=pm_area+512,
  player_height=128
BYTE
  player_x_coord[4]=$0000
BEGIN
  MOVE(address,height,player_start+y+player_
height*player)
  player_x_coord[player]=x
END
```

Note that the MOVE procedure in PL65 has the syntax:

```
MOVE(from,bytes,to)
```

whereas in Turbo BASIC it has the syntax:

```
MOVE from,to,bytes
```

Writing Text

That was easy! Next, I'll need a procedure to write a character onto a graphics screen. This is the toughie. I haven't written one before so this will be a useful addition to my library. For the moment I will restrict it to Graphics 3, and so have the following considerations:

- Graphics 3 is a four-colour mode, using 2 bits per pixel.
- Each character thus takes up two bytes horizontally by eight down.
- Characters are stored in a bit map starting at \$E000, each is eight bytes high.
- Characters aren't stored in ASCII order, but an 'internal' order.

So I will first write a routine that takes an internal character and an X and Y coordinate, and puts the character on the screen. For each byte I will have to split it into 2 and add the colour information. Some of it would probably be easier to write in assembly language. This is one of PL65's fortes because assembly language can be used anywhere in a program with almost complete freedom (see Listing Four).

What's new here? Firstly, the type POINTER. This operates much like INT, but in actuality it is assigned space in zero page so that it has a few advantages. Firstly, it is accessed quicker. Secondly, some assembly language instructions including a couple that I'm using will only work with zero page addresses. Thirdly, BASED variables can be defined. I

hardly ever use BASED variables although I suppose they could be quite powerful. They allow a variable to rove about in memory. For example, if I defined:

```
POINTER location
BYTE value BASED location
```

This would mean that 'value' would have no fixed abode in memory. Its address is determined by the value of 'location'. So if I then used

```
location=709
value=33
```

33 would be placed into memory location 709. Similarly if I did:

```
location=56
WRITE(value)
```

the contents of memory location 56 would be written to the screen. It took me a while to grasp that, but essentially it's quite straightforward.

Another new feature is the use of labels. Labels are much like labels in Turbo BASIC, and they are defined by using a colon followed by the label name. Labels are always local. Labels can be used in assembly language branch instructions, which is my main use, but they can also be used in the dreaded GOTO.

Lastly assembler users will have noted that I used ASLA rather than ASL A. This instruction is written such so as not

```
!a!
PROC write_string(BYTE x,y,colour
                  STRING text$[40])
BYTE
  text_chars[?text]=.text,
  internal,
  ascii
INT
  textlen=.text-2,
  loop
BEGIN
  FOR loop=0 TO textlen-1 DO
    ascii=text_chars[loop]
!b!
    IF ascii<32 THEN
      internal=ascii+64
    ELSE
      IF ascii<96 internal=ascii-32
    ELSE
      internal=ascii
    ENDIF
  ENDIF
  write_char(internal,x,y,colour)
  INC x
  INC x
  IF x>=screen_width THEN
    x=x-screen_width
    y=y+1
  ENDIF
NEXT
END
```

Listing Five

to confuse it with an instruction to shift the variable A. The same applies for ROLA, LSRA, and RORA.

Now I have my basic character plotting procedure, I can write one to plot a string, such as the procedure show in Listing Five. This has many new features, all based around string handling. String handling in PL65 is at least as good as Turbo BASIC. All strings must, unfortunately, end in the \$ sign, but this is dropped in compile-time expressions which is rather curious. The \$ does not form part of the name, it is just expected at the end in high-level code. I have defined a byte array called 'text_chars' which overlays the variable 'text\$'. This enables me to use the string when doing string processing and considering each character of the string as a byte when that suits. This means I needn't do a wasteful ASC conversion.

The textlen integer is used for a similar purpose. Two integer variables are declared below the string's address, so that one resides at '.text-4' and the other at '.text-2'. The first holds the maximum length of the string, that is '?text', and is used by the string handling routines to ensure that the string bounds are not exceeded. The second holds the dynamic length of the string at run-time. Again, I could write functions to return these values if I wanted to.

```
PROC turn_around(POINTER address BYTE
height)
BYTE rotate,savex
BEGIN
STX savex !Have to preserve the X reg
LDY height BEQ no_turn
DEY
:loop
LDA (address),Y
LDX #8
:loop2
ASLA
ROR rotate
DEX
BNE loop2
LDA rotate
STA (address),Y
DEY
BPL loop
:no_turn
LDX savex
END
```

Listing Seven: Procedure to turn PACMAN around

```
PROC pause(BYTE jiffies)
BYTE clock=20
BEGIN
LDA jiffies
BEQ end_pause
:loop
LDA clock
:loop2
CMP clock
BEQ loop2
DEC jiffies
BNE loop
:end_pause
END
```

Listing Six: PROC pause

An important point to remember is that all array indices start at zero in PL65. This includes strings, so that the last element in use in the string 'text\$' is 'text\$[textlen-1]' and not 'text\$(textlen)' as it would be in BASIC. In BASIC, numeric arrays start at zero and string arrays at one - a curious set-up, really. Also, a special form of the '+' operator is available. How often do you use BASIC statements like:

```
NAME$=DEVICE$
NAME$(LEN(NAME$)+1)=FILENAME$
NAME$(LEN(NAME$)+1)=EXTENSION$
```

Well, in PL65 you could write:

```
name$=device$
name$+filename$
name$+extension$
```

An interesting and very useful feature is that a string is passed to a procedure as two integers - an address and a length. This is normally used to copy the string into the local variable (in this case 'text\$') but I can access the two values directly, and the compiler will accept it. In fact, this is much more useful in many cases. I could write ASC and LEN functions very easily with:

```
FUNC ASC(POINTER address INT length)
BYTE result
BEGIN
LDY #0 LDA (address),Y STA result
END result

FUNC LEN(INT address,length)
BEGIN END length
```

So I would rewrite the top of the write_string function as:

```
PROC write_string(BYTE x,y,colour
INT address)
POINTER loop
BYTE ascii BASED loop
BYTE internal
BEGIN
FOR loop=address TO address+length-1 DO
```

and delete all lines between the comments !a! and !b! in the first version. I hope this makes sense, it does to me. Strings are tricky especially when it comes to substrings, which are handled as in Turbo BASIC. Of course, one could write a left\$, right\$ and mid\$ function.

By the way, have you noticed that FOR loops have no variable names after their NEXTs? The compiler keeps track of which NEXT belongs to which FOR.

Next I need a pause procedure (Listing Eight). Again this is very easily done in assembly language, in fact easier and quicker to write than in high level code. If you aren't well up on assembly language, then don't worry. All it does is counts down the jiffies value every clock tick (once every jiffy) until it reaches zero, then it exits.

Nearly there now. The last touch is an aesthetic one. As Gavan Moran's pacman chases the ghost, he



PACDEMO Revisited

goes backwards. This stance is not particularly effective, so I'll have a procedure (Listing Seven) to turn around a set of bytes horizontally, which I can then use to turn around the pacman:

Now I need a procedure (Listing Eight) that will plot all the players in the correct places and adjust the colours. This will be passed a flag which, if zero, will mean that the power pill is not plotted but instead is overwritten with the blank_out. This array will be zero filled at run time.

Now all I need is a brief controlling procedure to control the whole program (Listing Nine). In Turbo BASIC, a program starts running at the first line. In PL65, it starts at the first statement in a special type of procedure called MAIN. I try to keep my MAIN as simple as possible. Of course, it could be made simpler but it is a balance between simplicity and functionality. Make a procedure too small and it does so little as to be useless, too big and it might as well be a separate program.

That's it. Everything there should be fairly readable for a Turbo BASIC user. Of course, REPEAT..FOREVER is an infinite cycle, and it's not worth trying to break out of the program.

```
CONST
pacman_player=0,
ghost_player =1,
pill_player =2

BYTE
pacman_x, pacman_y,
ghost_x, ghost_y,
pill_x, pill_y,
blank_out[height]

PROC plot_players(BYTE plot_pill)
BEGIN
IF pacman_x MOD 8<4 THEN
copy_shape(.pacman_1,pacman_player,
pacman_x,pacman_y)
ELSE
copy_shape(.pacman_2,pacman_player,
pacman_x,pacman_y)
ENDIF
IF ghost_x MOD 9<4 THEN
copy_shape(.ghost_1,ghost_player,
ghost_x,ghost_y)
ELSE
copy_shape(.ghost_2,ghost_player,
ghost_x,pacman_y)
ENDIF
IF plot_pill THEN
IF ghost_x MOD 15<6 THEN
copy_shape(.pill_1,pill_player,
pill_x,pill_y)
ELSE
copy_shape(.pill_2,pill_player,
pill_x,pill_y)
ENDIF
ELSE
copy_shape(.blank_out,pill_player,
pill_x,pill_y)
ENDIF
END
END
```

Listing Eight: Procedure to plot PMGs

Summary

Although that's a lot of text, it's a lot of topics I've covered. Not only a simple demo, but an introduction into most of the main points of PL65 (I missed out CASE, ON, ERROR, TRAP, INTERRUPT - maybe next time!) and a few notes on Turbo BASIC. As I say, we have written the program from scratch, and after a short while using PL65 you build up a big library of routines such as pause, pos, pop_up and so forth. My library now has write_string added to it.

I hope you found it informative and an insight into other languages. PL65 is a really good language for assembler lovers. If you're happy with Turbo BASIC then stick to it, and don't get put down by people who think BASIC is for beginners. But are you really happy? As I said before, choose the language to fit the job and not the other way about.

My thanks must go to Gavan Moran who has often suggested good ideas which I have then nicked and developed. When are you going to improve my Cube then Gavan?

```
CONST
twenty_milliseconds=1 |jiffy!

MAIN()
BEGIN
set_up_pmg()
FILL(.blank_out,?blank_out,0)
pacman_y=56
ghost_y =56
pill_y =56
REPEAT
colour[ghost_player]=blue
pacman_x=208
ghost_x =221
pill_x =50
REPEAT
plot_players(1)
DEC pacman_x
DEC ghost_x
pause(twenty_milliseconds)
UNTIL pacman_x=pill_x
write_string(1,2,1,"PL65")
write_string(0,12,2,"POWER")
turn_around(.pacman_1,?pacman_1)
turn_around(.pacman_2,?pacman_2)
REPEAT
plot_players(0)
INC pacman_x
INC pacman_x
IF pacman_x MOD 12<>0 THEN
INC ghost_x
ENDIF
IF ghost_x MOD 12<6 THEN
colour[ghost_player]=blue
ELSE
colour[ghost_player]=white
ENDIF
pause(twenty_milliseconds)
UNTIL pacman_x>ghost_x
write_string(1,2,0,"PL65")
write_string(0,12,0,"POWER")
turn_around(.pacman_1,?pacman_1)
turn_around(.pacman_2,?pacman_2)
FOREVER
END
```

Listing Nine

Did You Know ...

By Paul Bird

Switching between MAC/65 and BASIC without re-booting.

A few years ago I saw a letter within one of the Atari magazines, from an 8-bit user who had the MAC/65 cartridge and was concerned about wearing out his edge connector when switching between the MAC/65 and BASIC (he had to remove the cartridge). Recently, curiosity drove me to open my MAC/65 and draw the circuit diagram. I have since deduced that the cartridge can be switched on and off by software control, thus enabling full use of the computer with the cartridge in place. It is rather sad that this information has now come so late in the day - even five years ago it would have been of only minor importance. However, for what its worth, here are the details.

The banked switched MAC/65 has a 16K EPROM addressed in 8K using software controlled switching of the EPROM to access 4 x 4K sections. The section from \$B000 (HEX) to \$BFFF is addressed normally. The section from \$A000 to \$AFFF is switched one of three ways

by address decoding using readily available TTL chips (2 x quad NOR gates and a dual D flip flop). An examination of the code, using MAC/65's DDT shows the switching codes STA \$D500, STA \$D501 and \$D509. These three addresses fall within an unused sector of ROM which has the advantage of switching the CCTL line on the edge connector. So any write to \$D500 - \$D5FF will switch CCTL. The MAC/65 decodes this to switch the EPROM.

For our purposes all we need to do is switch the cartridge on and off using STA \$D508 (off) and STA \$D500 (on). Incidentally the third digit is irrelevant. It is only the D% and the last digit which actually get decoded "bit wise".

So, to demonstrate. Switch on your disk drive, insert disk with DOS, switch on your computer with MAC/65 installed and wait for the screen to come up with DOS or MAC/65. If DOS on screen then enter cartridge mode (option B for DOS 2.5, CAR for SpartaDOS). Type DDT return

followed by E D508 return. The screen goes blank or very similar. Press RESET and wait. Eventually you will get the ready prompt for BASIC. The MAC/65 cartridge has been switched off and the computer ignores its presence. (Theoretically the machine would not crash if you now removed MAC/65 with the computer still powered on, but I haven't tried this). So you play merrily with BASIC. Incidentally if you want to reboot DOS you may have to type POKE 580,1 return and then press the RESET to get a cold start.

Now to return to MAC/65 without a complete re-boot. Type the decimal equivalent of POKE \$D500,0(POKE 54528,0). The value poked is irrelevant because it is the action of poking, ie writing to the hardware that does the switching. Now press RESET and you will be back in the land of MAC/65 very soon, even if you pass through DOS having to type B return (DOS 2.5) or CAR return (SpartaDOS).

Current Notes Subscription Form

Current Notes is published ten times a year by Current Notes inc. It contains informative articles on all the Atari range of computers, specializing in the ST and 8-bit.

Current Notes also have an extensive PD library of standard ST and Spectre format disks.

As a member of the BaPAUG (either full or as an 8:16 subscriber) you are entitled to the reduced subscription rates, payable in US funds:

\$31 / 1 year
\$58 / 2 years

Name:

Address:

Send to: CN Atari Clubs,
122 N. Johnson Road,
Sterling, VA 22170, USA
Telephone: 0101 703 450-4761

You may photocopy this application if you do not want to cut up your copy of 8:16.

Disk Drives & the Device Control Block

by Derryck Croker

Inside every 8-bit Atari computer is a 10 byte table called the Device Control Block (DCB). Each and every device that can be connected to the serial port makes use of the values placed in these registers to control its operation. Table 1 lists the addresses of the various registers, Atari's name for them, and their uses. As you can see, some of them are rather esoteric, others are of great use to the programmer who wants to access the disk drive at its most basic level, that of the sector.

Using the DCB to control the disk drive places the responsibility of sector and memory management with the programmer, a task which is normally taken by DOS, which makes the process of keeping track of which sector belongs to which file transparent to the user. Indeed, the management of the disk drive can be visualised as a tree, with the branches as DOS. Midway down the trunk is a small offshoot called DSKINV, below this is SIO, and right at the very bottom, the roots spread in all directions, hooking into all sorts of unlikely and esoteric places, including the POKEY chip which makes itself responsible for providing the two audio tones sent out on the serial line. These tones represent the on/off status of each data byte and are decoded by each device as addressed by the programmer through the DCB. The programmer who wishes to simply load and save files in the familiar way sees only the canopy from above and does not see any of the tree's supporting structure. As we progress further down the tree, the roots need more and more feeding by the programmer. Luckily for us, DSKINV has its own watering can, and can nourish SIO and hence the disk drive with just a little effort. The registers marked with an asterisk in Table 1 are the only ones that need

to be set up before calling DSKINV, which fills in the remainder before calling SIO. Since DSKINV is only a small branch, it can only perform a limited number of tasks, reading and writing to individual sectors, formatting and performing a status check. This latter command has little practical use, it is used by the computer's boot program to determine whether a disk drive is attached (and if so attempts to boot it) and sometimes as part of software protection schemes and so is beyond the scope of this article.

Controlling the drive on this sector basis has several uses, sector editors and copiers are two that come to mind, however for the purposes of demonstration Listing 1 shows how to format and then write a program to a self booting disk. Boot disks contain a continuous series of sectors which the OS's boot program loads to the intended load address. Since each sector contains \$80 bytes, 2 sectors will therefore load 1 page (256 bytes) of memory. Although it is written in machine language, it is readily adapted to BASIC since much of the program is storing values in the DCB and then calling DSKINV in a loop, incrementing memory buffers and sector numbers until finished. You can call DSKINV with this short program:

```
10 DIM DSKINU$(5):FOR A=1 TO 5:READ B
20 DSKINU$(A,A)=CHR$(B):NEXT A
30 DATA 104,32,83,228,96:REM PLA,JSR 5
8451,RTS
```

and then call DSKINV with DSKINU=USR(DSKINU\$).

The limiting factor of the OS's boot program is that it can load a maximum of 255 sectors, which is just under 32K.

Address	Name	Description
\$0300	DDEVIC	Bus ID. Each device has an ID number, to which it responds. Values are: \$31-\$34 for drives 1-4, \$40 for printer, \$50-\$53 for RS232 ports 1-4, and \$00 for the cassette. (See note 1)
\$0301	DUNIT *	Device number 1-4.
\$0302	DCMND *	Command code. \$21-format (see note 2), \$50-write without verify (See note 3), \$52-read, \$53-status check, \$57-write with verify.
\$0302	DSTATS *	Returns a status code after operation: \$01-success, \$80 plus indicates various failure modes. Also used to set data direction if using SIO direct: \$40-receive, \$80-send.
\$0304	DBUFLO *	Lo byte of memory buffer for data transfer. (See note 4)
\$0305	DBUFHI *	Hi byte of memory buffer for data transfer.
\$0306	DTIMLO	Timeout value. If addressed device has not responded by the time this value has expired, an error is returned.
\$0307	DUNUSE	Unused register.
\$0308	DBYTLO *	Lo byte of the number of bytes to be transferred. (See note 5)
\$0309	DBYTHI *	Hi byte of the number of bytes to be transferred.
\$030A	DAUX1 *	For the disk drive, this is the lo byte of the sector number to be accessed.
\$030B	DAUX2 *	Hi byte of sector number to be accessed.

Table 1: Device Control Block Registers

Notes

1. The cassette does not return any value to SIO since it does not contain its own controller.
2. The format produced is not compatible with DOS, since it does not write boot, VTOC or directory sectors. Only single density formatting is supported unless a modified drive is used and access is made through SIO.
3. The 400/800 OS has a couple of bugs. On occasion, the computer will time out during disk reads (it will resume if left alone), and it does not support write without verify if access is made through DSKINV.
4. The format command will place its sector map in the buffer specified in these two bytes. Always specify a safe memory buffer address before attempting to format. DOS uses \$0400, part of the cassette buffer. Other devices will use differing buffer addresses placed here by their respective handlers. For example, the cassette uses \$03FD (the 3 bytes from \$03FD to \$03FF are used for housekeeping, such as baud rate calculation), and the printer \$03C0. The status command uses \$02EA as a buffer address.
5. Single density formatted disks use a value of \$80 (lo) and \$00 (hi) in these two bytes. Other devices use differing values, for example the cassette (\$84 and \$00. The extra byte is the block checksum), and the printer (\$28 and \$00). The XL/XE OS will allow reading and writing to double density formatted disks through DSKINV using two alternative registers at \$02D5 (lo) and \$02D6 (hi). Use values of \$00 and \$01 respectively, which reflects the sector length of \$0100 bytes. The status command fetches 4 bytes.

Listing 1 can be readily adapted to a 1 sector boot program which continues to load memory up to a user-supplied maximum simply by changing the write command to read and making a suitable modification to the routine at lines 1690 and 1840 (I tend to simply check for DBUFHI being higher than the final load address). Initialisation and run addresses are of course dependent upon the program being loaded.

No error checking is performed by Listing 1, which can be added in one of 3 ways if required. On return from DSKINV, errors will be indicated by the minus flag being set and DSTATS containing a value of 128 or over (1 indicates a

successful operation). In addition, the Y register will hold a copy of DSTATS. This is in keeping with the usage of CIO, and the error codes have the same meaning, although not all will be applicable.

Finally, the format command transfers control to the disk drive's own control program, which does not return to the user until either the format is finished or until the drive times out in the event of an error. Remember that the format produced by accessing the drive through the DCB and DSKINV is not compatible with DOS.

Listing One:

```

1000 .OPT NO LIST,OBJ ;MAC/65 header
1010 ;*****
1020 ;* Listing 1 *
1030 ;* For 8:16 *
1040 ;* By Derryck Croker *
1050 ;* *****
1060 ;*****
1070 ;
1080 ;Format and write a boot disk
1090 ;(No error checks)
1100 ;
1110 ;equates
1120 ;
1130 DUNIT = 769 ;Drive number
1140 DCOMND = 770 ;Command register
1150 DBUFLO = 772 ;Buffer address lo
1160 DBUFHI = 773 ;Buffer address hi
1170 DBYTLO = 776 ;Sector size lo
1180 DBYTHI = 777 ;Sector size hi
1190 DAUX1 = 778 ;Sector number lo
1200 DAUX2 = 779 ;Sector number hi
1210 DOSINI = 10 ;run address
1220 ; for sample boot file
1230 FORMAT = 33
1240 READ = 82
1250 WRITE = 87
1260 DLIST = 560 ;display list
1270 ; address for sample boot file
1280 DSKINU = 58451
1290 ;
1300 ;Format disk-WARNING-wipes disk!
1310 ;
1320 *= $4000 ;relocatable
1330 LDA #1 ;drive 1
1340 STA DUNIT
1350 LDA *FORMAT ;format command
1360 STA DCOMND
1370 LDA * <1024 ;cass. buff.
1380 STA DBUFLO ;is safe
1390 LDA * >1024 ;place for
1400 STA DBUFHI ;sector map
1410 LDA #128 ;sector
1420 STA DBYTLO ;size
1430 LDA #0
1440 STA DBYTHI
1450 JSR DSKINU ;do format
1460 ;
1470 ;Set up buffer and
1480 ;sectors for writing
1490 ;
1500 LDA *WRITE
1510 STA DCOMND
1520 LDA * <START
1530 STA DBUFLO ;buffer lo
1540 LDA * >START
1550 STA DBUFHI ;and hi
1560 LDA #1 ;sector 1
1570 STA DAUX1
1580 LDA #0
1590 STA DAUX2
1600 ;
1610 ;Now we are ready to
1620 ;write to disk
1630 ;
1640 JSR DISKLOOP
1650 FINISH
1660 JMP FINISH ;all done!
1670 ;
1680 SECTORS
1690 .BYTE 1 ;number of boot
1700 ; sectors
1710 DISKLOOP
1720 JSR DSKINU ;write sector
1730 INC DAUX1 ;next sector
1740 BNE OVER ;check overflow
1750 INC DAUX2 ;we won't need this here
1760 OVER
1770 LDA DBUFLO ;increment
1780 CLC ;memory
1790 ADC #128 ;buffer
1800 STA DBUFLO
1810 LDA DBUFHI
1820 ADC #0
1830 STA DBUFHI
1840 DEC SECTORS ;check
1850 ; for last sector to write
1860 BNE DISKLOOP ;not yet
1870 RTS ;all done!
1880 ;
1890 ;A simple boot program
1900 ;
1910 START = $4100
1920 *= START
1930 ;Boot header
1940 .BYTE $00 ;dummy byte
1950 .BYTE $01 ;* sectors to boot
1960 .BYTE $00 ;lo byte load addr.
1970 .BYTE $41 ;hi byte load addr.
1980 .WORD RESET ;reset handler
1990 ;
2000 ;
2010 ;boot continuation
2020 ;
2030 LDA * <RUN
2040 STA DOSINI
2050 LDA * >RUN
2060 STA DOSINI+1
2070 CLC ;end of
2080 RTS ;boot continuation
2090 RESET ; reset handler
2100 RTS ;does nothing here
2110 RUN ; program start
2120 LDA * <DL ;make a new
2130 STA DLIST ;display list
2140 LDA * >DL
2150 STA DLIST+1
2160 ENDLESS
2170 JMP ENDLESS
2180 ;
2190 ;
2200 DL
2210 .BYTE $70,$70,$70
2220 .BYTE $70,$70,$70
2230 .BYTE $47
2240 .WORD TEXT
2250 .BYTE $70,$70,$70
2260 .BYTE $70,$70,$70
2270 .BYTE $47
2280 .WORD NAME
2290 .BYTE $41
2300 .WORD DL
2310 TEXT
2320 .SBYTE " SAMPLE BOOT FILE "
2330 NAME
2340 .SBYTE " by derryck croker "
2350 ;
2360 PROGEND

```


AtariWriter Plus Explained for People Who Don't Want to Read the Book

By Jimmy Boyce (CACE)

Reprinted from Atari Interface Magazine, December 1989

Hi people! How's the old heater doing? Are we having fun yet in this cold? You say, "Who cares, just tell me how to create my own personal dictionary?" My, my...aren't we edgy?

Why a personal dictionary? Mainly because AW+'s dictionary is so small. Another reason is that you may have a specialized vocabulary. An electronics technician, dentist, inventor or teacher of a special subject will each have need for a personal dictionary. If you have more than one specialized subject, you can make a dictionary for each subject.

There are two ways of going about this process. You can run a document through the spell check and then go back to menu and select FILE PERSONAL DICTIONARY or you can just create a list of words and save them.

Creating a Personal Dictionary using Corrected Words

So, let's see what is involved in saving words from a document that you have just checked for spelling <pg 57>. Keep in mind that as you are running a file through the Proofreader and you tell the computer to accept as correct a corrected word, that word is maintained in memory. When you are returned to the menu, press the key that gives you the File Personal Dictionary function in order to save this list of words (the list in memory).

At this point the computer will prompt you for a file name. I like PERDIC followed by a number, such as PERDIC1 or PERDIC2. This way, if you exceed the maximum allowed words (256 words per personal dictionary file), you still have continuity with the dictionary.

Now, insert a formatted disk (you know how to format a disk for AW+) into your drive and hit [RETURN]. If the file exists, you will be asked if you wish to replace the old file, if not press [N] and start over with a new name for your file.

Here is the nifty part of this whole rigmarole (that ought to mess up the

Proofreader but good) — you will be prompted with SELECT WHICH WORDS TO STORE? Y/N. If you select [Y], you can choose the words, letters, abbreviations and numbers that you wish to save. Each time a word flashes on the screen you will be asked if you wish to save it. Just type a [Y] or [N] to save or discard each word.

Now that you are done with the list, you are asked to press [RETURN] TO CONTINUE. I assume you can follow directions and you did indeed press [RETURN]. You will notice that your drive started running. Guess what? Your new personal dictionary is being saved.

Please remember that if your list is OVER 256 words long ONLY the first 256 words will be saved. If you have a list of over 256 words, save it as a regular AW+ file and split it up with the "saving a portion of a file" technique mentioned on <pg 28>. Then save it to your Dictionary disk (that will be discussed next).

If you type an [N] when asked SELECT WHICH WORDS TO STORE? Y/N, the entire list is saved and that is the end of the procedure.

Building a Dictionary Manually

Let's say that you already know that the technical terms you will be using are not in the AW+ dictionary, and you want to get a jump on things and create your own files without using the previous method. Here it is folks: first create a file containing all the specialized words you use. Your options are to create a column with a [RETURN] after each word or just type them with a space between each word. Then, just save this like you would any file but put it on your dictionary file disk with an appropriate file name. Isn't that just precious!

There are a few ground rules and they are quite simple to deal with. Start each word with a letter. Do not use punctuation marks, parentheses or numbers at the beginning of a word. Proofreader will accept ['] in a contraction such as "he'll" or "aren't."

Do not enter contractions showing possession such as "she's." Also, remember that your dictionary can contain up to 256 words (130XE owners — your list can be up to 628 words).

Using a Personal Dictionary

Let's move on to proofreading with personal dictionary files <pg 59>. Go to the main menu and gently depress the [V] key to verify the spelling of your document. Remember when it says to insert the program disk and press [RETURN], in reality it means leave the AW+ program disk in your drive because PROOFREADER is on that side of the disk.

When the PROOFREADER menu appears on the screen, you must gently strike the [A] key to add your personal dictionary. Now that was not too bad was it? Oh, I almost forgot...you must type in the name of your dictionary file (I use PERDIC1) at the prompt at the bottom of your screen and hit [RETURN]. This gets your part of the dictionary into memory so you can use it.

At this time, if you have another dictionary to load then do so, because you should be back at the menu with a notice telling you how many words you loaded last time and asking you to make another selection from the menu. Just repeat the process and the next list of words is loaded.

If you try to load more words than there is allocated memory space for, you will be told that you goofed with a terse little message saying MEMORY FULL — INCOMPLETE LOAD. However, if you have been fortunate and all has gone well, ask the computer to correct your spelling by depressing the [C] key and inserting the AW+ dictionary disk in your drive (don't try to jam that disk in without removing the other one first!).

Updating A Personal Dictionary

OK, you have corrected your sloppy spelling and keyboard work. Now let's update that dictionary disk. As a

ANTIC - The Magazine

By Thomas Holzer

Was it the Ultimate Atari magazine?

Gralin International have recently imported all available ANTIC magazines from the States (more than 70 issues) and they are selling like hot cakes. Even the the old issues dated 83/84 - collector items or old material for new users?

I, being a XE user myself, used to have a decent collection of ANTICs because it was sold in Germany on almost every news stand and magazine shop. In 1988 I sold my 8-bit gear, including the magazines, to try out the old C64 and after two years of boredom I sold that horrible piece of hardware and now I'm back to Atari computing.

ANTIC came first on my shopping list because it was The Atari magazine, even if it was State side orientated. You can still type in those fabulous listings from famous guys (and gals) like Stan Ocker and Linda Schreiber, and they usually work as well.

Further you can read with envy what was / is available for your 8-bit in the States which never reached the shores of Europe. Software like Universe I or Jupiter Mission 1999, all 3 or 4 disk products way back from 83/84. Or hardware, like the 3" disk drive, the 1200XL or ready to plug in modem from Atari.

What I like best in ANTIC are those in-depth interviews with companies like Sierra or Lucasfilm, praising their latest product and promising to continue to support the Atari.

ANTIC also tried to introduce a VCS and 5200 gaming section, but it didn't go down too well and it died after a few issues. But, did you know that Amiga, yes Amiga the company that designed the Amiga computer sold by Commodore, started on the VCS producing a Joyboard which plugged into your VCS and was controlled by you standing on it! With it came a skiing game, so this gives you the general idea of what you did with it!

Later, ANTIC introduced the ST and from then on the magazine was divided into 8-bit and ST sections. Fair enough, the ST is Atari, but after a few years the readers decided they didn't like it and the ST section was dropped - START had been in existence for sometime already. With the dropping of the ST section the readership also dropped and ANTIC got thinner and thinner until in 1990 it was almost a pure listing magazine. ANTIC returned to being bi-monthly in an attempt to stop the trend, but death was unavoidable and now ANTIC is a few pages in the back of START, the magazine ANTIC spawned.

Sad, but thanks to Gralin International you can now buy almost all the ANTICs ever released (along with issue disks). With those great tutorials and listings you can't go wrong, because no one knew the Atari better than ANTIC, and that's a fact.

reminder <pg57>, each time that you correct the spelling of a word that is not on the regular dictionary disk, that word is saved in memory. These words can be added to your personal dictionary when you have completed your spell-checking in the following way: when you have completed spell-checking and have been returned to the Proofreader Menu, depress the [F] key for File Personal Dictionary and you will be asked for a filename. At this point, let me tell you something I discovered about this process. For some reason if you save your dictionary file with the same filename as before, you will have a double list of your words in that file. So, here is what I do to eliminate the problem:

- Step 1: Save the file with another name (e.g., PERDIC2 assuming that your original file was called PERDIC1).
- Step 2: Delete the original file (PERDIC1).
- Step 3: Reload PERDIC2.
- Step 4: Save PERDIC2 with the new name PERDIC1.
- Step 5: Delete PERDIC2.

This may sound a bit much, but it will leave your disk un-cluttered. Plus, making it easier to load your dictionary next time, because you will not have to remember which file is which.

The next thing you will be asked is if you want to select the words you want to save. I always tell it [Y] just so I can eliminate anything I do not wish to have on the disk. One final note: save the file while in the Proofreader program because some of the 130XE's do not respond when returning to AW+ and your corrected file is lost upon re-booting AW+.

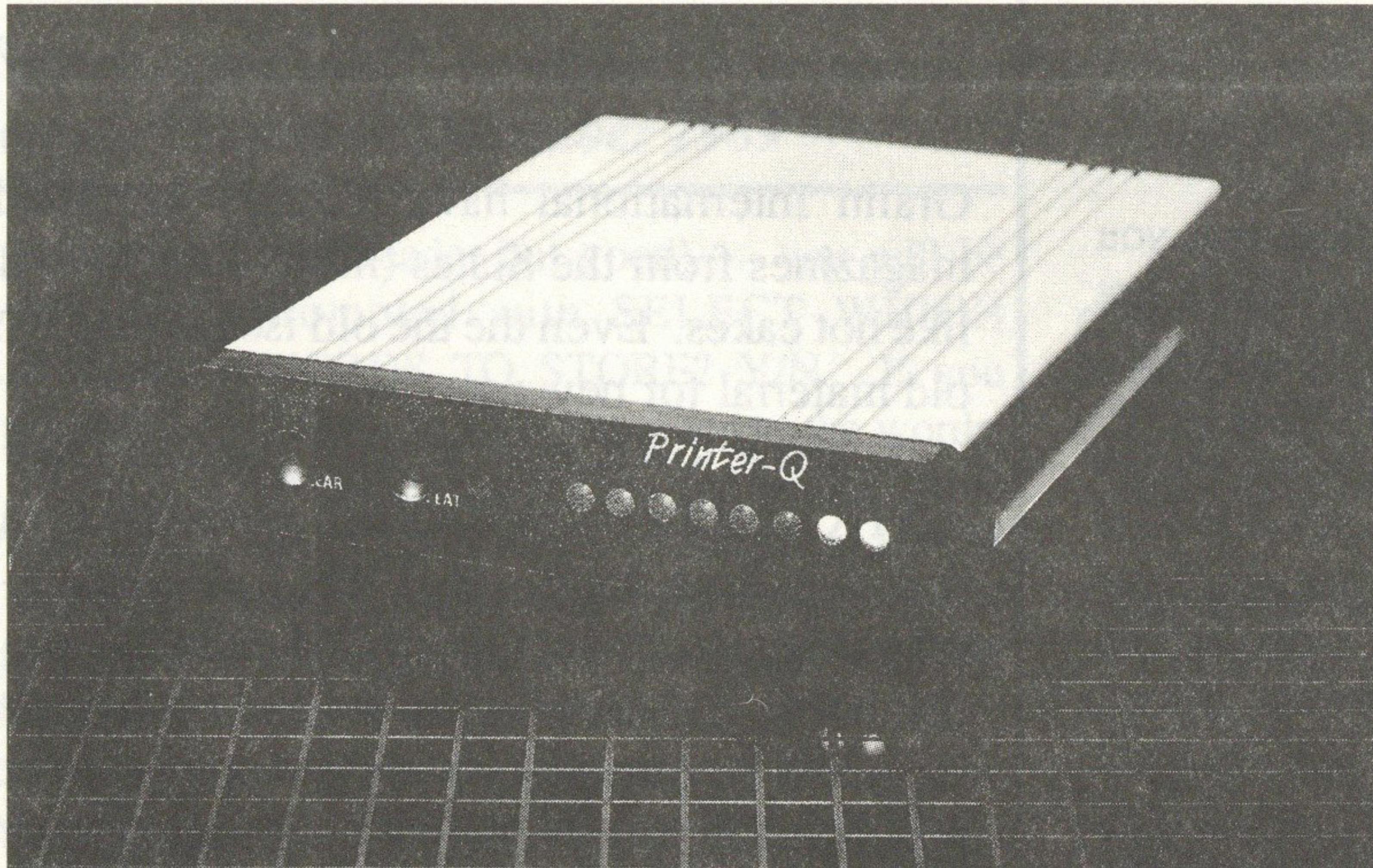
That's about the sum and substance of checking your spelling. Remember that your personal dictionary is limited to 256 words (for 130XE owners the limit is 628 words). That's it for now.

Wanted - Articles for future issues of 8:16

We are very short of articles and programs for future issues. Please help us to help you by submitting any articles on any subject you may be interested in. Subscription increased by one for each of your articles published.

Are you wasting your valuable time waiting for your printer to catch up?

Frontier's Printer-Q is the answer



Frontier Software's Printer-Q Expandable Printer Buffer

If your printer has an annoying habit of holding up your computer while it slowly prints out your latest creation then Frontier's Printer-Q expandable printer buffer is the inexpensive solution that you've been waiting for.

Easy To Use

The Printer-Q is a small device which sits between your computer and printer. Each time you print a document (whether it's a single page of simple text or multiple pages with large areas of graphics) the Printer-Q accepts the document from the computer at the fastest possible speed (up to 4000 characters per second) and then feeds the printer at its quickest rate freeing up your expensive computer for another task almost immediately. You can even leave the program that just printed the document and use another totally different program. The Printer-Q doesn't mind.

The Printer-Q gives you the option to print up to eight copies of the same document - you simply print one copy from the computer and then press the repeat button on the Printer-Q for as many copies as you need and the Printer-Q takes care of the rest. While the copies are being printed the readout on the front of the Printer-Q tells you how many copies there are remaining to print.

Just Plug It In And Go

Installation is simply a matter of plugging your computer's printer cable into the back of the Printer-Q and then plugging the Printer-Q's cable into your printer. Turn on the Printer-Q and it's ready to go. There

is no need to open either your printer or your computer.

Easily Expandable

The Printer-Q is easily expandable by you with no soldering or technical knowledge whatsoever. If you find out that the memory in most other buffers is not large enough for your requirements - the computer still has to wait for the printer because the buffer fills up too quickly - then you have to buy another larger printer buffer and scrap the current model. With the Printer-Q things couldn't be easier - starting at 128K (approximately forty pages of single spaced text) the Printer-Q is expandable up to a maximum 1MB in 128K stages. Each stage is an easy operation which takes minutes and costs very little - you simply open up the case, place another memory chip in a socket and close up the Printer-Q again.

Supplied Ready To Use

The Printer-Q is supplied with everything you need - A full manual, the amount of memory that you specify, 3 foot printer lead and power supply. The manual takes you through the easy installation and memory upgrade processes and includes a trouble shooting section if you have any problems.

Unlike most other printer buffers there is no need for any special installation



Frontier Software

software therefore the Printer-Q is compatible with nearly all computers from PCs through Amigas and STs to Amstrads. Since the Printer-Q is a separate peripheral it doesn't take any memory away from your computer which means that all of your programs can still run as normal. The Printer-Q supports the full Centronics standard, making it compatible with all Centronics printers and computers with Centronics ports.

Satisfaction Guaranteed

The Printer-Q is supplied under Frontier's ten day money back guarantee, which means that if you are unsatisfied with the Printer-Q then you can return it for a full refund within ten days of purchase. The Printer-Q carries a full twelve months' guarantee. Frontier manufacture the Printer-Q in the UK therefore knowledgeable support is always only a 'phone call away.

Prices

Printer-Q with 128K	£99.99
Printer-Q with 256K	£109.99
Printer-Q with 512K	£129.99
Printer-Q with 1MB	£169.99

(All models with less than 1MB are expandable by you with no soldering and no technical knowledge to 1MB).

Please add £3.00 for postage and packing to all orders under £150.00. Prices exclude VAT. Frontier accepts payment by Visa, Access, cheque or postal order. Prices subject to change without notice. Goods subject to availability.

Turbo - Info

Turbo-Info #5: New 65C816 Instructions
(Reprinted from Atari Interface Magazine, Nov 89)

This is the fifth instalment of a multipart information article on the Turbo-816 from DataQue Software, for the Atari XL/XE computers. The previous instalments went over the basic system, memory and OS. This article will cover some of the new 65C816 instructions.

Emulation in the 65C816

The Turbo-816 can operate in either its native mode, or the 6502 emulation mode. While in the emulation mode, the processor has many of the same limitations as the 6502. Even while in the emulation mode, many new instructions are available.

Some are of little use, because of being limited to the 64K addressing range of the 6502. Many of the new instructions allow for smaller, faster code. Other instructions offer power which was previously not feasible or practical.

PEA, PEI & PER Instructions

Of particular use are the PEA, PEI, and PER instructions. These allow the programmer to push data onto the stack, without using one of the normal registers. This is extremely useful where parameters are passed between subroutines.

The PEA instruction will push a word-sized value onto the stack. This is extremely handy for parameter passing, since you can pass absolute values or addresses.

You can also pre-load return addresses onto the stack, remembering of course that you must subtract a value of one from the destination address. PEA is also invaluable for setting the Data Bank, and Direct registers, without using a register.

The PEI instruction will push the contents of two consecutive direct page registers onto the stack. For example, PEI \$22 will push the contents of direct page address \$23 onto the stack, followed by the contents of direct page address \$22. This is very useful for pushing the contents of pointers onto the stack.

The PER instruction will push a relative value onto the stack, which is the relative offset from the current program counter to the word operand. This can be used to generate relocatable subroutines.

Since the offset is pushed rather than the actual address, the location of the code will have no bearing on the resulting value (assuming that both routines are moved an equal amount). The value pushed is a signed word value.

Another pair of useful instructions are the INA, and

By Chuck Steinman (Dataque)

DEA instructions. These instructions will increment or decrement the current value in the accumulator. It is nice where you just want to change the accumulator by a small amount, ignoring any carry. These can also be coded as INC A, and DEC A by many 65816 assemblers.

Branching Out

The 6502 has a nice set of branching instructions, except it overlooks the occasion where you would like to branch without any test. The reason for wanting to do this is to generate more relocatable code. The 65816 has two versions of branch always instructions. The BRA, which is like the other branch instructions in that it has a +127/-128 range. The BRL instruction will branch always to an address within a +32767/-32768 range. Both branches are limited to the current program bank.

A unique instruction of the 65816 is the COP instruction. This is an instruction for the (yet-to-be-seen) add-on math co-processor. But not only can it be used for that, but also users may add their own math "co-processor" (or other custom routine). The instruction has a required one byte parameter, which is not directly passed. Like the BRK instruction, the COP instruction generates an interrupt but does not push the operand onto the stack. The COP also has its own interrupt vector.

TRB and TSB Instructions

There are two new instructions for bit manipulations also. The TRB (test and reset bit) and TSB (test and set bit) instructions allow you to do a read-modify-write operation between the accumulator and either a direct memory or absolute memory address. I do not know why they did not allow for a long-absolute or indexed version, which would be useful. Any bits set in the accumulator when the instruction is executed, will clear (TRB) or set (TSB) bits in the specified destination address.

Since there are quite a few more registers in the 65816 than the 6502, there had to be an improvement made in stack handling. The 6502 only had two each of the push and pull instructions. You could push or pull the accumulator (PHA/PLA), and processor status register (PHP/PLP).

The 65816 adds to these instructions to push and pull both of the index registers (PHX/PHY/PLX/PLY), the direct register (PHD/PLD), the data bank register (PHB/PLB) and the program bank register (PHK). As you may notice, you can only push the data bank register, since popping it would be somewhat difficult to apply.

Extra 16Bit Addressing

Of course, with all of the added addressing capabilities of the 65816, there must be some way to directly call routines outside the base 64K of the machine. There is a JML instruction, which is available only as an indirect jump long. I also consider the JMP ABS_LONG a JML, but the manufacturer does not agree. There is also a JSL, which will call a long subroutine, available only in an absolute addressing mode. There also must be a way to return from one of these long subroutines, which is implemented in the RTL instruction.

A few odd instructions are STP, WAI, and WDM. The STP is used in applications (primarily battery-backed ones) where you need to put the processor into the low-power sleep mode. The only way to recover from this mode is to reset the processor. The WAI instruction is similar to the STA WSYNC that is used in the Atari. This instruction will halt the processor until an interrupt occurs. Unlike the WSYNC, the processor will wait for either an IRQ or NMI interrupt.

The WDM instruction has the most potential. This instruction is the gateway into the 32bit 65832 processor, which includes a math co-processor. After four years of waiting, this CPU has not yet been seen (sound familiar?).

XBA and XCE Instructions

The last two instructions I will cover are XBA and XCE. The XBA allows the programmer to access the upper byte of the accumulator while in either the NATive or EMULation mode of the processor. It is also handy for a temporary holding place for the accumulator. The upper byte of the accumulator is swapped with the lower byte each time the instruction is executed.

The XCE exchanges the processor status carry flag with the emulation status bit. This is how you move to and from the native and emulation modes. If the carry is set before the instruction, the processor will be in the 6502 emulation mode.

On to Applications

Now that the basics of the Turbo-816 system have been covered, I feel that you are ready to see how a Turbo-Application is designed. I will be using primarily Turbo-Calc as a reference.

I will cover how you make determinations of what resources are available and how to implement dual-function routines, without sacrificing much speed and code size.

Turbo-Info #6: Determining the System Configuration

(Reprinted from Atari Interface Magazine, December 1989)

This article will present one way to set up a program to run under the Turbo-OS, yet still be compatible with the Atari XL/XE operating system. It is assumed that the reader has some knowledge of programming at the assembly language level. This instalment will concentrate on the allocation of RAM.

To make memory management a bit easier for the programmer, the Turbo-OS features several different routines to control memory allocation and de-allocation. There are no such routines in the Atari OS, other than memory is allocated by the operating system for its own use. There are memory locations to indicate the lowest and highest available RAM locations. These pointers are modified by the application program rather than the OS.

When I set forth to write the Turbo-OS, I determined that there were four types of memory that I would support. To maintain compatibility with current programs, there was a limit to what could be

```
; first some memory locations need to be defined.
basebyte dsb 1 ;memory base address
basepage dsb 1
basebank dsb 1
lastbyte dsb 1 ;memory ending address
lastpage dsb 1
lastbank dsb 1
tempvar1 dsb 1 ;temporaries
tempvar2 dsb 1
tempvar3 dsb 1
expanded dsb 1 ;flag for expanded RAM
;availability

;note: this segment is fully code relocatable.
apropriate lda #$a5

cmp #$c001 ;this is a flag to indicated there
bne not_816 ;is a Turbo-8 16 installed
cmp #$c000
beq was_816

not_816 lda #$00 ;set all flag|pointer defaults
sta expanded ;no expanded RAM available
sta basebank ;don't really need to do these
sta lastbank ;two, but let's be official!
lda *memlo ;construct a pointer to the lowest
sta basebyte ;available standard RAM address
lda *memlo+1 ;as determined by the OS & DOS.
sta basepage
lda amemtop ;construct a pointer to the last
sta lastbyte ;available standard RAM address
lda *memtop+1
sta lastpage
ldy #$00
rts ;not 816, so we are all done!

was_816 lda #$03 ;scan for expanded RAM
jsr tramck ;Turbo-OS RAM check routine
lda taraml
```


done with standard memory, since current applications would not be calling my routines. The same would apply to the extended RAM, which is banked in the \$4000-\$7FFF range using the PORTB register of the PIA. The nature of the various types of RAM were discussed in an earlier AIM (8:16) issue.

There have been a few people tell me there is no need for any memory management on the Atari 8-bit computers. If an application needs RAM, it just uses it, and when the program terminates, another program can use it. Well, that presents problems when more than one program has to share memory. This happens quite often despite what you may think. If you boot a disk operating system into your computer, and then an application program, you would have two programs resident. There are many times when there are incompatibilities between application programs and particular DOS versions. DOS 2.x compatibles are usually the worst.

Having to load in DUP.SYS is a major trauma. If there is a program in memory you want to preserve, you have to have a MEM.SAV file on the disk. This allows you to save part of your program to disk, enough to load DUP.SYS into RAM. But then, if you happen to want to copy a file, you are just out of luck. You either have to corrupt your program or copy a sector at a time. This all happens, even though there probably is at least 16K of RAM being wasted under the OS ROM. RAMDisks were a fudge to help this problem, automatically protecting the resident program would have been a better solution.

Yeah, I know, all of you SpartaDOS users are grinning from ear to ear about now. Well, there are problems there also. Lets say my program wants to use the RAM under the OS, or one of the extended (XE type) banks? Can I jump in and use them? No, but not only that, I have no way to determine if that RAM is being used. There may be 4 to 16 banks of 16K just sitting there...or

```

ora taraml+1 ;was any expanded found?
beq not_816
acc 16

do_expand   lda taraml ;preset to max available

do_loop     sta tempvar1
            pha ;push amount onto stack
            acc 08
            lda *$03
            jsr talloc ;go allocate it
            acc 16
            pla ;pop base address off stack
            sta basepage
            cic
            adc tempvar1 ;add amount to get ending address
            sta lastpage
            acc 08
            cpy *$00 ;was there an allocation error?
            bpl no_error

error       acc 16 ;RAM may not be contiguous
            lda tempvat ;back off by a page, check again
            dea ;to find latgest contiguous block
            bne do_loop ;of unused RAM.
            acc 08 ;return with error status
            ldy *$ff ;RAM was not found wlo error
            rts

no-error    stz basebyte ;always begins on a page boundty
            lda *$ff
            sta lastbyte ;always ends befote a page
                    ;boundry
            sta expanded ;set flag for latet testing
            ldy *$00
            rts

```

SpartaDOS may be using part of them as a RAMdisk. But how does the application program know? It cannot, since there is no means provided to indicate what banks are available and how many banks there were to start with.

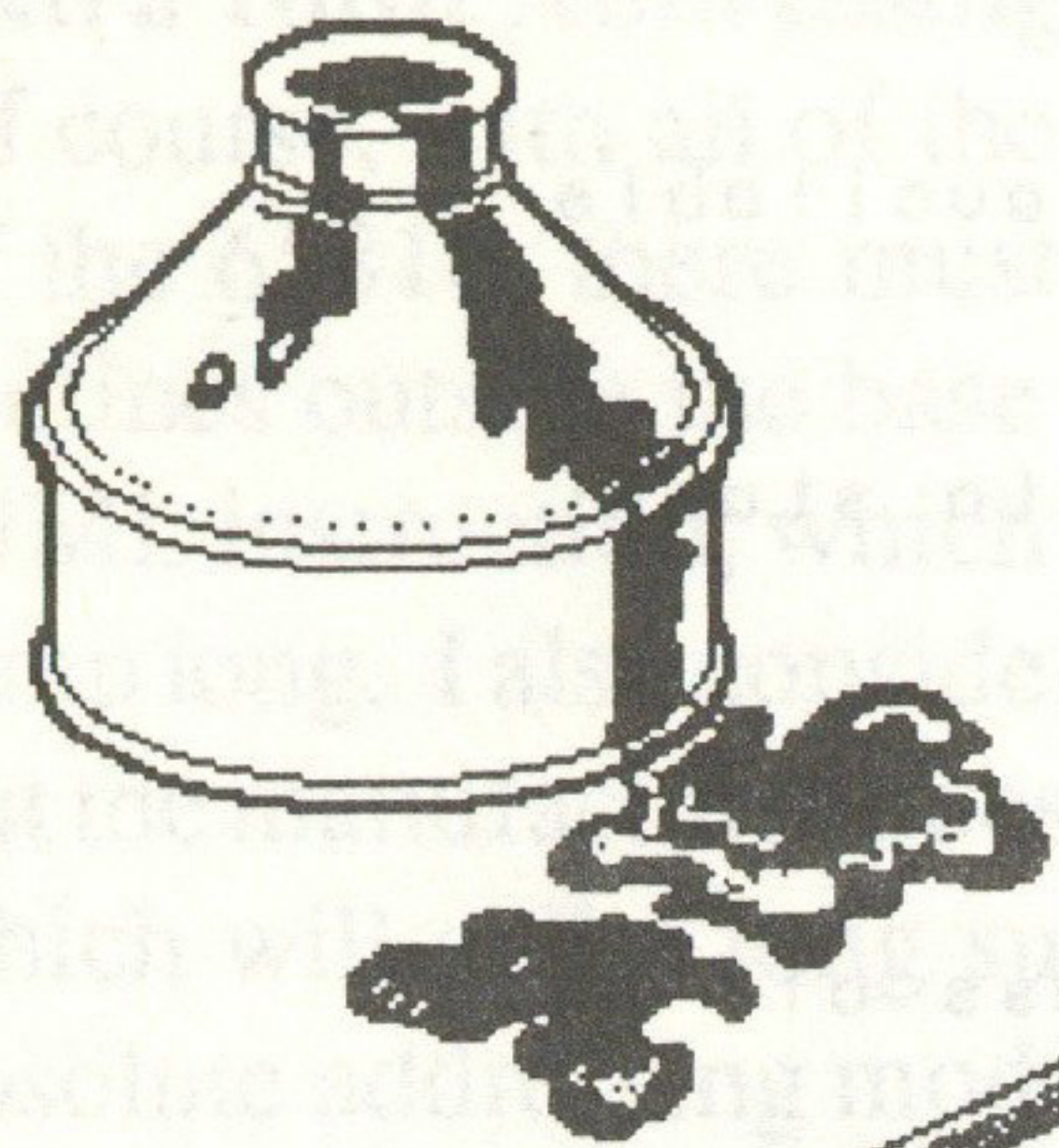
Well, I hope you can see my point. Programs should be written to be more dynamic. If the RAM under the OS is used, then look for banked RAM. If that is already used, or not available, then ask the user what he wants to do. Not just "Hey bud, can I overwrite your valuable data?" but give the guy the option to save the data, move it or destroy it. I know, I know, why do you need this type of sophistication in a game machine?

This is where the Turbo-OS

routines jump in. If you have an application that needs four banks of extended RAM, you make the Talloc call specifying you need four banks. Those four banks, if available, will be flagged as used, so that other programs will not use them. If there are not four banks available, then a bitmap of all free banks is returned. A similar procedure is followed by the extended and explicit RAM types.

Enough talk. I am sure you programmer types want to see some code fly by your tired eyes about this time. Well. I will present a code segment which will determine whether there is a Turbo-OS (and Turbo-816) installed. and what memory is available. Better buckle up, here we go!

Would you be interested in upgrading your Atari 8bit to 16bits, using the Dataque Turbo-816? If yes, let Gralin International know, as they are considering importing this product.



Dear 8:16,

Radio Atari Update

Many thanks for all your help etc, whilst I was in the UK and also for Issue 10 of 8:16. I've read the query regarding radio equipment and the Atari computer in 'Dear 8:16'. I know of a User Group in Canada who are Radio Hams and may offer advice to Brian Taylor, their address is :-

Atari Microcomputer Network
Gil Frederick (VE4AG)
130 Maureen Street
Winnipeg, MB
Canada R3K 1M2

They are an organization of Amateur Radio operators, short wave listeners and 8-bit users. They publish 'Ad Astra' Newsletter and maintain a weekly

national / international (Sunday) on-air meetings, starting at 1600z on 14.325MHz SSB-mode or on the regional nets following the National from 1830z on or about 7.235MHz. Call CQ ATARI for a contact.

I'm not a Radio Ham so that's all jibberish to me. Its the way they are listed in my Atari directory. I hope it's of help.

Kevin Le-Clercq
South Africa

Thanks for the information, I'm sure someone will understand it. Our readers may be interested to know that I recently met a chap from Poole who uses his ST for packet Radio. Hopefully we will have an informative article from him for a future issue of 8:16.

Music Studio

I wonder if you or any of your members have produced a utility to print out the music produced by "Music Studio". I purchased this from Gralin at the Stafford show and according to the manual the Commodore version has such a utility but not the Atari version. Any help would be much appreciated.

Charles Learthart
West Wickham

Sorry, we do not have such a utility, though I will continue to look out for one. Can anyone else help? Anyone fancy writing it? Also, Gralin can no longer supply Music Studio as their US supplier has no stock left and they have been unable to source an alternative.

ST Disk Magazine

We are a fairly new, disk based magazine aimed at the ST enthusiasts. The magazine is called STEN (ST Enthusiasts Newsletter) and is available free of charge to anyone who sends me a disc and SAE.

From the enclosed copy you will see that we try to cover most subjects of interest to ST owners. However as we depend on submitted articles for our content, publicity is very important.

With publicity very much in mind we are going to run a postal trivia league and would like to invite you to submit a team. There will be a 'game' every twelve days and you will 'play' another team. Both teams will have to answer and return ten trivia questions, the number of correct answers from each team will decide the points

awarded.

Every two games you will receive a printout of the league tables and the other teams scores which may be of use for inclusion in your catalogue/newsletter. All we ask is that you provide a SAE for replies.

If you are interested in participating let me know what you wish to call your team and I will return a copy of the rules and the date of the first game. All the questions will be sent out together.

Dave Mooney
14 School Road, Morningside,
Newmains, Lanarkshire

Thanks for the disk, I'm not sure if I/we have the time to participate within your game, but I'm sure some of our readers/users group might. All the best with the newsletter.

User Group Update

The Lea Valley Atari User Group is now entitled the **Cheshunt Computer Club**. All membership enquiries should now be addressed to me.

Derryck Croker
Watford

Thanks for keeping us informed. The necessary changes have been applied to the User Group section. If there any other groups out there that wish to be added please let us know.

Laserjet III Info Required

Saw your ad. in PC World and would like to make contact with serious (non-games) users of Atari, especially DTP.

My son has a 520STE upgraded to 2MB. We have just taken delivery of laser printer HP Laserjet III. We chose the machine for features, enhanced resolution and scaleable fonts.

We now find our software "Timeworks" does not permit control of resident HP Scaleable fonts, only drives in HP II mode.

We understand the Fleet Street Publisher III for Atari also, does not have driver for HP III. Software vendors

Any views on 8:16?

Any ideas for articles?

Any comments on computing in general?

Then write to:

**The Starfleet Controller, 8:16, 248 Wimborne Road,
Oakdale, Poole, Dorset BH15 3EF**

Computer Gift Vouchers

I enclose a copy of a letter sent recently to Evesham Micros together with a typed copy of their reply, both of which are explanatory (see letters below).

It occurred to me that this scheme, which I think would be useful to many computer enthusiasts, might stand a better chance of becoming reality if it were supported by others, particularly, perhaps, an established User's Group. I thought I'd let you know about it so that if you (and BaPAUG) are of the same opinion you might consider writing to Evesham Micros yourself to encourage the company to inaugurate such a scheme.

The only reason I chose to write to this particular company is that I understood them to give a generally good service.

I really do have enough pullovers and slippers (the main reason for me writing the original letter), but then, you won't understand this reference until you've read the copy letters.

Derek Warren
Fordingbridge

Dereks letter to Evesham Micros:

Dear Sir,

In common, I'm sure, with many other computer enthusiasts (I have an Atari 1040ST) my relations and friends, although well aware of my interest, have insufficient knowledge of my needs in so far as hardware or software are concerned, to attempt to purchase such

items as gift for me for Christmas, Birthdays etc. Furthermore some items are rather more expensive than would be appropriate for such gifts. What happens is that I may well receive yet another pullover or pair of slippers, of which I already have more than enough. I could get a book token, but the problem here is that there are no books on the Atari ST I require which I don't already possess or which, if needed for occasional reference, I can't get from the public library.

What a boom it would be if there were a reputable computer sales company dealing in Atari products (such as Evesham Micros) which had a simple gift voucher scheme, such as many stores these days. This would mean that the next time one of my daughters (or whoever) rings to ask my wife what I would like for (say) Christmas, she could say, always and without hesitation, "A gift voucher from Evesham Micros, whose address and telephone number are"

I would no doubt need to save up a number of such vouchers before I had enough for a "superior quality hard disk drive" as mentioned in your March '91 Atari ST User advertisement, but I'd get there in the end, even if it were with some judicious raiding of our own bank balance.

What do you think; would such a scheme be possible? Could it become a profitable venture, attracting business which, otherwise, might not be there?

It occurs to me that, on reflection, that countrywide, the profit margin of Marks and Spencer's pullover and slippers department might well be reduced to the benefit of yourselves.

Evesham Micros reply:

Dear Mr. Warren,

Thank you for your letter of 14th February, 1991.

I was gratified that you took the time to write us with your suggestion. As a premier supplier to the home computer market we are always looking for ways to improve our service to customers. Your idea certainly has possibilities, which we are currently considering.

If you have any further ideas on improving our service to you, please do not hesitate to let me know.

Best regards,
Richard Austin
Managing Director

I for one, would support such a scheme. It has been possible to purchase book tokens, record tokens and store vouchers for many years and in hindsight the computer industry appears to be lagging behind with this trend. If there is any way the BaPAUG could help support such a scheme please let us know.

My one concern would be if many different suppliers all came up with their own voucher - a unified system supported by many stores would stand a far better chance of survival.

no help. Any advice appreciated.

Bob Yates
Nottingham

I've had this letter quite a while now and can only suggest that you write to the software publishers requesting they write a suitable driver - only likely if there is enough demand. Anyone got any ideas?

Memory Upgrade

Thanks for the mention in 8:16, anyone made any comments about the RAM yet? Since then I've found an advert from Atari User, September 1986 which advertises QMEG, a 256K plug-in expansion port upgrade for £85. If only S. Fitzjohn of Stamford could be found. He claimed full compatibility with DOS2.5, SpartaDOS etc.

Brian Sheldon
Morecombe

Sorry, no information about the Yorkie board yet, Can anyone help Brian locate a QMEG?

Radio Atari Update #2

The details on Packet Radio can be found in the leaflets enclosed.

AR Knight
Poole

The leaflets enclosed were "Introduction to Packet Radio" by G3VPF and "Beginners Guide To Packet Radio" by MJ Martin (G4VRQ) both available from BARTIG, c/o Mrs Pat Beedie (GW6MOJ), "Ffynnonlas", Salem, Llandeilo, Wales, SA19 7NP. Also included was a copy of the Softville ST PD Catalogue with communications - radio & Ham radio pages highlighted - disks COMS6 and COMS11 being recommended.

The "Beginners Guide ..." looks very comprehensive and covers topics such as selection of equipment, getting started, beacons and packet bulletin board system over a total of 21 pages and is well worth the read if you are interested.

Portfolio <-> ST

I hope you can help me, at the present time I have an Atari 1040ST and a Panasonic KXP 1180 printer, which I am very pleased with.

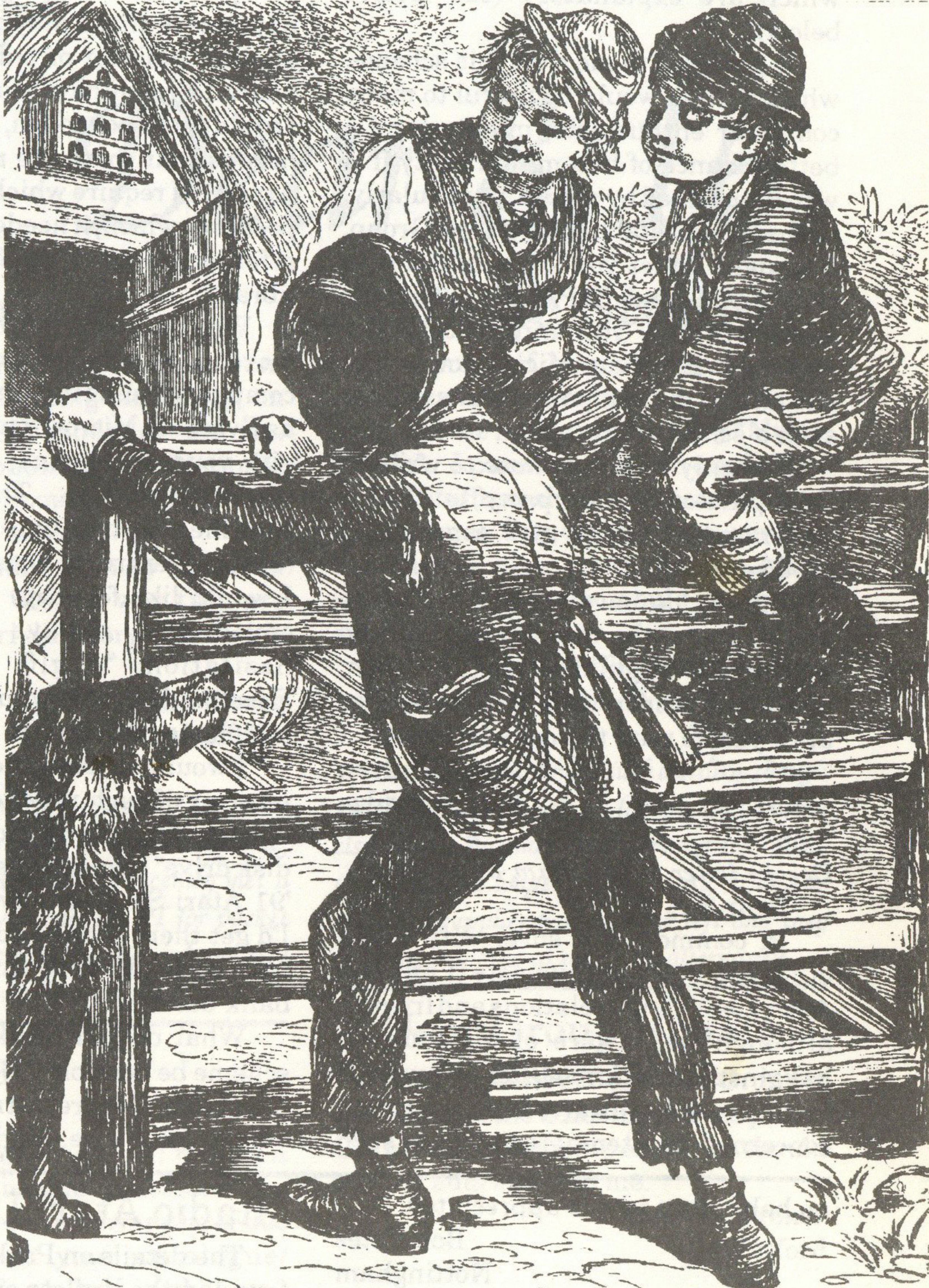
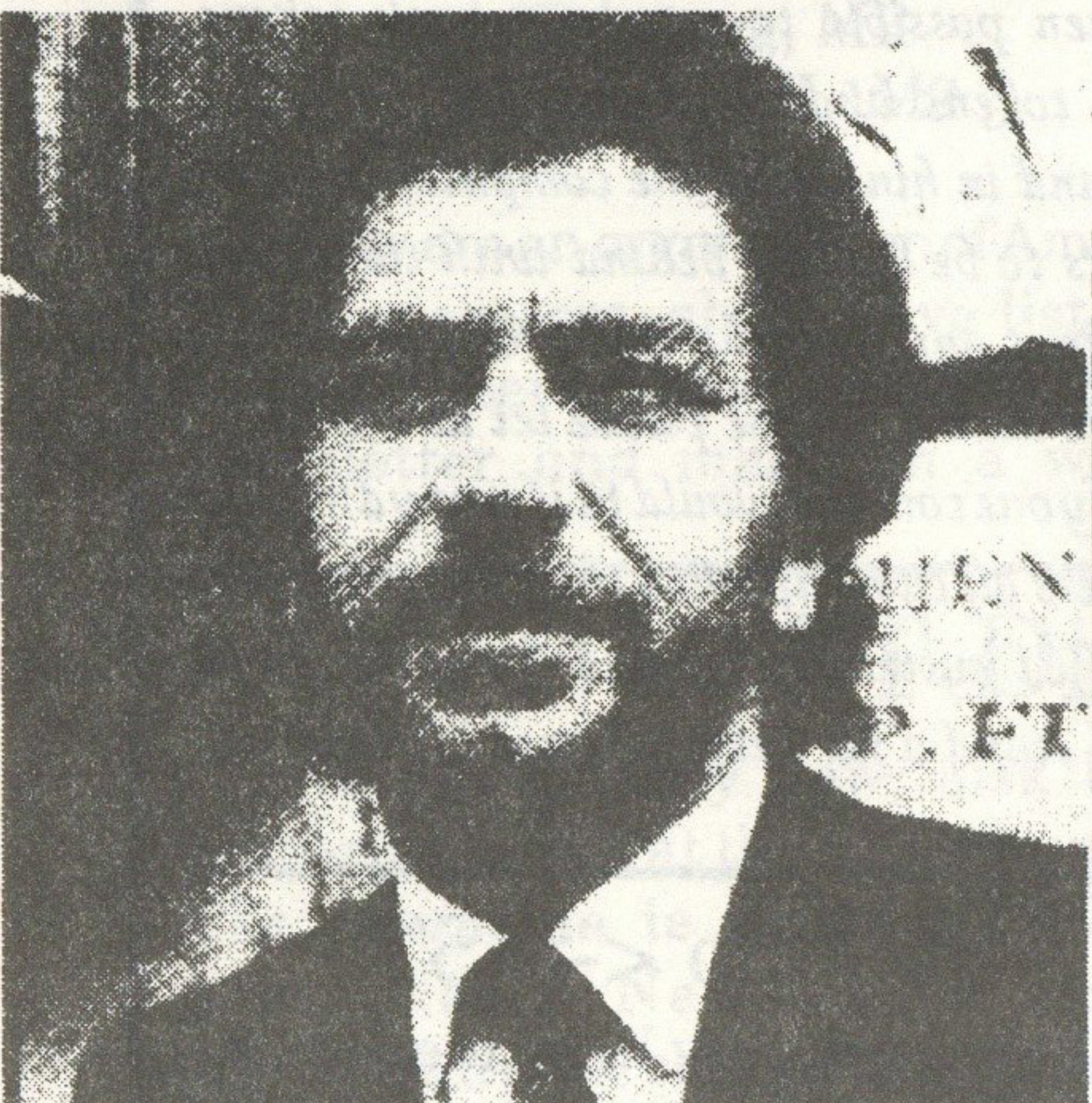
I have recently bought an Atari Portfolio with parallel interface which I have been able to use my printer with. But unfortunately I can find no way of connecting the Portfolio and ST together. Do you know of any means the ST can read the Portfolio without having to buy an expensive PC emulator.

Christopher Jones
Littlethorpe

Sorry, I currently do not know of any transfer / communications system that will help you. Try telephoning DIP at 0483 301555. Can any of our readers help?

DeskJet 500 Sample

The review and this page have been printed using the DeskJet 500 to give you an idea of just how good the results are. The picture on the right is part of a new clip-art collection I am currently compiling. More details in the next issue..



WRITE for 8:16

YOU can write for 8:16 about anything to do with Atari computers. For example you could review any new software that you may purchase so that other users may benefit from your experiences.

Mega STE News

The new Mega STE is the computer Atari need to launch a serious attack on the lower end PC market. It has a 16MHz 68000 processor with a 16k RAM cache, 68881 maths co-processor, VME expansion port, SIMM slots for easy memory expansion, a blitter chip and Appletalk port. It is housed in the same case as the TT, and is available with an internal hard disk drive. Prices are comparable to the current Mega's, making the Mega STE a very attractive machine.

CLUB NEWS

The MIDI evening was very enjoyable this year. Music Studio 88 made an appearance, as did Pro Twentyfour, Cubase and Cubeat. Demonstrations were given using some pretty impressive hardware configurations. My thanks to everyone who prepared a talk.

Development SIG

Currently this is just Neil and Myself. We are thinking about doing some articles for the magazine on basic electronics for the 8-bits and STs. Are there any suggestions for projects? Is there any interest? Write to us!

Deskjet 500 Review

By Paul Brookes

HP DeskJet 500

Hewlett Packard's DeskJet printer has just received another upgrade. Paul Brookes buys one and finds that for the money, there is very little competition...

Installation

The box arrived intact, and all of the promised contents were enclosed, along with a free printer lead supplied by the vendor. (This later proved to be a bit suspect). The documentation supplied with this printer is extensive. The installation manual is very well written, and I soon had the printer set up and performing a test print. One surprise was that the printer's power supply comes in a separate off-board unit requiring additional space for the PSU enclosure. However, the inlet (Mains) and outlet (Low Voltage) leads are generously long and should cause no problems for most set-ups.

The printer has serial and parallel interfaces located underneath the unit. The On/Off switch is mounted on the front, and access to the DIP switches is also from the front of the printer. This means that you can sandwich the printer between two other items and push it right up against the wall. It is only the front and top sides that require access. Paper is loaded from and returned to the front of the printer. The DeskJet has a built-in paper tray which

will hold approximately 100 sheets of paper. Envelopes can be printed too. The front panel selections are comprehensive but don't allow access to all of the DeskJet's internal fonts. The two cartridge slots will accept additional font families, the EPSON FX emulator and RAM packs for additional buffering or soft fonts.

HP supply a DeskJet accessory catalogue with the printer. The DeskJet Plus and 500 will also accept an IBM Proprinter III emulation cartridge. Some of the font cartridges look very tempting, but the software to drive them is rather lacking on the ST.

In Use

Now that it is nearly March, I have had time to get to know the printer.

INK - The method of squirting ink at the paper naturally means that there must be a period of time when the ink is still wet. With letter quality text printing the ink is still wet when the page is deposited in the paper out tray. However, I have only once managed to smudge the ink due to the haste in which I grabbed the paper from the machine. A few seconds wait ensures a smudge-free page.

WATER - Inkjet printed pages are prone to smudging if they come into contact with anything wet. Hewlett Packard now sell water-proof ink cartridges. Well

the cartridge might be, but the ink certainly isn't! I was recently asked about the ink problem by a gentleman at one of the club's gatherings. I had a sample page which I deliberately wet there and then, and the smudging was faint. More recently, however, I spilt a glass of water over a page and the ink ran instantly. I wasn't impressed. Mind you the page would have been ruined anyway!

CARD - I read somewhere that 200gsm thin card feeds through the DeskJet. Well I bought an assortment of thin card [150-200gsm], none of which has fed through from the paper-tray. I then tried manually loading the card as one does for envelopes, with some success. Infact after I realised that the card gets jammed if you don't nurse it into the feeder with the envelope feed buttons, everything went smoothly.

BLOTTING PAPER - The wrong type of card or paper can lead to a blotting paper effect. If you can't write on the card/paper with a fountain pen then don't print on it!

THIS SIDE UP - As I'm sure you know, paper has a 'right' side and a 'wrong' side. This is usually indicated on the packet. For example, I use Elan Executive Copier Paper [80gsm] which has the message "Important - Print on the side facing the seam of the wrapper" accompanied by a big arrow.

Deskjet 500 Review

This 'right' side should be facing down in the paper tray of the DeskJet.

NOISE - This printer is very quiet. The only annoying noise it makes is when it runs through its diagnostic tests on power-up, or after a RESET.

BUSY - Turning the DeskJet on after the ST causes the ST's printer strobe line to latch permanently low, and thus cause the DeskJet to return the BUSY status. This can be fixed by re-booting the ST. What a pain. I soon became fed up with this and decided to write an accessory to reset the strobe line without booting the ST. It works with any GEM program which allows accessories to be used. It is available directly from me for only £2.00 along with some very useful PD DeskJet Utilities (EPSON emulator, hardcopy routine etc).

GDOS - I hadn't got a driver for the DeskJet so I bought Turbojet. What a con. The latest version requires the user (me) to buy Wordup version 3.0 in order to decompress and decypher the archived file. There was no mention of this in the retailer's advert, which is not surprising because neither the packaging or the manual refer to this point. Only the README file on the disk mentions that the package only supports Wordup 3.0.

Turbojet is supposed to be a really excellent GDOS driver, but as I can't unpack the file, I

can't be sure of that. Fortunately I only use GDOS for graphics, not text, so I am able to convert the graphics into files which I can print from non-GDOS programs.

I am currently trying to contact Neocept's UK distributor to get the situation sorted out.

SPEED - I have no concerns over the speed of this printer. Standard text files print at high speed. Signa's Signum document processor, which is notoriously slow on 9-pin printers prints at a much greater rate with the DeskJet, and I presume is even faster with the DeskJet 500 as this is reported to be at least twice as fast at graphics printing than the original DeskJet. I'm told Calamus documents print very quickly when used with the DeskJet 500, second only to laser printers.

DeskJet V Laser

For most home users the DeskJet 500 offers many advantages over a laser. Firstly it is cheaper. Mine cost £430 including VAT. The Atari laser SLM605 costs nearly £900 and requires 1.5Mb of memory for its own use. Other lasers such as HP's laserjet series start from around £600 + VAT but only have 0.5Mb of memory as standard. 1.5Mb are required for most DTP uses.

Laser printers consume more power than the DeskJet 500's 25 Watts! Whilst idle the DeskJet 500 consumes just 8 Watts of

power. The old SLM804 laser consumes 400 Watts whilst idle and 700 Watts whilst printing!

Laser toners are approximately twice the price of the DeskJet injet cartridges, but last longer. However, lasers will ultimately require a drum replacement costing £150+. The ink cartridge on the DeskJet is effectively the print head, and so you get a new head every time you install a new cartridge! I have seen cartridges as low as £12 incl. for a one off purchase. Most toners are £30+.

Conclusions

Gone are the days of streaky graphics print-outs and that relentless high-pitched hammering associated with dot matrix printers. I still use my 9-pin for long program listings, mainly because it is cheaper to run! The Epson emulator cartridge is on my shopping list as a luxury item just to maintain 100% compatibility with my existing software.

Product : HP DeskDest 500

Price Guide : £430 to £500

The DeskJet Disk is available for £2 from:

Paul Brookes
32 Dudsbury Road,
West Parley,
Wimborne,
Dorset. BH22 8RE.

User Group File

Local Groups (see map below)

- Name : **Atari User Group Of Ireland**
Contact : Mike Casey
3 St. Kevins Park, Kilmacud, Co. Dublin
Notes : XL-ST-Others, Meetings, Newsletter, PD
- Name : **Bloxwich Computer Club**
Contact : Edward Hunt - 0922 409291
29 Station Street, Bloxwich, Walsall, WS3 2PD
Notes : ST-Others, Meetings, PD
- Name : **Bournemouth & Poole Atari User Group (BaPAUG)**
Contact : Ian Brooker; 163, Verity Crescent, Canford Heath, Poole, Dorset, BH17 7TX
Meetings : 1st Friday every month at the Kinson Community Centre, Pelhams, Millhams Lane, Kinson.
Newsletter : 8:16 (You're reading it)
- Name : **Cheshunt Computer Club**
Contact : Derryck Croker - 0923 673719
196 Coates Way, Garston, Watford, Herts. WD2 6PE
Notes : ST-Others, Meetings
- Name : **Mid-Cornwall Co-Op Computer Club**
Contact : Mike Richards - 0726 890473
8 Victoria Road, Roche, St. Austell, Cornwall PL26 8JF
Notes : ST-Others, Meetings
- Name : **Norwich User Group**
Contact : Ken Ward - 0603 661149
45 Coleburn Road, Lakenham, Norwich NR1 2NZ
Meetings : 1st Sunday every month. Contact Ken for time & place.
- Name : **South West ST User Group**
Contact : David Every
5 Turbill Gardens, Chaddlewood, Plympton, Plymouth, Devon, PL7 3XF
Notes : XL-ST, Meetings, Newsletter, BBS, PD
- Name : **Swindon Computer Club**
Contact : Mike Bird - 0793 539105
46 Eastcott Road, Swindon, Wilts. SN1 3LR
Notes : XL-ST-Others, Meetings, PD
- Name : **The Friday Club**
Contact : Nicholas Bavington (0908) 612272
8 Byron Drive, Newport Pagnel, Bucks. MK16 8DX
Meetings: Every Friday at Ousedale School Physics Dept. OR a members house.
Notes : XL-ST, Hardware & Software development.
- Name : **Wigan Computer Club**
Contact : Alan Owen - 0942 212662
1 Lidgate Close, Wigan, Lancs. WN3 6HA
Notes : ST-Others, Meetings, Newsletter, PD
- Name : **XL/XE Alive**
Contact : Bill Sutton - 0784 255894
13 St. Annes Avenue, Stanwell, Middlesex TW19 7RL
Notes : 8-bit

Notice To All User Groups

If you run or belong to a user group that supports any of the Atari range of products and wish your group to be listed please forward details to the BaPAUG and AAUG.

Special Interest Groups

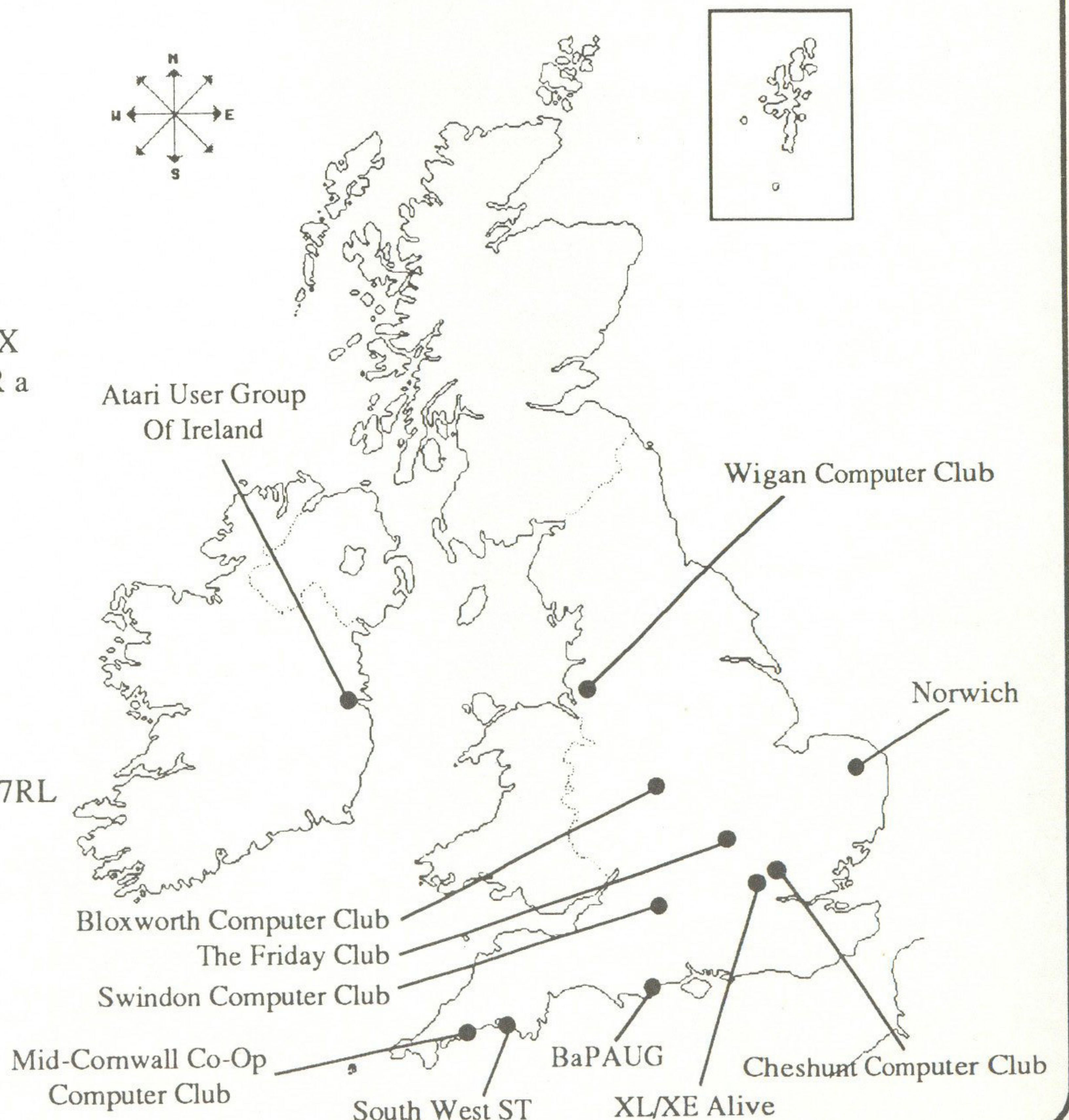
- Name : **GFA User Magazine**
Address : 186 Holland Street, Crewe, Cheshire CW1 3SJ
Telephone : 0270 - 256429

National Groups

- Name : **Association of Atari User Groups**
Address : 45 Coleburn Road, Lakenham, Norwich NR1 2NZ
Telephone : 0603 - 661149

International Groups

- Name : **Club Cenacle**
Address : B.P. 49, 95110 Sannois, France
- Name : **Genesee Atari Group**
Address : PO Box E, Flint, MI 48507, USA
- Name : **Johannesburg Atari Computer Enthusiasts (JACE)**
Address : 2 Whitehall Street, Hurst Hill, Johannesburg, South Africa, 2092
- Name : **Maryland Atari Computer Club (MACC)**
Address : 8591 Wheatfield Way, Ellicott City, Maryland 21043, USA
Telephone: (301) 461-7556
Newsletter: M.A.C.C. News
- Name : **North East Atari Team (NEAT)**
Address : P.O. Box 18150 - 0150, Phila., Pa. 19116, USA
Newsletter: The Atarian
- Name : **Northern Virginia's Atari Computer User Group**
Address : 8612 Thames St., Springfield, VA 22151, USA
- Name : **Pittsburgh Atari Computer Enthusiasts (PACE)**
Address : P.O. Box 13435, Pittsburgh, PA 15243, USA



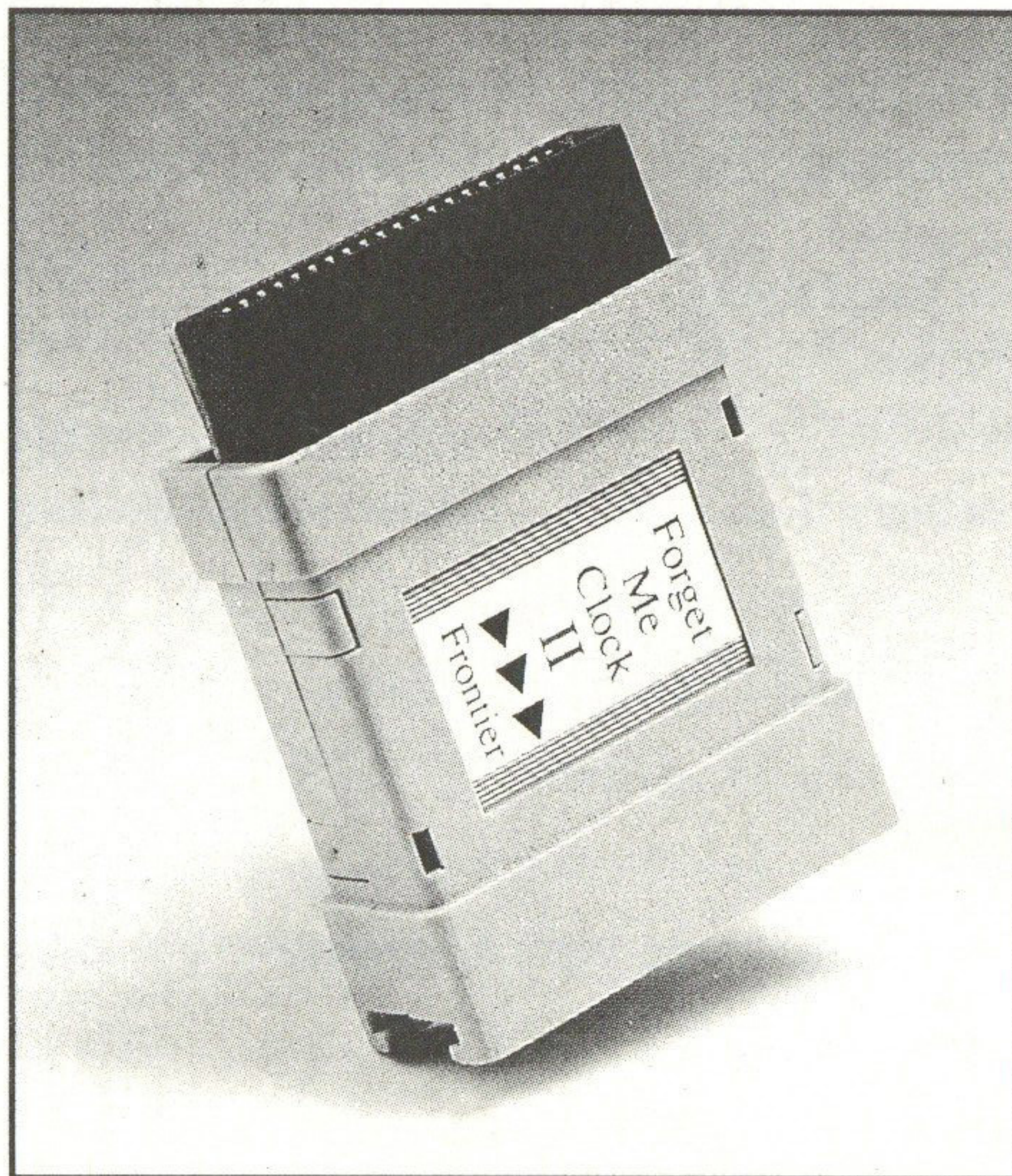
Are you wasting your valuable time setting your ST's clock?

Frontier's Forget-Me-Clock II is the answer

If you're one of those people who has an auto-run program annoyingly prompting you for the time and date every time you turn on your ST or ST^E or even worse, if you're one of those ST or ST^E users who doesn't set the system clock then Frontier's Forget-Me-Clock II cartridge is the solution you've been hoping for.

With the Forget-Me-Clock II plugged into your ST or ST^E's cartridge port the system clock (used by the Control Panel) and keyboard clock will automatically be set at turn on or reset by a small auto-run program supplied with the Forget-Me-Clock II. No longer will you have to waste your time setting your ST's clock. Using the Forget-Me-Clock II also means that files saved on disk are always properly time and date stamped making finding the latest version of a file the simple task of just looking for the file with the latest time and date.

Frontier's Forget-Me-Clock II is a clock cartridge unlike any other. All other clock cartridges for the ST will tie up the cartridge port making it impossible to have your ST's system clock set automatically while still using another cartridge. The Forget-Me-Clock II has a full cartridge pass through which means that any other cartridge for the ST can be plugged into it while it is plugged into your ST. The Forget-Me-Clock II remains totally invisible so



Frontier's Forget-Me-Clock II Cartridge With Pass Through

that the other cartridge can be used normally, but it still automatically sets the system and keyboard clocks in your ST.

Installing some clock cards for the ST means that you have to open your ST's case and pry computer chips out of their sockets. The Forget-Me-Clock II is a cartridge which plugs into the cartridge port on the side of your ST which means that its installation couldn't be simpler - you just plug it in and turn on your ST.

Every Forget-Me-Clock II cartridge is supplied with time and date setting software for the Forget-Me-Clock II's clock together with a small auto-run

program which automatically sets your ST's system and keyboard clocks every time you turn on or reset your ST. Built into the setting software is the facility to stop the Forget-Me-Clock II's clock to save on battery life when the Forget-Me-Clock II is not being used.

The Forget-Me-Clock II is supplied under Frontier's ten day money back guarantee, which means that if you don't like the Forget-Me-Clock II for any reason, you can return it for a full refund within ten days of purchase. The Forget-Me-Clock II has been designed to work with any model of ST whether it be ST, STM, STF, STFM, ST^E or Mega ST.

The Forget-Me-Clock II carries a full two year guarantee which includes the battery. Battery life has been tested to be many times the guarantee period. Frontier will supply replacement batteries outside of the guarantee period for a small charge.

Forget-Me-Clock II Cartridge £24.99

Price includes VAT. Please add £1.15 for postage and packing to all orders under £50.00. Frontier accepts payment by Visa or Access. Price subject to change without notice. Goods subject to availability.



Frontier Software