NEW!

ZX COMPUTING

Summer 1982
£1.75

Britain's Biggest Magazine For The Sinclair User

Over 100 pages of information and programs for the ZX81 and ZX80 user including a 1K Chess routine!

PLUS

- Software Reviews — which programs are the best buy?
- Business Routines — put your ZX to work!
- Expansion Systems — how good and how much?
- Machine Code for ZX-81 — secrets revealed at last!
- DIY Memory Upgrade — cheap way of adding bytes

BUG-BYTE

ZX81 SOFTWARE

BUG-BYTE UTILITIES

ZXAS £5.00

This full specification Z80 Assembler assembles all the standard Zilog mnemonics, which are simply written into REM statements within your BASIC program.When assembled, the assembly listings, together with assembled codes and addresses are displayed on the screen. The assembled codes and addresses, are displayed on the screen. The assembled code is executed by USR. The program occupies 5K. This means that ZXAS maybe used in conjunction with ZXDB. The program is available for both the 16K ZX81 and the 8K ROM ZX80. Full documentation on how to use the assembler (including a list of the mnemonics) is supplied.

ZXDB £6.50

The perfect complement to the ZXAS assembler, ZXDB is a complete combined machine code disassembler and debugging program.
Apart from the DISASSEMBLER, the program has features inc including SINGLE STEP, BLOCK SEARCH, TRANSFER AND FILL, HEX LOADER, REGISTER DISPLAY and more, all of which are executed by simple one key commands from the keyboard.

ZXTK £6.00

ZXTK enhances the facilities offered by ZX81 BASIC - a must for the serious BASIC programmer. Includes full re-numbering, the ability to manipulate groups of lines and merge files, read filenames from tapes and more.

RENUM £4.00

Renum is a full renumbering program. Renumbers line numbers, GOTOs and GOSUBs, and will renumber existing programs.

BUG-BYTE ADVENTURES

The Damsel and the Beast
A game of concentration and suspence in which you, the intrepid hero, must wander in the darkness and dangers of the Beast's palace, find the Damsel hiding or imprisoned there, kill the Beast, and then last but not least, lead the Damsel to the palace exit before she starves to death. An extremely complex, frustrating and entertaining game.
Price £6.50

Dictator
You have just become the 129th president for life of the state of Ritimba. The object of your rule is to do what all previous presidents failed to do, that is, to take full advantage of the situation for your own good.
The program is supplied on high quality cassette, together with an 8 page booklet containing detailed descriptions, instructions and hints.
Price £9.00

Star Trek
The classic computer game in which you trek across the galaxy in search of Klingons to zap with your phasers and photon torpedoes. You have long and short range scanners to help you find them, Starbases to refuel your ship at and, of course, various witty comments from the crew.
Price £6.50

House of Gnomes
Another fantastic adventure game for the 16K ZX81.
Price £7.00

INVADERS

Fast moving, machine code version of the famous arcade game, for the 16K ZX81. Shields are provided to help protect you from the bombs of the marching aliens. Ten levels of play from easy to suicidal. On screen scoring.
Price only £4.00

BUG-BYTE GAMES PACKS

Program Pack	1	£3.50
Program Pack	2	£3.50
Program Pack	3	£4.50
Program Pack	4	£4.50
Program Pack	5	£4.50
Program Pack	6	£4.50
Program Pack	7	£5.00
Program Pack	8	£6.00

OTHERS

Multifile
A multipurpose filing system for the 16K ZX81. A wide range of possible applications.
Price £17.50

Videograph
Graphical display and testing program. Hundreds of possible applications.
Price £7.50

Viewtext
Paged information system for the 16K ZX81. Great for displays
Price £7.00

Constellation
Turn your ZX81 into a telescope. Gives an accurate representation of the night sky on any date of the Century.
Price £8.00

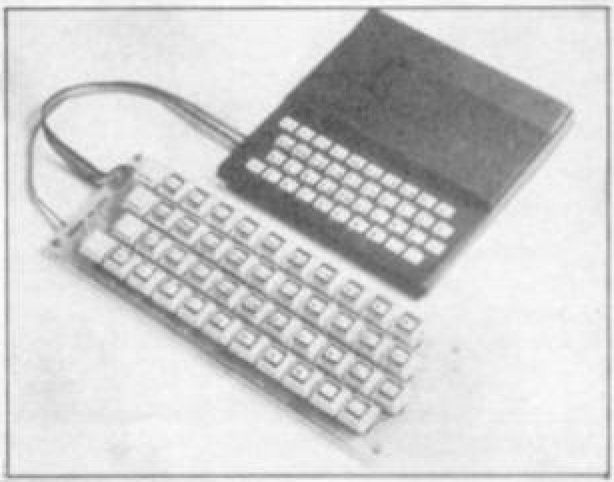
All prices inclusive.

VISA

BUG-BYTE
Microcomputer Software
98-100 THE ALBANY
OLD HALL STREET
LIVERPOOL L3 9EP

ZX 80/81 HARDWARE/SOFTWARE

ZX KEYBOARD



Now with repeat key and facilities to add numaric pad. The keyboard has all the 80/81 functions on the keys, and will greatly increase your programming speed. It is fitted with push type keys as in larger computers.

The keyboard has been specially designed for the Sinclair computer and is supplied ready-built. It also has facilities for 3 extra buttons which could be used for on/off switch, reset, etc. **£27.95.** Numaric add on **£10.**

4K GRAPHICS ROM

The dK Graphic module is our latest ZX81 accessory. This module, unlike most other accessories fits neatly inside your computer under the keyboard. The module comes ready built, fully tested and complete with a 4K graphic ROM. This will give you 448 extra pre-programmed graphics, your normal graphic set contains 64. This means that you now have 512 graphics and with there inverse 1024. This now turns the 81 into a very powerful computer, with a graphic set rarely found on larger more expensive machines. In the ROM are lower case letters, bombs, bullets, rockets, tanks, a complete set of invaders graphics and that only accounts for about 50 of them, there are still about 400 left (that may give you an idea as to the scope of the new ROM). However, the module does not finish there; it also has a spare holder on the board which will accept a further 4K of ROM/RAM. This holder is fitted with a 1K/2K RAM and can be used for user defineable graphics so you can create your own custom charactor sets. **£29.95.**

MEMORY 80/ 81

16K RAM
Massive add-on memory for 80/81. **£32.95.**

2K & 4K RAM
Static Ram memory expansion for the 80/81. They both work with onboard Ram i.e. 4K plus onboard = 5K. This is the cheapest small memory expansion available anywhere. 2K RAM **£14.95.** 4K RAM **£19.95.**

IO PORT
Interface your computer with the real world, our IO port has holders to add up to 6K of RAM. **£12.95.**

16K 81 SOFTWARE

As seen at the ZX Microfair.

DEFLEX This totally new and very addictive game, which was highly acclaimed at the Microfair, uses fast moving graphics to provide a challenge requiring not only quick reaction, but also clever thinking. One and two player versions on same cassette. £3.95

3D/3D LABYRINTH You have all seen 3D Labyrinth games, but this goes one stage beyond; you must manoeuvre within a cubic maze and contend with corridors which may go left/right/up/down. Full size 3D graphical representation. £3.95.

CENTIPEDE. This is the first implementation of the popular arcade game on any micro anywhere. Never mind your invaders, etc., this is positively stunning, the speed at which this runs makes ZX invaders look like a game of simple snap. £4.95.

Graphic ROM Software Centipede. The graphic ROM version of our popular Centipede game. The only real version of Space Invaders on the ZX81.

dK'tronics

23 Sussex Road, Gorleston,
Great Yarmouth, Norfolk.
Telephone: Yarmouth (0493) 602453
Dept. ZXC582

ZX CO

ZX Computing
Vol. One
Number One
Summer 1982

Editor: Tim Hartnell
Editorial Assistant: Tina Boylan
Group Advertising Manager: Ron Moore
Managing Editor: Ron Harris.
Managing Director: T J Connell.

Origination and design by MM Design & Print, London

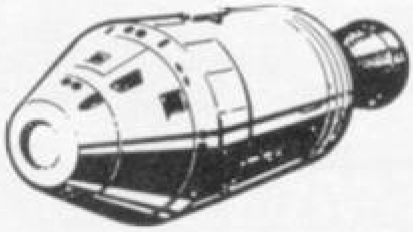
Published by Argus Specialist Publications Ltd, 145 Charing Cross Road, London WC2H 0EE.

CONTENTS

ZX Computing is constantly on the look-out for well written articles and programs. If you think that your efforts meet our standards please feel free to submit your work to us for consideration.

Material should be typed if possible. Any programs submitted must be listed, cassette tapes alone will not be accepted, and should be accompanied by documentation to explain how they work and make it easy to run them. All submissions will be acknowledged. Any published work will be paid for.

All work for consideration should be sent to the Editor at our Charing Cross Road address.



MPUTING

ZX Computing is published quarterly in April, July, October and January. Distributed by: Argus Press Sales & Distribution Ltd, 12-18 Paul Street, London EC2A 4JS. 01-247 8233. Printed by: Henry Garnett Ltd., Rotherham.

Sinclair ZX81 Personal Comp
the heart of a system that grows with you.

1980 saw a genuine breakthrough – the Sinclair ZX80, world's first complete personal computer for under £100. Not surprisingly, over 50,000 were sold.

In March 1981, the Sinclair lead increased dramatically. For just £69.95 the Sinclair ZX81 offers even more advanced facilities at an even lower price. Initially, even we were surprised by the demand – over 50,000 in the first 3 months!

Today, the Sinclair ZX81 is the heart of a computer system. You can add 16-times more memory with the ZX RAM pack. The ZX Printer offers an unbeatable combination of performance and price. And the ZX Software library is growing every day.

Lower price: higher capability
With the ZX81, it's still very simple to teach yourself computing, but the ZX81 packs even greater working capability than the ZX80.

It uses the same micro-processor, but incorporates a new, more powerful 8K BASIC ROM – the 'trained intelligence' of the computer. This chip works in decimals, handles logs and trig, allows you to plot graphs, and builds up animated displays.

And the ZX81 incorporates other operation refinements – the facility to load and save named programs on cassette, for example, *and* to drive the new ZX Printer.

New BASIC manual

Every ZX81 comes with a comprehensive, specially-written manual – a complete course in BASIC programming, from first principles to complex programs.

Kit: £49.95

Higher specification, lower price – how's it done?
Quite simply, by design. The ZX80 reduced the chips in a working computer from 40 or so, to 21. The ZX81 reduces the 21 to 4!

The secret lies in a totally new master chip. Designed by Sinclair and custom-built in Britain, this unique chip replaces 18 chips from the ZX80!

New, improved specification

● Z80A micro-processor – new faster version of the famous Z80 chip, widely recognised as the best ever made.

● Unique 'one-touch' key word entry: the ZX81 eliminates a great deal of tiresome typing. Key words (RUN, LIST, PRINT, etc.) have their own single-key entry.

● Unique syntax-check and report codes identify programming errors immediately.

● Full range of mathematical and scientific functions accurate to eight decimal places.

● Graph-drawing and animated-display facilities.

● Multi-dimensional string and numerical arrays.

● Up to 26 FOR/NEXT loops.

● Randomise function – useful for games as well as serious applications.

● Cassette LOAD and SAVE with named programs.

● 1K-byte RAM expandable to 16K bytes with Sinclair RAM pack.

● Able to drive the new Sinclair printer.

● Advanced 4-chip design: micro-processor, ROM, RAM, plus master chip – unique, custom-built chip replacing 18 ZX80 chips.

Built: £69.95

Kit or built – it's up to you!
You'll be surprised how easy the ZX81 kit is to build: just four chips to assemble (plus, of course the other discrete components) – a few hours' work with a fine-tipped soldering iron. And you may already have a suitable mains adaptor – 600 mA at 9 V DC nominal unregulated (supplied with built version).

Kit and built versions come complete with all leads to connect to your TV (colour or black and white) and cassette recorder.

1(
p
a

De
fit
RA
ex
of
da

pr
Ye
of

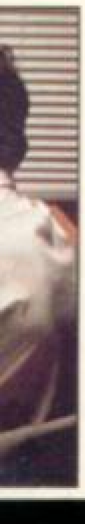
al
ca
Hc
fo

6 K
Te

mpter-

s to
her
ours'
g iron.
table
DC
with

com-
o
ite)



Available now- the ZX Printer for only £49.95

Designed exclusively for use with the ZX81 (and ZX80 with 8K BASIC ROM), the printer offers full alphanumerics *and* highly sophisticated graphics.

A special feature is COPY, which prints out exactly what is on the whole TV screen without the need for further intructions.

At last you can have a hard copy of your program listings – particularly useful when writing or editing programs.

And of course you can print out your results for permanent records or sending to a friend.

Printing speed is 50 characters per second, with 32 characters per line and 9 lines per vertical inch.

The ZX Printer connects to the rear of your computer – using a stackable connector so you *can* plug in a RAM pack as well. A roll of paper (65 ft long x 4 in wide) is supplied, along with full instructions.

16K-byte RAM pack for massive add-on memory.

Designed as a complete module to fit your Sinclair ZX80 or ZX81, the RAM pack simply plugs into the existing expansion port at the rear of the computer to multiply your data/program storage by 16!

Use it for long and complex programs or as a personal database. Yet it costs as little as half the price of competitive additional memory.

With the RAM pack, you can also run some of the more sophisticated ZX Software – the Business & Household management systems for example.

How to order your ZX81
BY PHONE – Access, Barclaycard or Trustcard holders can call 01-200 0200 for personal attention 24 hours a day, every day.
BY FREEPOST – use the no-stamp-needed coupon below. You can pay by cheque, postal order, Access, Barclaycard or Trustcard.
EITHER WAY – please allow up to 28 days for delivery. And there's a 14-day money-back option. We want you to be satisfied beyond doubt – and we have no doubt that you will be.

To: Sinclair Research, FREEPOST, Camberley, Surrey, GU15 3BR. **Order**

Qty	Item	Code	Item price £	Total £
	Sinclair ZX81 Personal Computer kit(s). Price includes ZX81 BASIC manual, excludes mains adaptor.	12	49.95	
	Ready-assembled Sinclair ZX81 Personal Computer(s). Price includes ZX81 BASIC manual and mains adaptor.	11	69.95	
	Mains Adaptor(s) (600 mA at 9 V DC nominal unregulated).	10	8.95	
	16K-BYTE RAM pack.	18	49.95	
	Sinclair ZX Printer.	27	49.95	
	8K BASIC ROM to fit ZX80.	17	19.95	
	Post and Packing.			2.95

□ Please tick if you require a VAT receipt TOTAL £ ______

*I enclose a cheque/postal order payable to Sinclair Research Ltd, for £ ______

*Please charge to my Access/Barclaycard/Trustcard account no.

*Please delete/complete as applicable. | | | | | | | | | | | | | | | | | |

Please print

Name: Mr/Mrs/Miss | | | | | | | | | | | | | | | | | | | |

Address: | | | | | | | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | | | | | | | | |

FREEPOST – no stamp needed. ZXC 05

sinclair
ZX81

6 Kings Parade, Cambridge, Cambs., CB2 1SN.
Tel: (0276) 66104 & 21282.

Welcome

What we're trying to do for you . . . and how you can help us.

Welcome to the first bumper issue of *ZX Computing*. The ZX81 and the ZX80 are great computers, and we're going to make sure you make the most of your Sinclair micro.

As you'll see by looking through this magazine, we've got a host of programs for you to try, to demonstrate how flexible your micro can be. There are a lot of games, because most of us like playing games, but there are many other programs as well to show you, for example, how you can use your ZX computer for business.

If you don't yet understand machine code, but you want to learn about it, we have two articles in this issue to help you fathom out the mystery. Both of the articles contain sample routines for you to enter and run, so you can see your machine code in action immediately.

If you're tired of trying to make your programs fit into 1K, and don't want to spend too much expanding your computer's memory, this issue's construction article will give you additional bytes very cheaply and simply.

The number of ZX81 books on the market is bewildering. How can you decide which is the best book for you at this stage in your programming development? In this issue of *ZX Computing*, we take a trip through the ZX library, giving you an honest assessment of some of the books you can buy.

The same goes for software. It seems that every cottage in the UK hides an industry producing ZX software. How good is it? And is it worth the money? Again our reviewers look fearlessly at a selection of the software packs on the market . . . and reach some surprising conclusions.

At *ZX Computing* we're committed to producing a magazine which will be of genuine assistance to you to ensure you make the most out of your computer, whether you want it to help you develop your programming skills or learn machine code, play games or use it in business. But we need you to help us. The only way we can keep up the high standard we've set in this first issue is for you to send us your best programs, routines, construction projects and discoveries. We'll pay for any material we use. If you have any opinions on software or hardware you've bought for your ZX81 or ZX80, or on Sinclair's delivery, repair or back-up service, please share them with us, so we can share them with other readers.

Tim Hartnell

The @*?$!!&£@ RAM pack

Dear ZX Computing,
I've read many letters which mirror my own problems with Sinclair's 16K memory pack.

I bought a 16K RAM pack, and — like so many correspondents — have suffered with bytes of information altering themselves before the computer crashed. When I first encountered the problem, I sent my 16K pack back as I thought it was faulty. It was returned with a note to the effect that there was nothing wrong with it.

I have now relieved the problem of disappearing data but at the expense of my ZX80. I made a number of 'alterations', the first of which was to abandon the flimsy top part of the casing. I presumed that the fault was a heat problem and hoped this would cure it. It partly worked and so I then unbolted the regulator and heatsink from the board and left it standing in free air with an additional heatsink. This helped greatly, but did not totally solve the problem.

The final 'modification' was to remove the casing around the 16K RAM pack. There are 15 chips inside the case which get quite hot. I now have no problems with data altering — except under one condition. When working on a large program (3K plus), and extensively altering the program, I find there comes a point where the computer cannot handle the extreme alterations.

K Huber,
Hayes End, Middlesex.

Catty wonder

Dear ZX Computing,
The wonders of C.C's little machine never ceases to amaze me. Did you know that this program actually works:

```
10  LET C = 1.5
20  LET A = 4.9
30  LET T = 20
40  LET CAT = 49
50  LET CA = 12
60  INPUT X$
70  PRINT VAL X$
80  PRINT VAL X$ (TO 2)
```

Entering either C, CA or CAT produces the appropriate answer on line 70, and will even break down CAT to CA on line 80 to print 12 as the answer.

Andrew Field,
Epping, Essex.

And an answer

Dear ZX Computing,
Having read about the various problems with the 16K RAM pack, I am beginning to wonder if I am exceptionally lucky, or are all the other satisfied ZX80/81 users too busy writing long and complex programs to find the time to write?

I did have some crashes with the 3K RAM, so I wrote to Sinclair and received a letter which included the following: "We have found that lubrication on the contacts solves the problems of RAM pack connections. You should first clean the edge connector with surgical spirit and then smear it with Vaseline."

I did this and had no further trouble, and when my 16K RAM arrived I repeated the treatment. Again, no trouble at all.

With regard to Sinclair service, I can only say that I am very happy. Like many others, I received a letter about the 16K RAM delivery. Mine arrived in four weeks. I had a keyboard fault on the ZX81 kit that I assembled for my son. It was replaced by *return post* after phoning Cambridge. We had another intermittent fault, and as I was able to go to Cambridge, I took it to the Sinclair office. I was directed to the Service Department where, after spending some time on it and failing to find the offending component, they replaced it with an assembled ZX81.

Note that I have no connection with Sinclair, except as a satisfied user.

D E F Rolfe,
Haslemere, Surrey.

Screen invert

Dear ZX Computing,
When writing games for my ZX81, I find that I sometimes need to 'inverse' everything on the screen at a particular stage in the program. It is possible to use a BASIC subroutine to do this but this is a slow process — half a minute or so — so I set about writing a machine code routine which could be used in the SLOW mode.

Type in the following program:

```
10  POKE 16388, 0
12  POKE 16389, 127
14  FOR C=32600 TO 32624
16  INPUT N
18  POKE C, N
20  NEXT C
22  NEW
```

The numbers to be entered (line 16) are as follows:

42, 14, 64, 6, 22, 126, 254, 118, 32, 8, 5, 120, 254, 0, 32, 5, 24, 6, 198, 128, 119, 35, 24, 237, 201

The routine will still remain at the top of RAM, even after NEW, so it can be used for any program LOADed during a programming session. You need to include the following routine in the BASIC program:

```
9000  PRINT AT 0,0;
9010  LET RR=USR 32600
```

Line 9000 ensures that the m/c routine starts scanning at the beginning of the screen. The routine takes less than an eighth of a second to RUN and is relocatable, that is, it can be placed anywhere in RAM.

John Miller,
Farlington, Portsmouth.

Och aye the '81

Dear ZX Computing,
Following an advertisement and a feature in the local paper, a ZX81 users' club has been formed in Inverclyde (Gourock, Greenock and Port Glasgow). The club is meeting weekly on Wednesdays at members' homes in turn.

Our numbers are as yet small, but they are increasing. We would be grateful if you could mention us in 'ZX Computing'. The number to phone for meeting details is Gourock 39967.

The club secretary (me) and one of our other members attended the ZX Microfair, bringing back the saga of a day in London neither will forget. Not that we would want to. We only hope it will be possible to arrange something similar further north so that all our members can have the opportunity of seeing a wide range of both hardware and software.

Robert Watt,
Gourock, Renfrewshire.

●*Eric Deeson, the organiser of the EZUG (Educational ZX Users Group) is planning a microfair in Birmingham in September. More details on that in the next issue of ZX Computing. To find out more about EZUG, send a large, SAE to EZUG, Highgate School, Birmingham B12 9DS. TH*

Memotech's New Memory System for the **ZX81**

It grows as you progress

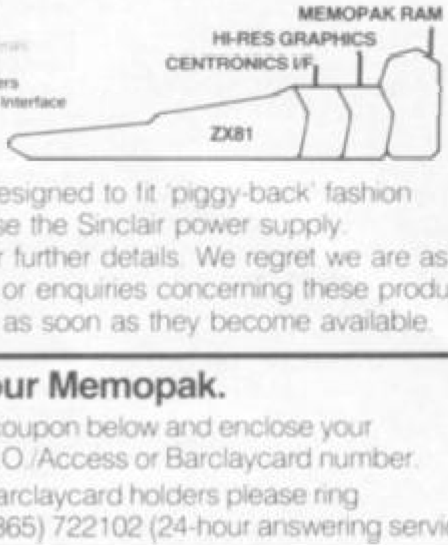
Memopak 16K Memory Extension - £39.95 incl.VAT

It is a fact that the ZX81 has revolutionised home computing, and coupled with the new Memopak 16K it gives you a massive 16K of Directly Addressable RAM, which is neither switched nor paged. With the addition of the Memopak 16K your ZX81's enlarged memory capacity will enable it to execute longer and more sophisticated programs, and to hold an extended database.

The 16K and 64K Memopaks come in attractive, custom-designed and engineered cases which fit snugly on to the back of the ZX81, giving firm, wobble-free connections. See below for ordering information.

Coming Soon...

– a complete range of ZX81 plug-in peripherals
Memotech Hi-Res Graphics
Centronics Interface and Software Drivers
Memotech Digitising Tablet RS232 Interface



All these products are designed to fit 'piggy-back' fashion on to each other, and use the Sinclair power supply. WATCH THIS SPACE for further details. We regret we are as yet unable to accept orders or enquiries concerning these products – but we'll let you know as soon as they become available.

How to order your Memopak.

By Post: Fill in the coupon below and enclose your cheque/P.O./Access or Barclaycard number.

By Phone: Access/Barclaycard holders please ring Oxford (0865) 722102 (24-hour answering service).

Memopak 64K Memory Extension -£79.00 incl.VAT

The 64K Memopak is a pack which extends the memory of the ZX81 by a further 56K, and together with the ZX81 gives a full 64K, which is neither switched nor paged, and is directly addressable. The unit is user transparent and accepts basic commands such as 10 DIM A(9000).

BREAKDOWN OF MEMORY AREAS

0–8K . . . Sinclair ROM

8–16K . . . This section of memory switches in or out in 4K blocks to leave space for memory mapping, holds its contents during cassette loads, allows communication between programmes, and can be used to run assembly language routines.

16–32K . . . This area can be used for basic programmes and assembly language routines.

32–64K . . . 32K of RAM memory for basic variables and large arrays.

With the Memopak 64K extension the ZX81 is transformed into a powerful computer, suitable for business, leisure and educational use, at a fraction of the cost of comparable systems.

Unique 3 month trade-in offer!

When your programming needs have outgrown the capacity provided by 16K RAM, and you find it necessary to further extend your ZX81's capacity, we will take back your 16K Memopak and allow a discount of £15.00 against your purchase of our 64K model.*

**We reserve the right to reject, for discounting purposes, units which have been either opened or damaged in any way.*

© R. Branton & G A C Boyd 1982

Please make cheques payable to Memotech Limited

Please debit my Access/Barclaycard* account number

____

**Please delete whichever does not apply*

SIGNATURE ________________ DATE ________

Please send me:

	Quantity	Price	Total
16K RAM, Assembled		£39.95	
64K RAM, Assembled		£79.00	
		Postage	£2.00
		Total Enclosed	

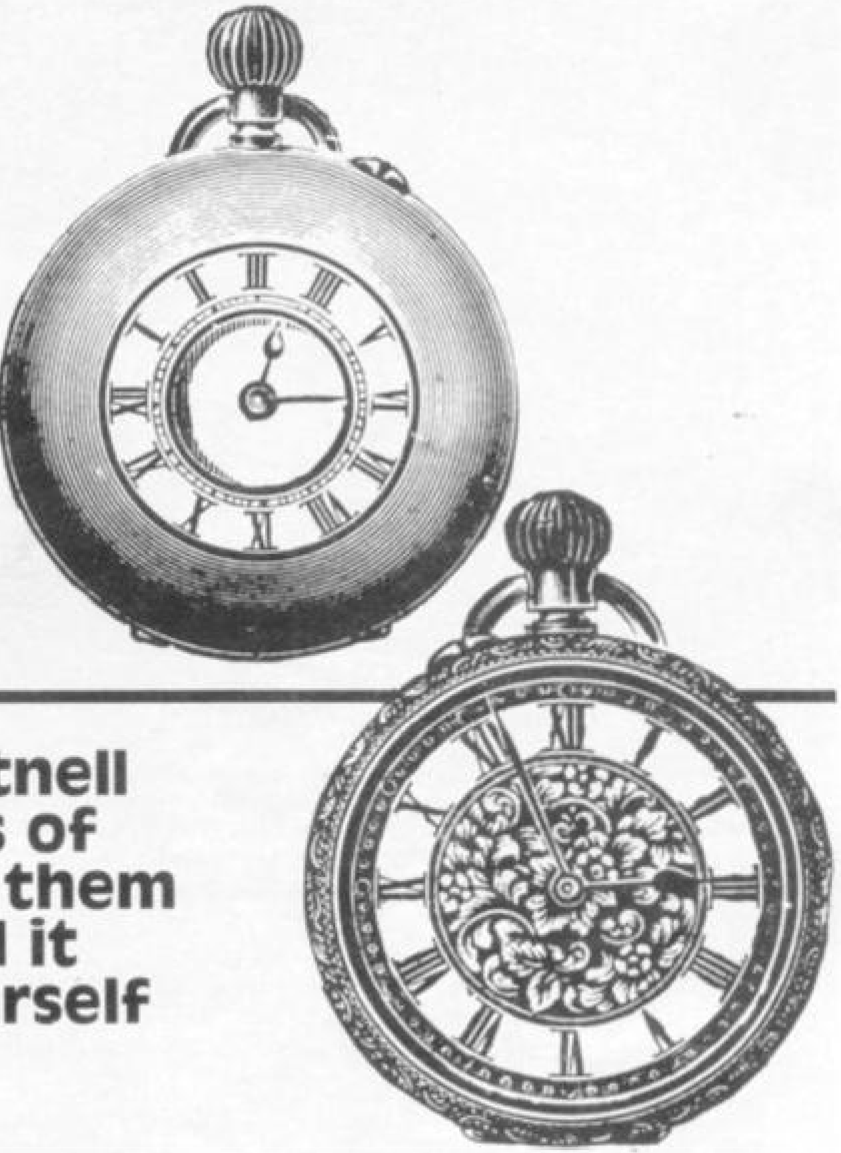
NAME ________________ ADDRESS ________________

ZX1

We want to be sure you are satisfied with your Memopak – so we offer a 14-day money back Guarantee on all our products.

Memotech Limited, 3 Collins Street, Oxford OX4 1XL, England Telephone: Oxford (0865) 722102/3/4/5

Testing your ZX81's speed



Henry Budgett and Tim Hartnell look at the 'standard' series of benchmark tests and apply them to the ZX81 to see how well it performs. Try the tests yourself and see if your machine measures up.

There are a number of ways to establish the efficiency of a computer, and the tests performed are generally called 'benchmarks'. Each set of these standard tests tries out a function or functions of the computer, and produces a measureable result which can be compared with other computers performing similar tasks.

The most commonly used tests in the microcomputer world are those introduced way back in 1977 by the Yankee magazine KILOBAUD. While they're not particularly rigorous, they do offer a quick and simple solution to the problem of checking out how well the computer performs certain arithmetic functions.

The speed of processing depends on the speed of the microprocessor's clock. In the ZX81, the BASIC runs at 3.25 MHz.

There are eight benchmarks in the series. Apart from the last one — which requires the presence of certain mathematical functions and demands floating-point numbers — you'll be able to run a version of them on the ZX80. Each test should be run and timed 10 times, then an average of the results obtained.

The first test is a simple loop program that sets up a FOR . . . NEXT loop of 1000 counts. This, and the other tests, waits until a key is pressed after pressing RUN before starting the program. You enter RUN, then press NEWLINE, and then touch any key (except BREAK) as you start your timing. Lines 5 and 10 in each program ensure that the program waits until (a) you've taken your finger off the RUN key (line 5) and then until you've pressed any key (line 10). You'll obviously have to omit these lines on the ZX80. You stop timing when the zero appears in the top left-hand corner of the screen.

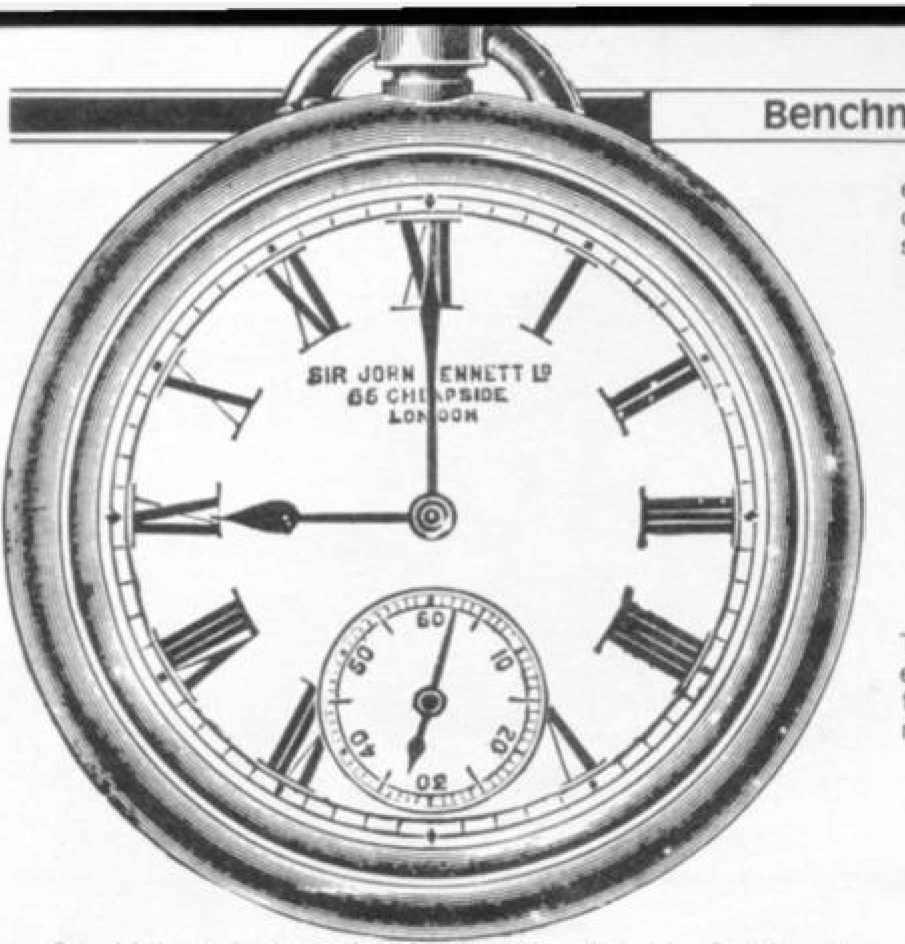
The tests are particularly fascinating if you can run them on another computer to compare the times. The ZX81 is not a very fast machine, even in FAST mode. Part of the slowness can be attributed to the fact that the ZX81 stores all numbers as if they were floating point numbers (ie as if they had digits after a decimal point), even when they are integers (whole numbers). The more bytes a variable occupies, the longer it is going to take to process the information stored, and storing each number as a floating-point number uses up much more space than does storing such numbers as integers. Other BASICs (such as the Acorn Atom and the BBC Microcomputer, for example) allow you to specify which sort of numbers you wish to deal with, and allots only the necessary space to do so, thus maximising memory and processing speed. The ZX80, of course, works only in integers.

Anyway, here is the first benchmark:

```
 1 REM *BENCHMARK ONE*
 5 IF INKEY$<>"" THEN GOTO 5
10 IF INKEY$="" THEN GOTO 10
20 FOR K=1 TO 1000
30 NEXT K
40 PRINT "0"
```

The built-in FOR . . . NEXT function on the ZX81 incorporates a routine to compare the variable K with 1000, so this program runs relatively quickly. The second test uses the comparison statement IF. This runs more slowly, because each time the computer comes across the IF K less than, it has to look up the value of K in its variable store, and compare this with 1000. This, as you'll see, takes time. If you find, by the way, that you do not have the patience to wait while your little computer runs through each of these things a thousand times, change the 1000 in line 50 to 100. This is benchmark two:

```
 1 REM *BENCHMARK TWO*
 5 IF INKEY$<>"" THEN GOTO 5
10 IF INKEY$="" THEN GOTO 10
20 LET K=0
30 LET K=K+1
50 IF K<1000 THEN GOTO 30
60 PRINT "0"
```



Our third test in the series simply adds a little bit of arithmetic within the loop (line 40). The result of the calculation is assigned to the second variable, that is, it is set equal to A. The difference in your computer time between the running of test two and test three is directly related to the time it takes your computer to do a few simple sums. Benchmark three:

```
  1 REM *BENCHMARK THREE*
  5 IF INKEY$<>"" THEN GOTO 5
 10 IF INKEY$="" THEN GOTO 10
 20 LET K=0
 30 LET K=K+1
 40 LET A=K/K*K+K-K
 50 IF K<1000 THEN GOTO 30
 60 PRINT "0"
```

In test four, we use numeric constants (ie numbers) instead of variables. This test should run slightly faster than the previous one because there is less need for the ZX81 to look up variables. Benchmark four:

```
  1 REM *BENCHMARK FOUR*
  5 IF INKEY$<>"" THEN GOTO 5
 10 IF INKEY$="" THEN GOTO 10
 20 LET K=0
 30 LET K=K+1
 40 LET A=K/2*3+4-5
 50 IF K<1000 THEN GOTO 30
 60 PRINT "0"
```

The fifth benchmark introduces a wonderful character, the Phantom Subroutine Call. This tests just how well a computer stores the return address from a subroutine. A subroutine, as you probably know, sends action to the line specified (as in GOSUB 700) then follows the program through from that line until it hits the word RETURN then sends action back to the line *after* the line which had the GOSUB command (in this case it is line 50). If a computer system is well designed, the extra time taken to process the phantom call should be minimal. Run benchmark five, and see if it takes significantly longer than test four:

```
  1 REM *BENCHMARK FIVE*
  5 IF INKEY$<>"" THEN GOTO 5
 10 IF INKEY$="" THEN GOTO 10
 20 LET K=0
 30 LET K=K+1
 40 LET A=K/2*3+4-5
 45 GOSUB 700
 50 IF K<1000 THEN GOTO 30
 60 PRINT "0"
700 RETURN
```

Things get a bit more hairy in benchmark six. Not only do we keep our Phantom Subroutine Call, but we tell the computer to DIMension an array. When the computer initialises an array, it sets aside in memory a number of locations to store the contents of the elements of the array. Specifying memory requirements takes a certain time, depending on the way in which the variables are stored. Benchmark six:

```
  1 REM *BENCHMARK SIX*
  5 IF INKEY$<>"" THEN GOTO 5
 10 IF INKEY$="" THEN GOTO 10
 20 LET K=0
 25 DIM M(5)
 30 LET K=K+1
 40 LET A=K/2*3+4-5
 45 GOSUB 700
 46 FOR L=1 TO 5
 48 NEXT L
 50 IF K<1000 THEN GOTO 30
 60 PRINT "0"
700 RETURN
```

The seventh benchmark fills up this array (lines 46, 47 and 48) every time you run through the loop. If you need to cook a dinner for 10 guests, take the dog for a 20 mile walk or repaint the living room, do it while this test is underway. Benchmark seven:

```
  1 REM *BENCHMARK SEVEN*
  5 IF INKEY$<>"" THEN GOTO 5
 10 IF INKEY$="" THEN GOTO 10
 20 LET K=0
 25 DIM M(5)
 30 LET K=K+1
 40 LET A=K/2*3+4-5
 45 GOSUB 700
 46 FOR L=1 TO 5
 47 LET M(L)=A
 48 NEXT L
 50 IF K<1000 THEN GOTO 30
 60 PRINT "0"
700 RETURN
```

The final test in the series — which you won't be able to run on a ZX80 — was introduced as a test of the various numeric functions of the interpreters. A badly written logarithm calculation may cause the result time to be very slow. It is worth testing each function that is available on your computer separately to establish both its accuracy and speed of operation. The 'raising to a power' (**, shift H) is particularly slow on the ZX81, a fact which can be put to good use if you wish to slow down a loop in a program, without using the jerkiness of PAUSE, and without using two lines to set up a FOR . . . NEXT loop.

Entering LET Z = RND**RND** RND will use the slowness of the RND function, and the lethargic nature of the ** to produce an impressive drop in speed. This is ideal in programs when you want the computer to pause slightly to make it look as if it is thinking. Anyway, back to benchmark eight. Run this, noting that the 1000 used in the other tests has been cut to 100 (line 70) because you could well leave school, marry, have kids and all that before the test was over if you let it mount up until K was 1000. If you have already cut the 1000 in the previous tests to 100, drop this one to 20. Benchmark eight:

```
  1 REM *BENCHMARK EIGHT*
  5 IF INKEY$<>"" THEN GOTO 5
 10 IF INKEY$="" THEN GOTO 10
 20 LET K=0
 30 LET K=K+1
 40 LET A=K**2
 50 LET B=LN (K)
 60 LET C=SIN (K)
 70 IF K<100 THEN GOTO 30
 80 PRINT "0"
```

Try running the tests in both SLOW and FAST modes, and if you get the chance, try them on another kind of computer to see how well the ZX81 measures up.

The tremendous slowdown in speed which occurs in SLOW happens because the computer spends far more time looking after the steadiness of the picture than it does actually doing any other useful work. In SLOW, the ZX81 spends only around 1.28 milliseconds of every 20 milliseconds doing anything other than looking after the TV picture.

Let us know how you get on with these tests, and tell us how your results on the ZX machines compare with the same results run on other makes of computers owned by your friends. We'll be discussing your results in the next issue of ZX Computing.

akes a
les are

```
5
10
```

nd 48)
dinner
e living

```
5
10
```

run on a
nctions
on may
ch func-
ish both
er' (**,
e put to
hout us-
set up a

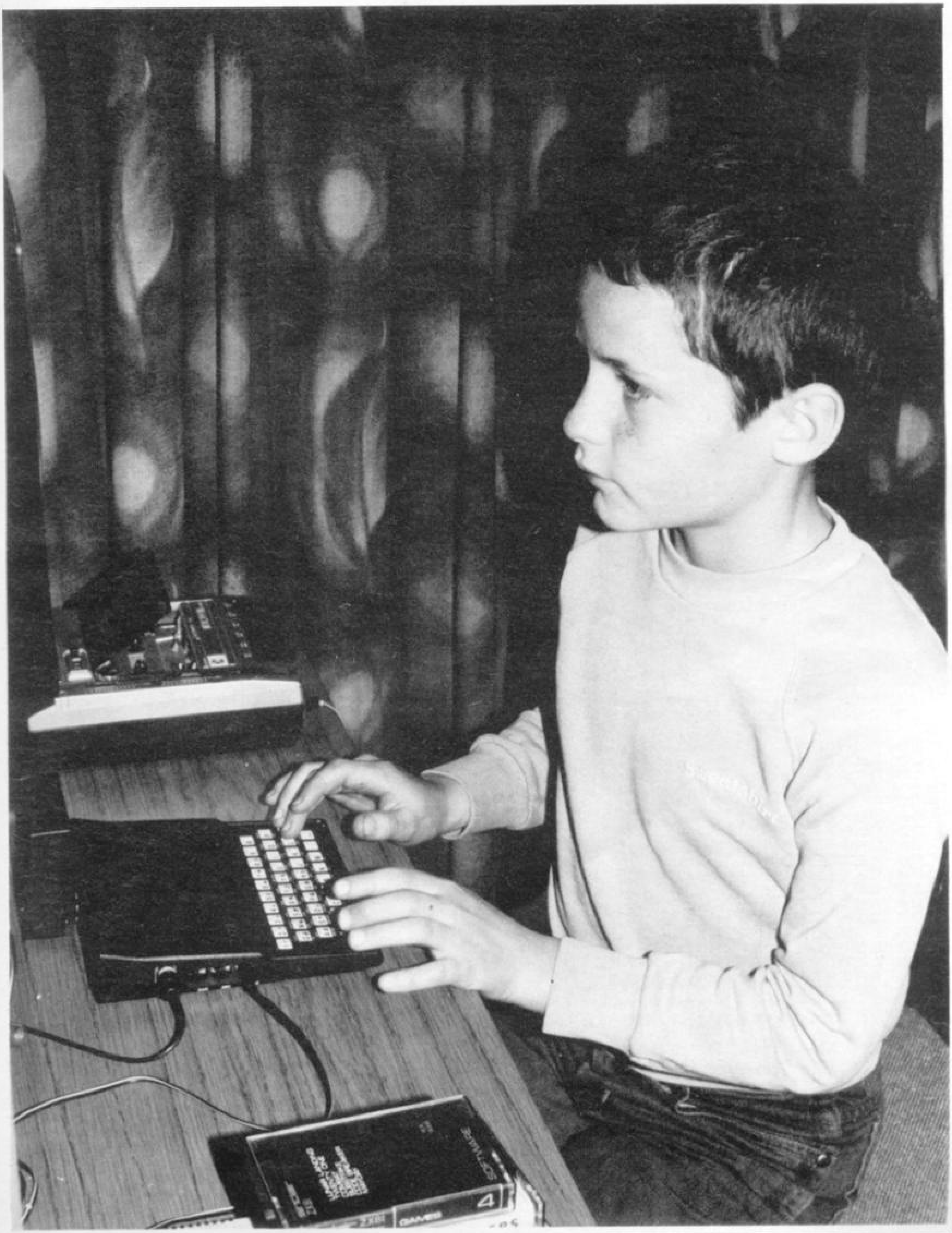
yness of
duce an
ou want
hinking.
e 1000
use you
fore the
ou have
s one to

```
5
10
```

nd if you
see how

SLOW
ng after
ny other
28 milli-
an look-

us how
sults run
We'll be
g.



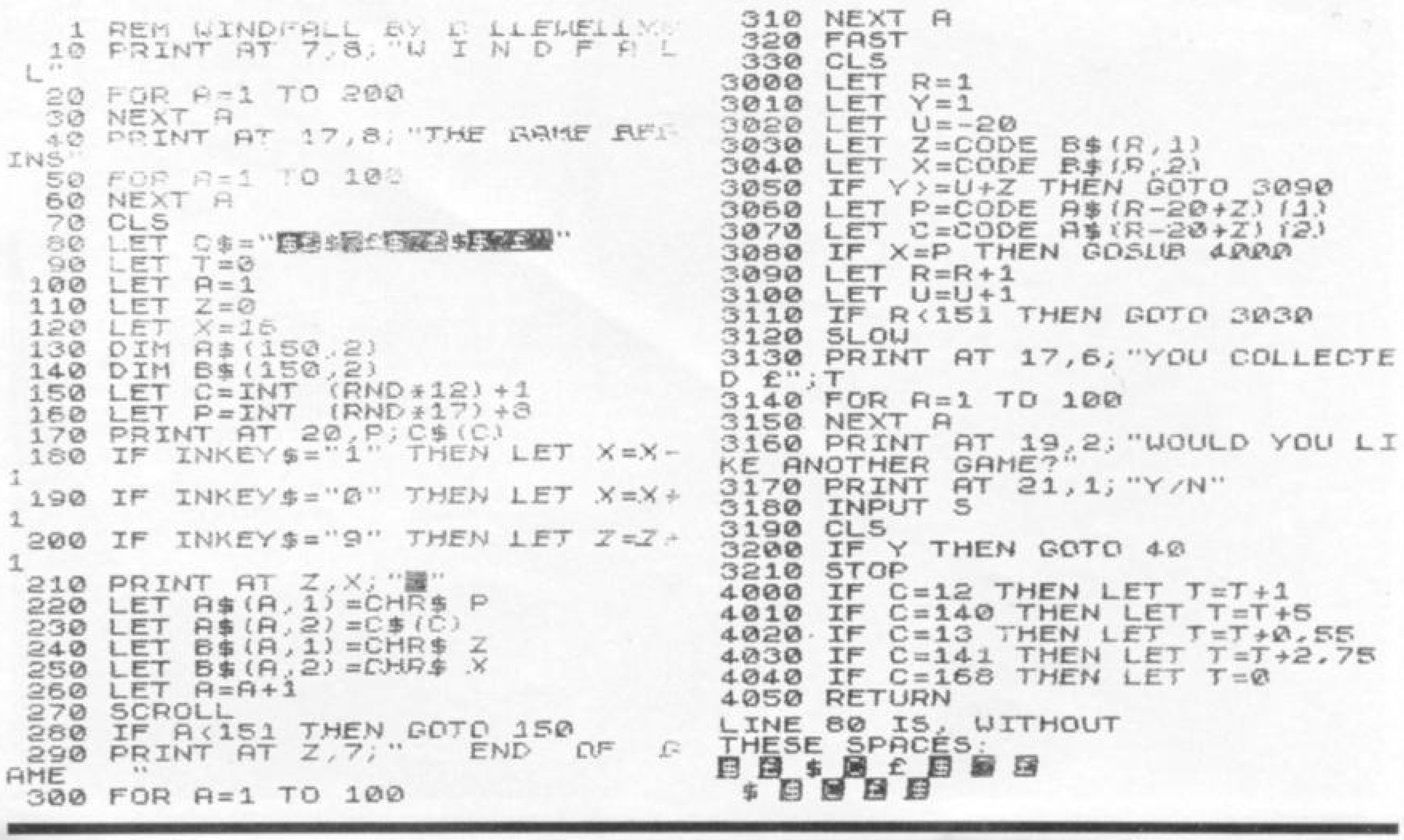
Windfall

From Ottery St Mary in Devon comes this program by C Llewellyn, designed to appeal to your materialistic instincts. Run over the money flowing up fro mthe bottom of your TV, avoiding the counterfeit cash.

The aim of the game is to acquire as many currency notes as possible during the one minute 23 seconds the game runs, without picking up any inverse Cs, which determine that all previous notes collected are counterfeit. In order to keep the game moving at a fairly rapid pace, the SCROLL function is used, and the information needed for scoring is fed into strings, which are processed at the end of the game. All currency, you'll be delighted to know, is converted into Sterling, at an exchange rate of 55p to the dollar. The standard pound sign is worth a pound (surprise, surprise), with the inverse pound sign worth a fiver. An ordinary dollar sign is 55p, and an inverse dollar sign is $5, or £2.75.

You are an inverse blob at the top, with "1" to move left, "0" to move right and "9" to move down. You score by running over the relevant sign, making sure you avoid the inverse C.

Line 80 reads LET C$ = "inverse$ inverse£ $ inverseC £ inverse$ inverseC inverse£ $ inverseC inverseC inverse£ inverse$" without any spaces.



```
   1 REM WINDFALL BY C LLEWELLYN
  10 PRINT AT 7,8;"W I N D F A L
 L"
  20 FOR A=1 TO 200
  30 NEXT A
  40 PRINT AT 17,8;"THE GAME BEG
INS"
  50 FOR A=1 TO 100
  60 NEXT A
  70 CLS
  80 LET C$="▨▨$▨£▨▨▨$▨▨▨▨"
  90 LET T=0
 100 LET A=1
 110 LET Z=0
 120 LET X=16
 130 DIM A$(150,2)
 140 DIM B$(150,2)
 150 LET C=INT (RND*12)+1
 160 LET P=INT (RND*17)+3
 170 PRINT AT 20,P;C$(C)
 180 IF INKEY$="1" THEN LET X=X-
1
 190 IF INKEY$="0" THEN LET X=X+
1
 200 IF INKEY$="9" THEN LET Z=Z+
1
 210 PRINT AT Z,X;"▨"
 220 LET A$(A,1)=CHR$ P
 230 LET A$(A,2)=C$(C)
 240 LET B$(A,1)=CHR$ Z
 250 LET B$(A,2)=CHR$ X
 260 LET A=A+1
 270 SCROLL
 280 IF A<151 THEN GOTO 150
 290 PRINT AT Z,7;"   END  OF  G
AME  "
 300 FOR A=1 TO 100
 310 NEXT A
 320 FAST
 330 CLS
3000 LET R=1
3010 LET Y=1
3020 LET U=-20
3030 LET Z=CODE B$(R,1)
3040 LET X=CODE B$(R,2)
3050 IF Y>=U+Z THEN GOTO 3090
3060 LET P=CODE A$(R-20+Z)(1)
3070 LET C=CODE A$(R-20+Z)(2)
3080 IF X=P THEN GOSUB 4000
3090 LET R=R+1
3100 LET U=U+1
3110 IF R<151 THEN GOTO 3030
3120 SLOW
3130 PRINT AT 17,6;"YOU COLLECTE
D £";T
3140 FOR A=1 TO 100
3150 NEXT A
3160 PRINT AT 19,2;"WOULD YOU LI
KE ANOTHER GAME?"
3170 PRINT AT 21,1;"Y/N"
3180 INPUT S
3190 CLS
3200 IF Y THEN GOTO 40
3210 STOP
4000 IF C=12 THEN LET T=T+1
4010 IF C=140 THEN LET T=T+5
4020 IF C=13 THEN LET T=T+0.55
4030 IF C=141 THEN LET T=T+2.75
4040 IF C=168 THEN LET T=0
4050 RETURN
LINE 80 IS, WITHOUT
THESE SPACES:
▨ ▨ $ ▨ £ ▨ ▨ ▨
 $ ▨ ▨ ▨ ▨
```

ZX COMPUTING

Autumn issue on sale July 23

And we've got even more for you next time!

Make sure you get the next bumper issue of ZX Computing.

Features include:

. How to produce your own user-defined graphics
. Fearless book reviews
. Penetrating software reviews
. Using the ZX81 in business
. Getting to grips with machine code
. A host of games, ZX80 and ZX81, 1K and 16K

If you're serious about exploring the full potential of your Sinclair computer, and you want an easy-to-understand series of articles to improve and develop your programming skills, then ZX Computing is for you.

Fill in the coupon below and give it to your newsagent

Newsagent,
Please reserve me a copy of the Autumn issue of
ZX Computing, available July 23rd.

Name ....

Address ....

....

....

1K Chess?

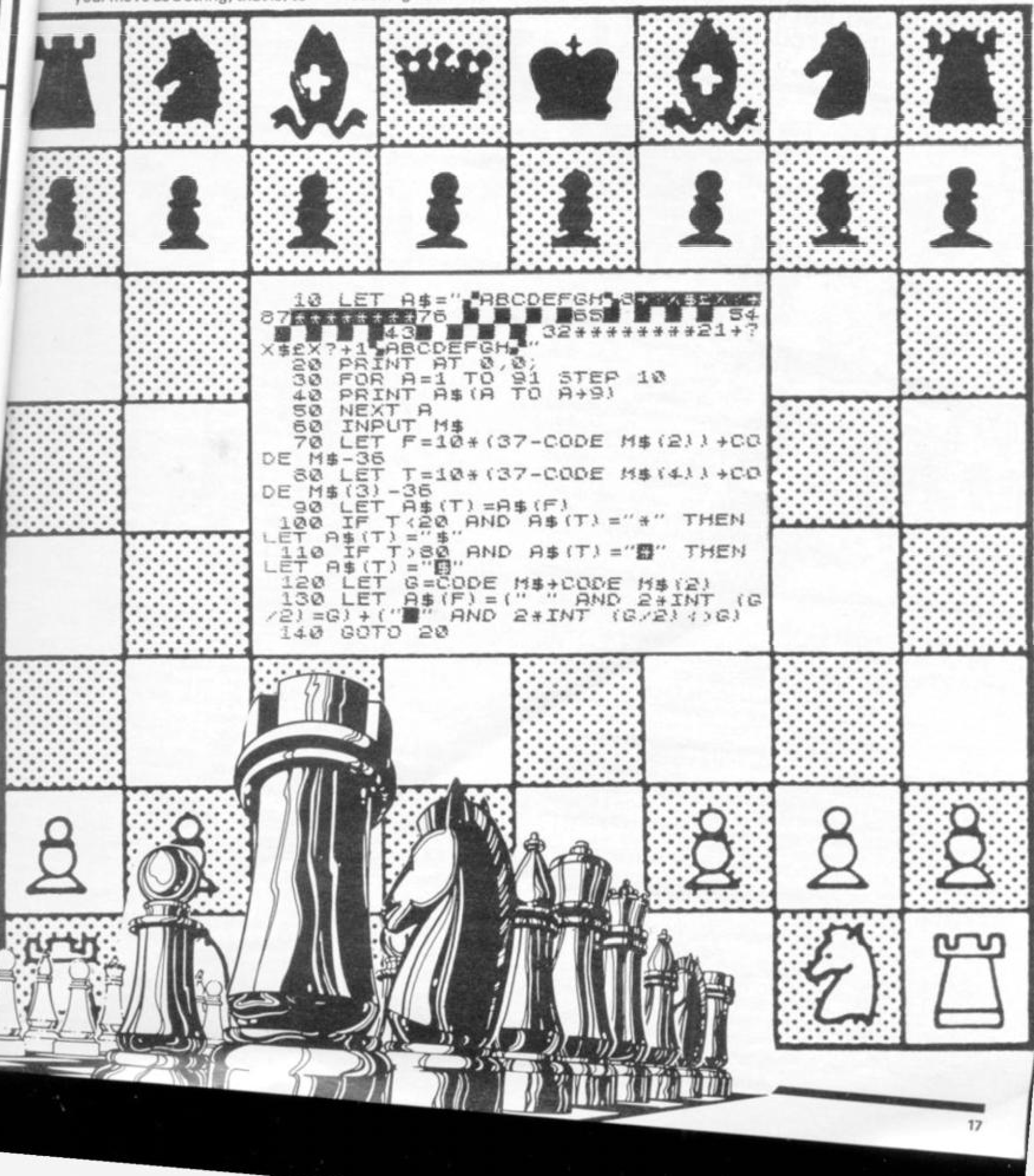
Graham Charlton shows you how to do the impossible — squeeze a kind of chess into 1K.

This program, which just fits into 1K, allows two players to play chess using the ZX81 as their board. The program can easily be modified to trigger the printer so you have a permanent record of the game.

It is very easy to use. When the prompt appears, just enter your move as a string, that is, to move from E2 to E4, enter "E2E4", then press NEWLINE, and the board will be reprinted, with your move shown. You need to enter two separate strings (such as E1G1 and H1F1) to castle.

Pawns are automatically promoted to Queen when reaching the final rank. The pieces are: + Rook; X Bishop; $ Queen; £ King and * Pawn.

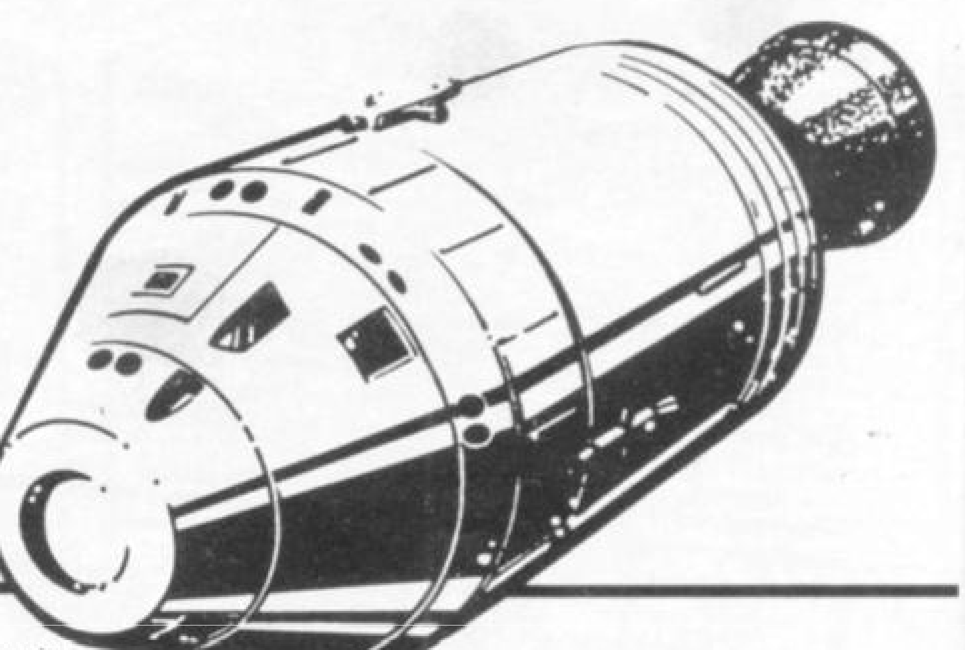
If you have extra memory, you can enter an entire game in one string — M$ — at the beginning of the program, deleting line 60 and adding 135 LET M$ = M$ (5 TO). Then the computer will whizz through a game move by move. It is fascinating to watch. You may wish to add a loop to slow things down a bit so you can work out what is going on.



```
  10 LET A$=" ▗ABCDEFGH▝8+?X$£X?+
87********76 █ █ █ █65█ █ █ █ 54
█ █ █ █43█ █ █ █ 32********21+?
X$£X?+1▝ABCDEFGH▖"
  20 PRINT AT 0,0;
  30 FOR A=1 TO 91 STEP 10
  40 PRINT A$(A TO A+9)
  50 NEXT A
  60 INPUT M$
  70 LET F=10*(37-CODE M$(2))+CO
DE M$-36
  80 LET T=10*(37-CODE M$(4))+CO
DE M$(3)-36
  90 LET A$(T)=A$(F)
 100 IF T<20 AND A$(T)="*" THEN
LET A$(T)="$"
 110 IF T>80 AND A$(T)="▓" THEN
LET A$(T)="▓"
 120 LET G=CODE M$+CODE M$(2)
 130 LET A$(F)=(" " AND 2*INT (G
/2)=G)+("█" AND 2*INT (G/2)<>G)
 140 GOTO 20
```

Orbit

S M King from Bristol takes you and your ZX81 far out into space, where you can obsereve the four innermost planets of the solar system — Mercury, Venus, good old Earth and Mars — happily circling the sun.



S M King from Bristol takes you and your ZX81 far out into space, where you can observe the four innermost planets of the solar system — Mercury, Venus, good old Earth and Mars — happily circling the sun.

Orbit is a program which was originally intended to produce an animated simulation of the movement of the four inner planets in our solar system around the sun. However, it soon became apparent that producing accurate elliptical orbits was going to be rather difficult, especially with the limited 64x44 resolution available. The result is that the program uses circular orbits to display the relative motion of the planets at the expense of accuracy. Whilst it may still find use in an instructive role it would be much better placed in, say, a "Star Trek" type of program as a "long-range scanner report".

The Program

The program was written on a Sinclair ZX81 with 8K Floating-point BASIC and 16K of RAM, although it by no means uses this much. With the omission of REM statements and with the use of other space-saving tips (which will be discussed later) the program should fit easily within 7K — with four planets.

Documentation is given to aid conversion to other micros and the program should RUN unaltered on a ZX80 with the 8K BASIC.

Lines 30-76: PRINT the introduction. Line 35, for example, PRINTs one blank line.

Lines 80 & 85: cause the introduction to be displayed for 4000/50 (80) seconds. The POKE instruction simply resets the system variable. These lines could be replaced by a FOR/NEXT loop.

Line 90: CLS is the "Clear Screen" instruction.

Lines 100-115: assign variables to the number of radians each planet moves per day. The numbers are expressed in scientific notation; ie, 57·1E-03 = 0·0571.

Lines 120 & 125: pixel co-ordinates for the start position of Mercury (Hermes).

Lines 130 & 135: pixel co-ordinates for the start position of Venus.

Lines 140 & 145: pixel co-ordinates for the start position of Earth.

Lines 150 & 155: pixel co-ordinates for the start position of Mars.

Lines 165-180: assign variables to the "orbital radii" of each planet. The numbers are the number of pixels.

Lines 200-220: display the first "question". The animated display will show successive earth days, months or years.

Line 225: causes the program to wait until a number (the choice) is entered.

Line 230: checks to see if the number entered is a 1, 2 or 3. If it is then the program continues at line 245. If not, then a reminder is PRINTed (line 235) and the program returns to the INPUT statement at 225.

Lines 250-260: choose the period incrementing factor (in days).

Line 265: sets the period counter to one.

Lines 300-305: display the second "question". How many days, months or years are to be displayed?

Line 315: makes the number entered at line 310 an integer if it was a decimal. It has no effect if the number was already an integer. Note that the number must be greater than one for the program to continue.

Line 335: displays the third "question". What real-time time delay is required between successive "frames" of the animation? Line 345 prevents the delay from being zero seconds. Line 350 converts the figure entered into a number used by the PAUSE instruction.

Lines 415-443: DIMension 8 arrays (2 per planet) to hold the pixel co-ordinates for PLOTting and UNPLOTting. The first array in each case holds X-coords, the second Y-coords.

Lines 445-520: calculate the positions that each of the four planets will be at during the simulation. Lines 450, 470, 490 and 510 calculate X pixel co-ordinates, whilst Lines 455, 475, 495 and 515 calculate Y pixel co-ordinates.

Lines 525-540: "black-in" the starting positions of the planets.

Line 545: sets the main loop counter to one.

Line 550: increments the period counter.

Lines 560-575: display the positions of the planets during the simulation.

Line 587: checks to see if the required number of days/months/years have been displayed. If they have then the program jumps to line 700.

Lines 590-605: "black-in" the positions of the planets during the simulation.

Line 610: increments the loop counter.

Line 620: returns the loop.

PROGRAM BETWEEN LINES 700 AND 999 ONLY WORKS WHEN THE MAIN SIMULATION HAS FINISHED.

Lines 700-706: PRINT the fourth "question" down the right hand side of the screen.

Line 710: waits until a letter(s) are entered. Line 715 causes the program to jump to line 730 if a "Y" is entered. If an "N" is entered then the program STOPs, otherwise the program waits at the INPUT statement at line 710.

Lines 735-780: display the fifth "question" and wait for a proper INPUT at line 775.

Lines 790-840: execute the various subroutines.

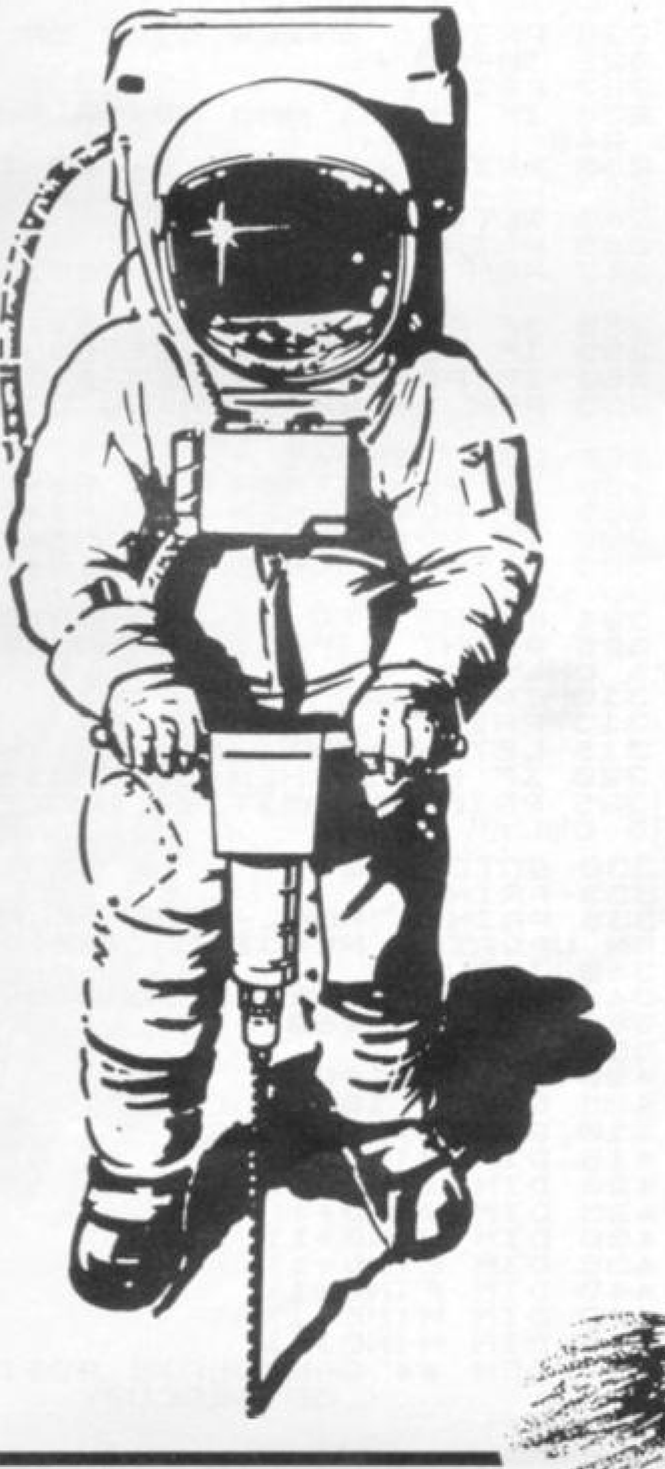
Line 999: causes the program to halt. Equivalent to an END statement.

SUBROUTINE AT LINE 1000:
"blacks-in" the left ¾ of the screen.
Line 1010 contains 23 inverse spaces.
Line 1025 puts the sun (an inverse asterisk) in the centre of the black square.

SUBROUTINE AT LINE 1050:
displays the planets in their starting positions.

SUBROUTINE AT LINE 1100:
displays the period.

SUBROUTINE AT LINE 1145:
removes any print statements on the right-hand side by PRINTing blank lines. Nine spaces are contained in the string in line 1155.



SUBROUTINES AT LINES 1495, 1595, 1695 and 1795:
contain the different pieces of data on each of the four planets.

SUBROUTINE AT LINE 1995:
PRINTs the general headings for the data.

SUBROUTINE AT LINE 2100:
converts plot co-ordinates to PRINT AT co-ordinates and then displays each planet's initial letter by the pixel square representing it on the display.
Line 2145 contains an inverse "H".
Line 2150 contains an inverse "V".
Line 2155 contains an inverse "E".
Line 2160 contains an inverse "M".

The Display

(a) PRINT AT co-ordinates:
The ZX81 PRINT AT statement is of the form — 9999 PRINT AT (line), (column); "(whatever is to be PRINTed)" — where:
line numbers are from 0 to 21 inclusive.
column numbers are from 0 to 31 inclusive.

(b) PLOT/UNPLOT co-ordinates:
The ZX81 PLOT/UNPLOT statements are of the form — 9999 PLOT (X pixel coord), (Y pixel coord) . . . "blacks-in" pixel
9999 UNPLOT (X pixel coord), (Y pixel coord) . . . "whites-out" pixel — where:
X coords are from 0 to 63 inclusive
Y coords are from 0 to 43 inclusive

Normally, the ZX81 prints BLACK ON WHITE. During the simulation, the first 23 columns (down all 22 lines) are "blacked-in" (Line 1010).

So that the planets and sun can be seen on the black background an UNPLOT statement DISPLAYS the planet, whilst a PLOT statement will "black-in" the square again. If your computer normally writes WHITE ON BLACK then it may be found better to make the right hand side of your screen WHITE with your inverse spaces. If you do so remember to change all writing in PRINT statements after line 620 to your inverse graphics, delete lines 1000, 1005, 1010 and 1015, and change the characters in lines 1025, 2145, 2150, 2155 and 2160 to your "normal" and NOT inverse characters.

It may have been noticed that in lines 560-575 and 590-605 the Y coords are being deducted from 44 (The greatest Y pixel coord). This causes the planets to revolve ANTICLOCKWISE about the sun. If the "44 —" is omitted then the planets revolve CLOCKWISE. All PRINT AT, PLOT/UNPLOT statements are referenced to the top LEFT-HAND corner of the screen.

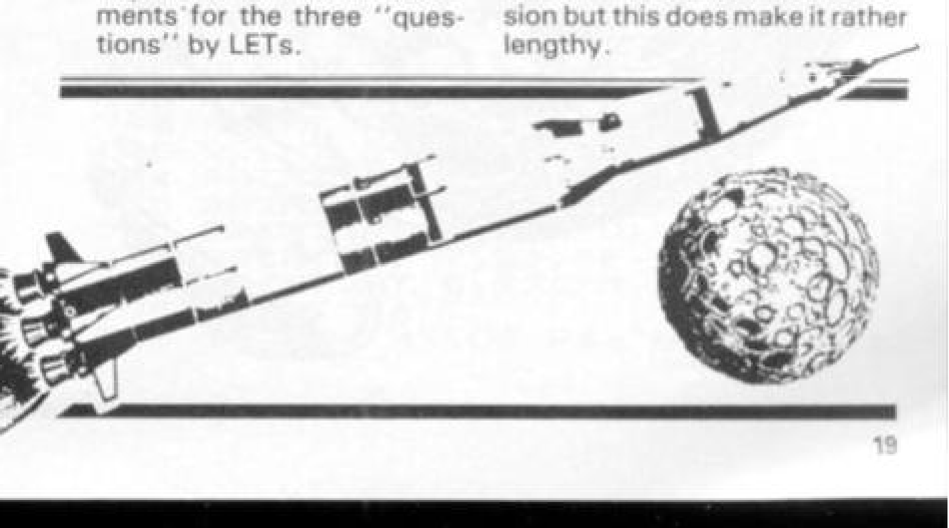
Space Saving

If the program IS to be used as part of some space-borne adventure game then its length will undoubtedly want to be shortened. I suggest the following course of action:

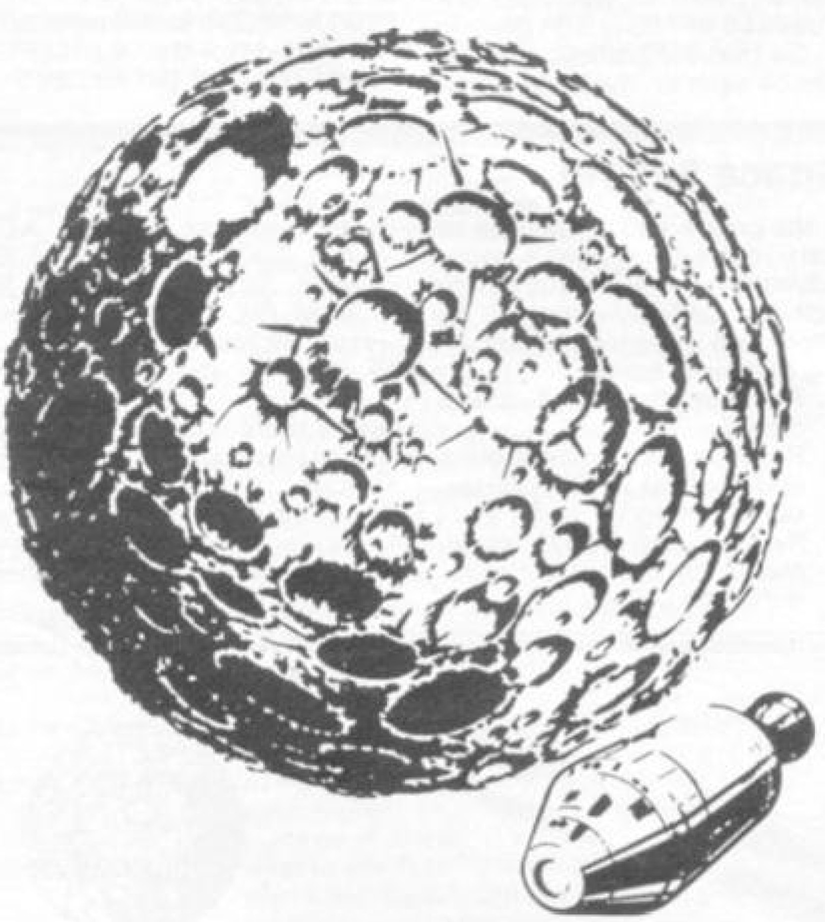
1. Remove the REM statements.
2. Remove the introduction and the first THREE "questions".
3. Replace the INPUT statements for the three "questions" by LETs.
4. Put variables AH, AV, AE, AM, SXH & SYH, SXV & SYV, SXE & SYE, SXM & SYM, RH, RV, RE and RM into their respective routines.

(5. Reduce the number of planets.)

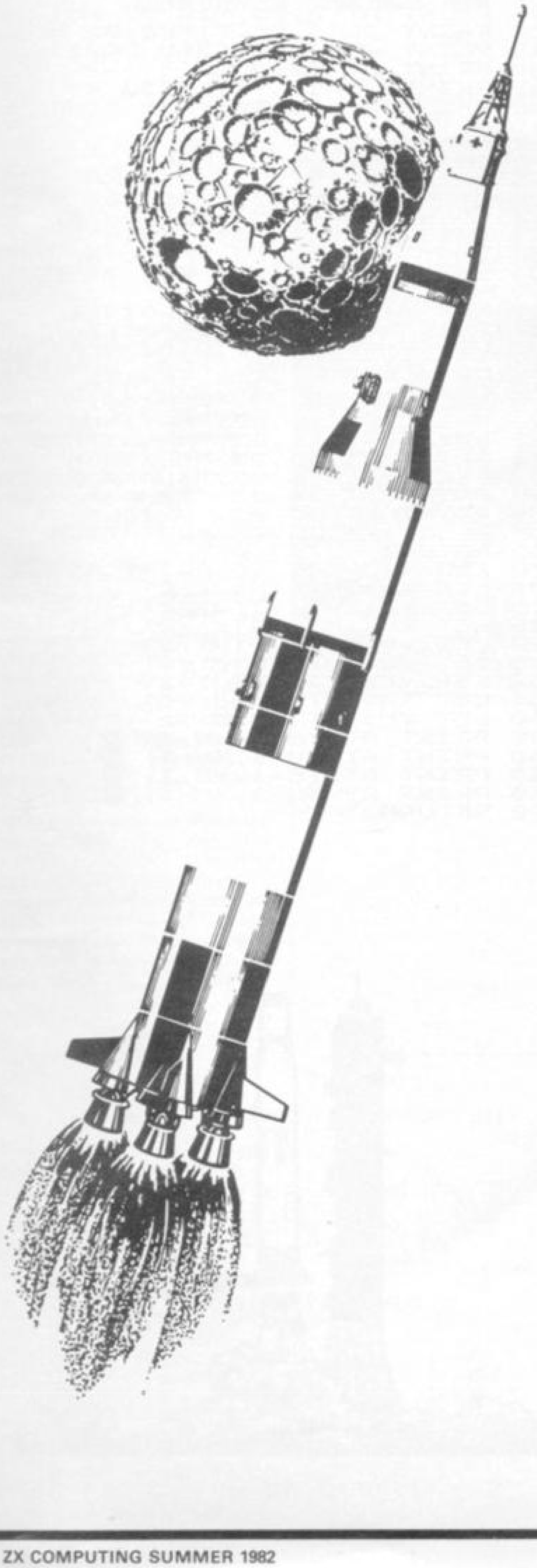
Again there is no reason why the planets should be called Mercury, Venus or whatever. The program was written in a "modular" form to aid conversion but this does make it rather lengthy.



```
  10 REM ** ORBIT  BY S.KING
  15 REM ** WRITTEN FEB 1982 FOR
  20 REM ** 8K ZX81 WITH 16K RAM
  25 REM
  30 PRINT "ORBIT"
  35 PRINT
  40 PRINT "THIS SIMULATION DISP
LAYS THE      ORBITS OF THE FOUR I
NNER PLANETSOF OUR SOLAR SYSTEM.
"
  45 PRINT
  50 PRINT "THE PLANETS START AT
 THERE        POSITIONS (RELATIVE
TO THE SUN  AND EACH OTHER) AS O
F JAN 1 1982AND MOVE AT SCALE SP
EEDS."
  55 PRINT
  60 PRINT "THE ORBITAL RADII HA
VE ALSO BEENSCALED DOWN ACCORDIN
GLY."
  65 PRINT
  70 PRINT "THE PROGRAM ALSO PRO
VIDES THE    USER WITH THE FACILI
TY FOR A     SHORT RUN-DOWN ON TH
E PHYSICAL  FEATURES OF EACH PLA
NET."
  75 PRINT
  76 PRINT "THE PROGRAM DOES NOT
 ALLOW FOR  ORBITAL ECCENTRICITY
.  HENCE IT IS NOT VERY ACCURATE
 IN PLOTTING"
  80 PAUSE 4000
  85 POKE 16437,255
  90 CLS
  95 REM ** RADIANS MOVED EACH
            DAY BY PLANETS
  97 REM ** MERCURY
 100 LET AH=57.10E-03
 103 REM ** VENUS
 105 LET AV=27.62E-03
 107 REM ** EARTH
 110 LET AE=17.21E-03
 113 REM ** MARS
 115 LET AM=08.30E-03
 117 REM ** STARTING POSITION
            MERCURY (HERMES)
 120 LET SXH=21
```

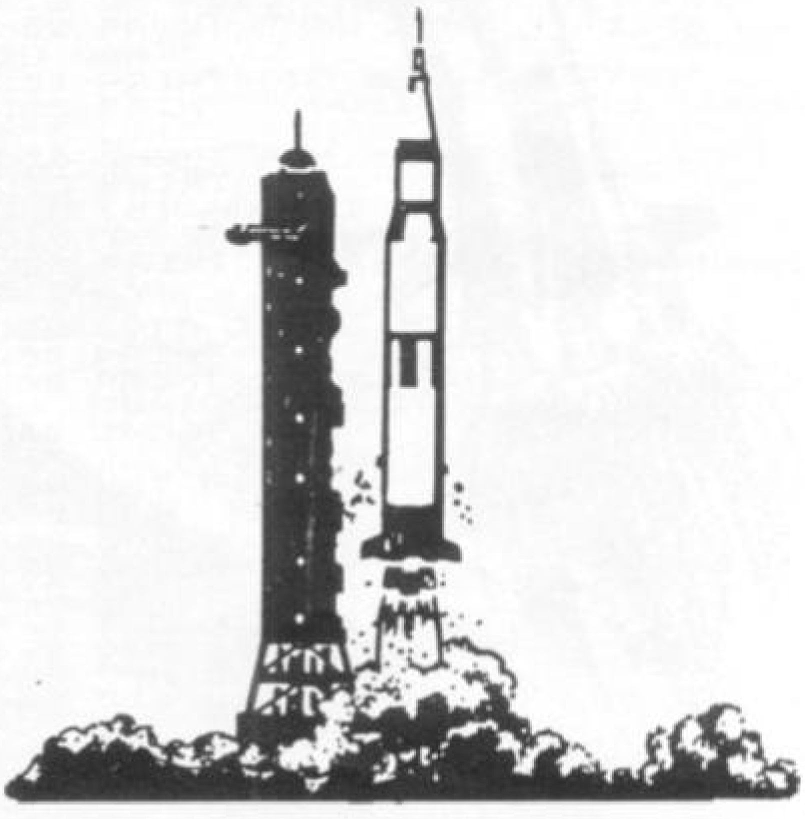


```
 125 LET SYH=17
 127 REM ** STARTING POSITION
            VENUS
 130 LET SXV=17
 135 LET SYV=14
 137 REM ** STARTING POSITION
           EARTH
 140 LET SXE=22
 145 LET SYE=9
 147 REM ** STARTING POSITION
            MARS
 150 LET SXM=42
 155 LET SYM=24
 160 REM ** ORBITAL RADII
 163 REM ** MERCURY
 165 LET RH=5
 167 REM ** VENUS
 170 LET RV=9
 173 REM ** EARTH
 175 LET RE=13
 177 REM ** MARS
 180 LET RM=20
 200 PRINT "AFTER WHAT TIME PERI
OD WOULD YOULIKE THE POSITIONS O
F THE        PLANETS UPDATED?"
 203 PRINT
 205 PRINT "1. EACH DAY"
 210 PRINT "2. EACH 30-DAY MONTH
"
 215 PRINT "3. EACH 365-DAY YEAR
"
 220 PRINT "ENTER 1, 2 OR 3"
 225 INPUT PD
 227 PRINT
 230 IF PD>=1 AND PD<=3 THEN GOT
O 245
 235 PRINT "PLEASE ENTER 1, 2 OR
3"
 240 GOTO 225
 245 PRINT
 247 REM ** S = TIME PERIOD
          INCREMENTING FACTOR
 250 IF PD=1 THEN LET S=1
 255 IF PD=2 THEN LET S=30
 260 IF PD=3 THEN LET S=365
 263 REM ** SET PERIOD COUNTER
            TO ONE
 265 LET TIME=1
 280 IF PD=1 THEN LET A$="DAY"
 285 IF PD=2 THEN LET A$="MONTH"
 290 IF PD=3 THEN LET A$="YEAR"
 300 PRINT "HOW MANY ";A$;"S DO
YOU WANT"
 301 PRINT "TO BE DISPLAYED?"
 305 PRINT "(POSITVE WHOLE NUMBE
RS ONLY)"
 310 INPUT ND
 313 PRINT
 315 LET ND=INT (ND)
 320 IF ND>1 THEN GOTO 335
 325 PRINT "POSITIVE WHOLE NUMBE
RS ONLY"
 330 GOTO 310
 333 PRINT
 335 PRINT "REAL-TIME DELAY BETW
EEN UPDATES REQUIRED (SECONDS)?"
 340 INPUT P
 345 IF P<1 THEN GOTO 340
 350 LET P=P*50
 355 CLS
 400 GOSUB 1000
 405 GOSUB 1050
 410 GOSUB 1100
 415 DIM H(ND+1)
 420 DIM I(ND+1)
 425 DIM V(ND+1)
 430 DIM W(ND+1)
 435 DIM E(ND+1)
 440 DIM F(ND+1)
 442 DIM M(ND+1)
 443 DIM N(ND+1)
 444 REM ** CALCULATE POSITIONS
            OF MERCURY
```



```
 445 LET Z=1
 446 FOR R=86 TO (86+ND*S) STEP
S
 450 LET H(Z)=22+RH*SIN (AH*(86+
S*Z))
 455 LET I(Z)=22+RH*COS (AH*(86+
S*Z))
 457 LET Z=Z+1
 460 NEXT R
 463 REM ** CALCULATE POSITIONS
          .OF VENUS
 464 LET Z=1
 465 FOR R=207 TO (207+ND*S) STE
P S
 470 LET V(Z)=22+RV*SIN (AV*(207
+S*Z))
 475 LET W(Z)=22+RV*COS (AV*(207
+S*Z))
 477 LET Z=Z+1
 480 NEXT R
 483 REM ** CALCULATE POSITIONS
           OF EARTH
 484 LET Z=1
 485 FOR R=1 TO 1+ND*S STEP S
 490 LET E(Z)=22+RE*SIN (AE*(1+S
*Z))
 495 LET F(Z)=22+RE*COS (AE*(1+S
*Z))
 497 LET Z=Z+1
 500 NEXT R
 503 REM ** CALCULATE POSITIONS
           MARS
 504 LET Z=1
 505 FOR R=183 TO (183+ND*S) STE
P S
 510 LET M(Z)=22+RM*SIN (AM*(183
+S*Z))
 515 LET N(Z)=22+RM*COS (AM*(183
+S*Z))
 517 LET Z=Z+1
 520 NEXT R
 525 PLOT SXH,SYH
 530 PLOT SXV,SYV
 535 PLOT SXE,SYE
 540 PLOT SXM,SYM
 545 LET J=1
 550 LET TIME=TIME+1
 555 GOSUB 1100
 560 UNPLOT H(J),(44-I(J))
 565 UNPLOT V(J),(44-W(J))
 570 UNPLOT E(J),(44-F(J))
 575 UNPLOT M(J),(44-N(J))
 580 PAUSE P
 585 POKE 16437,255
 587 IF J=ND-1 THEN GOTO 700
 590 PLOT H(J),(44-I(J))
 595 PLOT V(J),(44-W(J))
 600 PLOT E(J),(44-F(J))
 605 PLOT M(J),(44-N(J))
 610 LET J=J+1
 620 GOTO 550
 700 PRINT AT 5,23;"WOULD YOU"
 701 PRINT AT 6,23;"LIKE A"
 702 PRINT AT 7,23;"DETAILED"
 703 PRINT AT 8,23;"RUN-DOWN"
 704 PRINT AT 9,23;"ON A"
 705 PRINT AT 10,23;"PLANET?"
 706 PRINT AT 12,23;"Y OR N"
 710 INPUT Q$
 715 IF Q$="Y" THEN GOTO 730
 720 IF Q$="N" THEN STOP
 725 GOTO 710
 730 GOSUB 1150
 735 PRINT AT 0,23;"WHICH"
 740 PRINT AT 1,23;"PLANET?"
 745 PRINT AT 3,23;"ENTER"
 750 PRINT AT 4,23;"CODE:"
 755 PRINT AT 5,23;"H-MERCURY"
 760 PRINT AT 6,23;"V-VENUS"
 765 PRINT AT 7,23;"E-EARTH"
 770 PRINT AT 8,23;"M-MARS"
 775 INPUT P$
 780 IF P$="H" OR P$="V" OR P$="
```

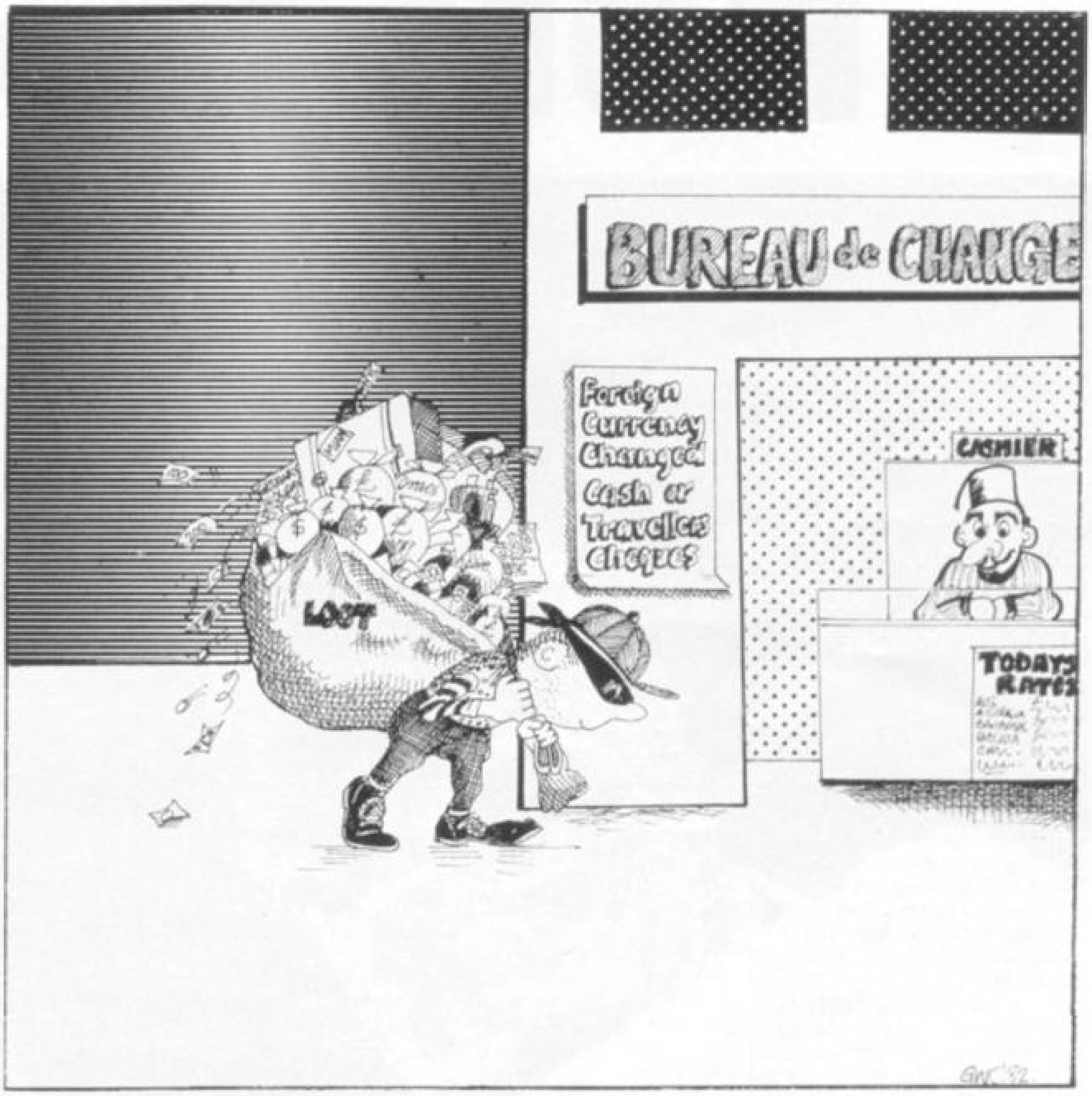
```
E" OR P$="M" THEN GOTO 790
 785 GOTO 775
 790 GOSUB 1150
 795 GOSUB 2000
 800 IF P$="H" THEN GOSUB 1500
 805 IF P$="V" THEN GOSUB 1600
 810 IF P$="E" THEN GOSUB 1700
 815 IF P$="M" THEN GOSUB 1800
 820 PRINT AT 21,23;"PRESS N/L"
 825 GOSUB 2100
 830 INPUT Z$
 835 GOSUB 1150
 840 GOTO 700
 999 STOP
1000 REM ** BLACK SCREEN SUB/R
1005 FOR I=1 TO 22
1010 PRINT "█████████████████████
██"
1015 NEXT I
1020 REM ** POSITION CENTRAL SUN
1025 PRINT AT 10,11;"■"
1030 RETURN
1050 REM ** DISPLAY PLANETS
            STARTING POSITIONS
1055 UNPLOT SXH,SYH
1060 UNPLOT SXV,SYV
1065 UNPLOT SXE,SYE
1070 UNPLOT SXM,SYM
1075 RETURN
1100 REM ** DISPLAY PERIOD
1105 PRINT AT 0,23;"EARTH"
1110 PRINT AT 1,24;A$;":"
1115 PRINT AT 2,25;TIME
1120 RETURN
1145 REM ** CLEAR R.H.S
1150 FOR I=0 TO 21
1155 PRINT AT I,23;"         "
1160 NEXT I
1170 RETURN
1495 REM ** MERCURY DATA
1500 PRINT AT 0,23;"MERCURY"
1505 PRINT AT 3,23;"4988 KM"
1510 PRINT AT 6,23;"56E+09 M"
1515 PRINT AT 9,23;"88 DAYS"
1520 PRINT AT 12,23;" -1.9"
1525 PRINT AT 16,23;" 673 K"
1530 PRINT AT 19,23;"59 DAYS"
1540 RETURN
1595 REM ** VENUS DATA
1600 PRINT AT 0,23;"VENUS"
1605 PRINT AT 3,23;"12389 KM"
1610 PRINT AT 6,23;"108E+09 M"
1615 PRINT AT 9,23;"225 DAYS"
1620 PRINT AT 12,23;" -4.4"
1625 PRINT AT 15,23;" 623 K"
1630 PRINT AT 19,23;"247 DAYS"
1640 RETURN
1695 REM ** EARTH DATA
1700 PRINT AT 0,23;"EARTH"
1705 PRINT AT 3,23;"12753 KM"
1710 PRINT AT 6,23;"150E+09 M"
1715 PRINT AT 9,23;"365 DAYS"
1720 PRINT AT 12,23;"  ---    "
1725 PRINT AT 16,23;" 333 K"
1730 PRINT AT 19,23;"23H 56MIN"
1740 RETURN
1795 REM ** MARS DATA
1800 PRINT AT 0,23;"MARS"
1805 PRINT AT 3,23;"6758 KM"
1810 PRINT AT 6,23;"228E+09 M"
1815 PRINT AT 9,23;"687 DAYS"
1820 PRINT AT 12,23;" -2.8"
1825 PRINT AT 16,23;" 298 K"
1830 PRINT AT 19,23;"24H 37MIN"
1840 RETURN
1995 REM ** GENERAL HEADINGS
2000 PRINT AT 2,23;"DIAMETER:"
2005 PRINT AT 4,23;"DISTANCE"
2010 PRINT AT 5,23;"FROM SUN:"
2015 PRINT AT 7,23;"SIDERIAL"
2020 PRINT AT 8,23;"PERIOD:"
2025 PRINT AT 10,23;"MAXIMUM"
2030 PRINT AT 11,23;"MAGNITUDE"
2035 PRINT AT 13,23;"MAXIMUM"
2040 PRINT AT 14,23;"SURFACE"
2045 PRINT AT 15,23;"TEMP:"
2050 PRINT AT 17,23;"AXIAL"
2055 PRINT AT 18,23;"ROTATION:"
2060 RETURN
2100 REM ** PLOT TO PRINT AT
2105 LET XH=INT (H(J)/2)
2110 LET YH=INT (I(J)/2)
2115 LET XV=INT (V(J)/2)
2120 LET YV=INT (W(J)/2)
2125 LET XE=INT (E(J)/2)
2130 LET YE=INT (F(J)/2)
2135 LET XM=INT (M(J)/2)
2140 LET YM=INT (N(J)/2)
2145 PRINT AT YH-1,XH-1;"H"
2150 PRINT AT YV-1,XV-1;"V"
2155 PRINT AT YE-1,XE-1;"E"
2160 PRINT AT YM-1,XM-1;"M"
2170 RETURN
```



Bureau de Change

If you're always jetting off to exotic parts of the world, like Paris, Geneva or Blackpool, you'll be in need of this program which rapidly converts your money from one currency to another.

The program, as written by Bob Perrigo, caters for six different currencies, but it can easily be modified to handle as many currencies as you need. The values of major currencies and their exchange rates with the pound are published in many newspapers, particularly the *Financial Times*. Current values have to be entered when you first RUN the program. Once the values are in RAM, you can use the program again by entering GOTO 6 instead of RUN. The word END after the semi-colon in line 100 is not a mistake. It acts just as END does in other BASICs, stopping the program. It does so in this case because the ZX81 considers the word END to be an unassigned variable, and will stop the program with the error code 2/100. Any other letter, or combination of letters, which is not used elsewhere in the program, will stop it as effectively as does END,



```
  3 REM *CURRENCY CONVERTER*
  4 REM *BOB PERRIGO/
       TIM HARTNELL*
  5 GOSUB 110
  6 PRINT "ENTER CURRENCY 1"
  7 PRINT "£ - POUNDS","$ - DOL
LARS","D - DEUTSCH MARKS","S - S
WISS FRANKS","Y - YEN","R - RAND
"
  8 INPUT A$
 10 PRINT "ENTER AMOUNT"
 12 INPUT B
 15 LET P=0
 20 IF A$="£" THEN LET P=B
 25 IF A$="$" THEN LET P=B*1/D
 30 IF A$="D" THEN LET P=B*1/M
 35 IF A$="S" THEN LET P=B*1/F
 40 IF A$="Y" THEN LET P=B*1/Y
 45 IF A$="R" THEN LET P=B*1/R
 50 IF P=0 THEN GOTO 5
 55 PRINT "ENTER CURRENCY 2"
 57 INPUT C$
```

```
 58 LET E=0
 60 IF C$="£" THEN LET E=P
 65 IF C$="$" THEN LET E=P*D
 70 IF C$="D" THEN LET E=P*M
 75 IF C$="S" THEN LET E=P*F
 80 IF C$="Y" THEN LET E=P*Y
 85 IF C$="R" THEN LET E=P*R
 90 IF E=0 THEN GOTO 55
 95 LET E=INT (E*100+.5)/100
100 PRINT A$;B;" = ";C$;E;END
110 PRINT "DEUTSCH MARK?"
120 INPUT M
130 PRINT "YEN?"
140 INPUT Y
150 PRINT "RAND?"
160 INPUT R
170 PRINT "SWISS FRANC"
180 INPUT F
190 PRINT "US DOLLAR?"
200 INPUT D
210 CLS
220 RETURN
```

The dreaded ROM-bug

Frank O'Hara looks at the bug in the original 8K ROM, and explains how it occurred. He reveals a new bug which only comes to light when running a ZX printer, and gives a checking routine which prints the powers of two from two squared to two to the 32nd, exactly.

I first became aware of the bug in the Sinclair 8K ROM after writing a fairly fast BASIC program to obtain the smallest prime factor of any odd number up to $2^{32}-1$ in a matter of seconds rather than minutes. When I tried the program out on $2^{32}-5$, the largest prime number in this range, it seemed to be taking a long time to run. On breaking in and checking the divisor, I found that it had gone well past the square root of the number (about 65536). On checking the square root, I found that the ZX81 was holding a 12 digit number (instead of 65536) as the square root of the 10 digit number $2^{32}-5$.

Further investigation revealed errors in exponentiation and even in subtraction. In the event all these errors were to trace back to a programming mistake in subtraction. To see how they arose, it is best to follow a simple example through to its source.

The ZX81 gives SQR (¼) correctly as .5. But SQR .25 is given as 1.3591409. Listing the ROM shows that SQR X is evaluated as X**.5. The instruction "PRINT .25**.5" gives the same wrong answer 1.3591409. The function X**Y in turn is evaluated as EXP (Y * LN X). The instruction "PRINT LN .25" returns the answer 0.61370564 instead of −1.3862944. Two has been added to the correct answer. At this stage we could well suspect that addition is at fault.

The last stage of detection requires one to split the LN routine in two, by uploading it into RAM and running it there. Happily this works. At least the first half runs and that is all we need. To evaluate LN X the ZX81 has to add the log of the exponent and the log of the mantissa of X. In the case of .25, the log of the exponent is correctly evaluated as −2 log 2. The log of the mantissa should then yield zero. But .25 is held approximately on the ZX81, unlike (¼), which is held exactly. The result of this is that log of the mantissa comes out as $-\frac{1}{2}^{32}$, an approximation to zero. Finally, the sum of −2 log 2 and $-\frac{1}{2}^{32}$ adds two to the correct answer because of the bug in subtraction.

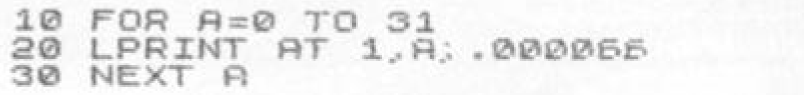
This brings us to the source of the bug in subtraction. Because subtraction just changes the sign and adds, it will help to think of simply adding on a negative number. In floating point addition, the addend (ie the number being added on) has to be lined up with the other number (the "augend") to make the addition possible. If the addend is too small to make any appreciable difference to the augend (and $-\frac{1}{2}^{32}$ is a good example when added to −2 log 2) then the addend is set to zero and nothing happens. Unfortunately the programmers thought they should do a bit more at this stage, and introduced the bug. Three bytes which were not needed (at 1733 to 1735 hex in the old ROM) had the effect of causing a "carry adjust" mechanism to be wrongly invoked, shifting the result one place right, preserving the sign bit and adding one to the exponent. This sometimes trebled the number ($4-\frac{1}{2}^{32}$ gives 12) and sometimes increased it in other ways ($10-\frac{1}{2}^{32}$ gives 26; and as stated above −2 log $2-\frac{1}{2}^{32}$ gives −2 log 2 + 2).

The sequel is that a new ROM has been produced by Sinclair Research without these three bytes. It is supposed to be available on demand, but mine took three months to arrive.

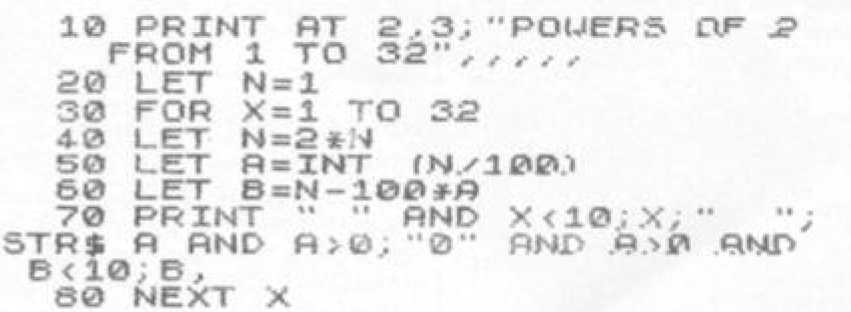
There are ways of getting around the bug. For example, LET S = 10*SQR(N/100) works for square roots.

A new, and rather subtle bug, comes to light on the printer. Enter program one, and run it. You'll be quite surprised at the result.

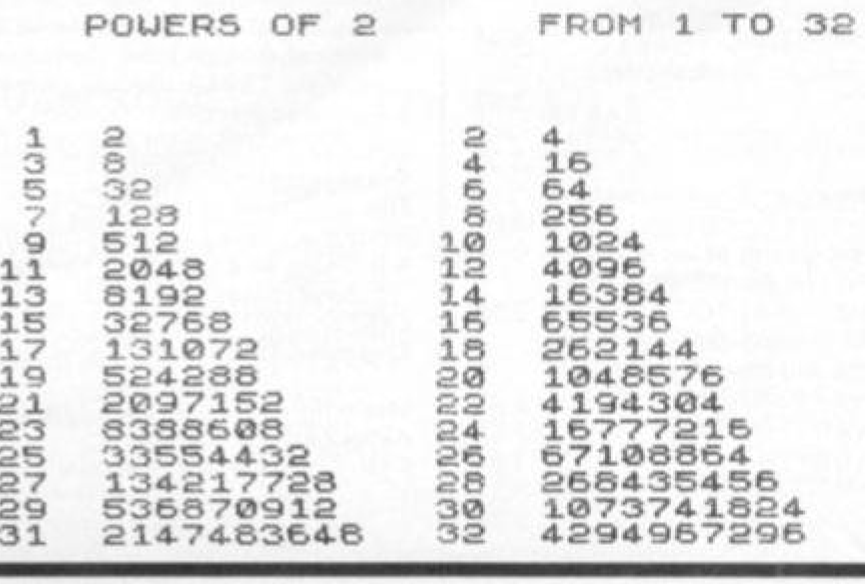
Even with the amended ROM, certain powers of two are not worked out exactly (such as 2**31), even if you display all 10 digits. Program two prints the powers of two from two squared up to two to the 32nd exactly, with all their digits. The list it produces can be useful for checking certain expressions on the ZX81. Once you've run program two as listed, try running it with LET N = 2**I in line 40. As alternative, and completely accurate expressions, I recommend LET N = 2**32/2 and LET N = 2**32 − 2 + 1. These can be fully displayed by PRINT "21";N − 21E8 and PRINT "42";N − 42E8 respectively.



```
 10 FOR A=0 TO 31
 20 LPRINT AT 1,A;.000066
 30 NEXT A
```



```
 10 PRINT AT 2,3;"POWERS OF 2
   FROM 1 TO 32",,,,,
 20 LET N=1
 30 FOR X=1 TO 32
 40 LET N=2*N
 50 LET A=INT (N/100)
 60 LET B=N-100*A
 70 PRINT " " AND X<10;X;"  ";
STR$ A AND A>0;"0" AND A>0 AND
 B<10;B,
 80 NEXT X
```



```
   POWERS OF 2        FROM 1 TO 32


  1  2                 2  4
  3  8                 4  16
  5  32                6  64
  7  128               8  256
  9  512              10  1024
 11  2048             12  4096
 13  8192             14  16384
 15  32768            16  65536
 17  131072           18  262144
 19  524288           20  1048576
 21  2097152          22  4194304
 23  8388608          24  16777216
 25  33554432         26  67108864
 27  134217728        28  268435456
 29  536870912        30  1073741824
 31  2147483648       32  4294967296
```

ZX81 Workstation...

. . . is a stylish and ergonomic plinth for the ZX81. It raises and tilts the TV to avoid eyestrain, holds the 16KRAM in place and hides the wiring and power supply. This very professional unit costs £15, a built-in power switch is £3, plus postage at £1.50, inc. VAT.

Peter Furlong Products, 125 Catford Hill, London SE6 4PR.
Callers by appointment, please. Tel 01690 7799. Visa, Access.

Moi

Mine of Information Ltd

Microcomputer Consultancy & Booksellers

ZX OTHELLO™

A major game of strategy
for the Sinclair ZX81
personal computer +
16KB RAM

- ✱ Easy to learn
- ✱ For ages 7 to 107
- ✱ Pit your wits against an inhuman intelligence!
- ✱ Nine levels of play from novice to expert

£10

SEND FOR OUR FREE CATALOGUE OF 100 SELECTED MICROCOMPUTER BOOKS

Prices include P·P in UK
Add 20% for delivery to Europe, Africa or the Middle East

Orders to: Moi (223) 1 Francis Avenue · St Albans · Herts
AL3 6BL · England · Phone 0727 52801 · Telex 925859

ZX81 SOFTWARE FROM VIDEO SOFTWARE LTD 1K & 16K

16K SOFTWARE	PRICE
VIDEO-SKETCH (ZX81 only). Move the cursor to any part of the screen. Draw or rub-out as you move it. Mix in text or graphics. Save picture in memory. Save picture sequence on cassette.	7.95
VIDEO-PLAN (ZX81 only). Performs the function of an analysis book. Arithmetic functions include addition, subtraction, multiplication.	7.95
VIDEO-AD. Rotating display of 16 pages of advertising material. Set-up your own pages and change them as and when required.	7.95
VIDEO-GRAPH. Planning and design aid. Create pictures/charts/graphs and store within the program.	£5.95
VIDEO-VIEW. Do it yourself teletext. Create pages of data. Store them within the program.	5.95
VIDEO-MAP (ZX81 only). Educational game based on maps. Navigate your plane to its destination.	5.95
FORCE-FIELD (ZX81 only). Animated bombardment game.	3.95
SPACE-RACE (ZX81 only). Party game for eight players.	3.95
TEST-MATCH. Realistic simulation of a test match series. Dynamic scoreboard.	3.95
FOOTBALL-LEAGUE. Realistic simulation of an entire season. Every match played and results shown with progressive league table.	3.95
STOCK-MARKET (ZX81 only). An exciting game of skill and judgement. Buy and sell stocks and shares as prices change in response to world events.	3.95
PACK 1 – VIDEO-PLAN + VIDEO-AD	17.95
PACK 2 – VIDEO-VIEW + VIDEO-GRAPH	13.95
GAMESET – VIDEO-MAP + 5 GAMES	19.95
1K PARTY TRICKS (ZX81 only)	4.95

NEW VIDEO-INDEX NEW

Q. How do you store 57,000 characters in a 16K RAM.
A. Use VIDEO-INDEX.

Catalogue your cassettes, your record collection, references to magazine articles. Amazingly versatile. Capacity for 1,000 individual references each of which can generate 57 characters of text. Powerful search facilities once the index has been created.

How is it done? The system cannot really fit 57,000 characters in a 16K RAM. It gives the illusion of doing so by eliminating duplicates. For instance in an index of articles in computer magazines about the ZX81 certain words occur time and again. VIDEO-INDEX detects these duplications and thereby conserves space.

What do you get?

1. A machine code master program which sits at the top of RAM. This is initially loaded like a BASIC program.
2. A demonstration index containing 1,000 references to articles about the ZX81 in the popular magazines.
3. A detailed instruction manual.

You may then proceed to create your own catalogues and indexes and save them on cassette.

This program is fast, efficient and ingenious and is by far the best product we have introduced for the ZX81.

N.B. There isn't room for a commentary on this cassette so you'll have to read the manual.

ZX81 or ZX80 + 8K ROM.
ZX printer is useful but not essential. Price: £9.95

Mail order customers add £1 per order.
Allow 28 days for delivery.
Cash with order please to: **VIDEO SOFTWARE LTD.**
Stone Lane, Kinver,
Stourbridge,
West Midlands DY7 6EQ

Personal callers welcome during office hours.

ZX80 Re-Number

A. Beasley

After using the ZX80 for a few months I found that there was a need for a simple renumbering program. In the attempt to solve the problem a BASIC program was written out but this took far too much memory space. While machine code was the obvious solution it did raise yet another problem. How could the program be stored so that it could be used without any trouble. After attempting to store it in a REM line it was found that some of the codes made the system crash when the program was listed.

Solutions

To get over the problem the following method was developed. First all the variables are CLEARed. A string variable is now set up to contain the required number of bytes and the machine code is POKEd into it. As this string variable is the first in the list its location can be found from the two bytes called VARS, see page 122 in the manual. By adding one to the value obtained you have the location of the first character in the string. To call the program you simply find the value of VARS, add one and use this number as a USR call.

More Problems

This method generates its own set of problems, however. If you are using it for program operation you cannot use the following commands, RUN, CLEAR or NEW. By using GOTO you can get over the RUN problem and the others are not really drastic.

The main advantage of this method is that when you save the program you still preserve the string for the next time. It should be noted that the GOTO and GOSUB statements are not altered but you do get everything into 35 bytes.

Operation

To use the program type in with Z$ containing 33 characters. Now run the program then remove it by typing just the line numbers and then 'Newline'. The program you wish to renumber can be keyed in but remember not to use the RUN or CLEAR keys and make sure that the program does not contain Z$.

To activate the renumber type PRINT USR (1+PEEK(16392)+PEEK(16393)*256).

```
1    CLEAR
2    LET Z$ = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
3    LET
     A$ = "06000E0A2128407023713E0A814F30-
     0404  CB70C0237EFE7620FA237ECB7FC018E6"
4    LET A = 1 + PEEK(16392) + PEEK(16393*256)
5    FOR C = 1 TO 33
6    LET B = CODE (A$) - 28
7    LET B = B*16
8    LET A$ = TL$(A$)
9    LET B = B + CODE(A$) - 28
10   LET A$ = TL$(A$)
11   POKE A,B
12   LET A = A + 1
13   NEXT C
```

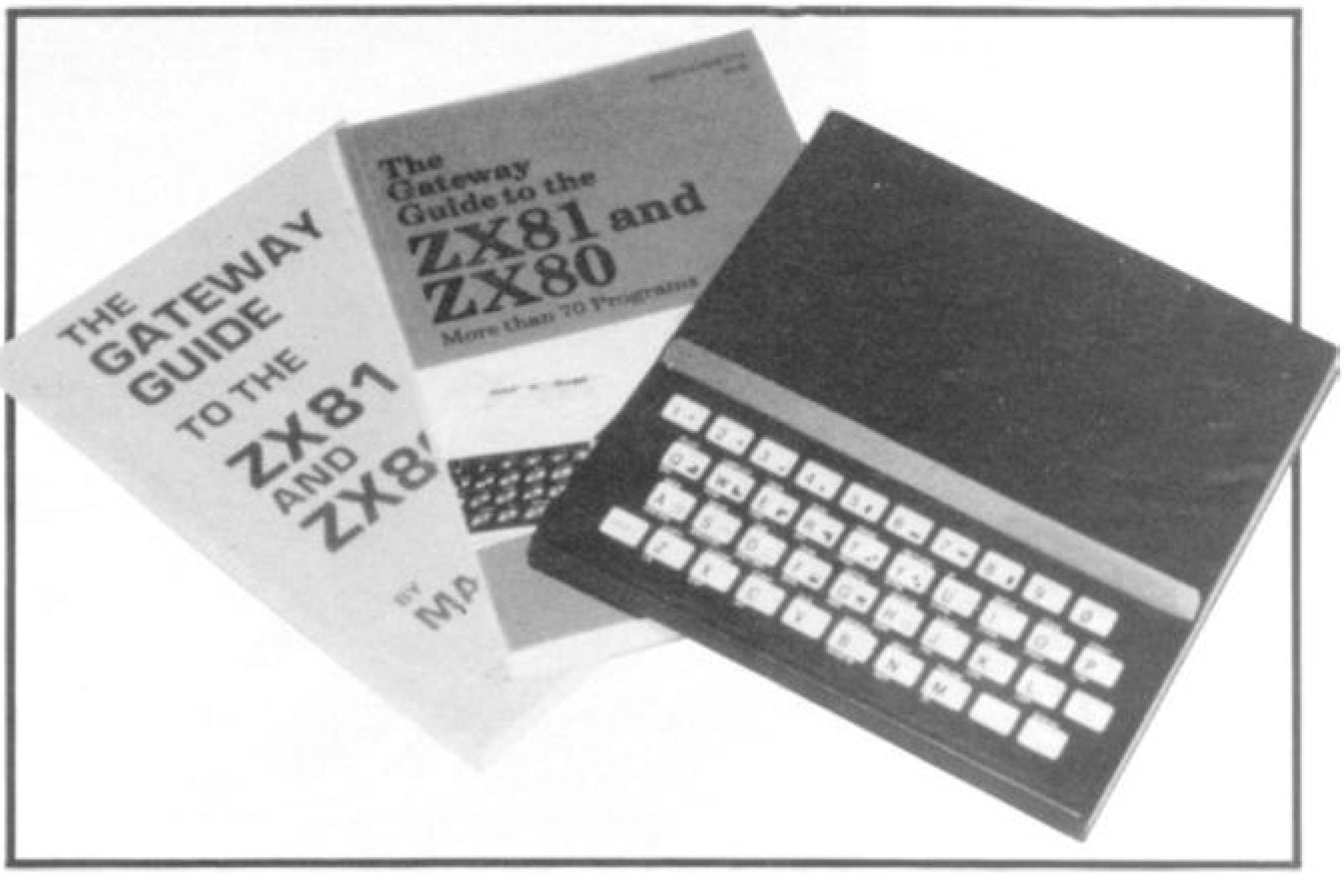
Building a library

After the first mad programming spree with your machine you may like to build yourself a library of useful programs.

The first few weeks after the purchase of your ZX81 may justifiably be defined as the "infatuation" stage. The power of the machine to generate data at apparently phenomenal speed is fascinating, even exciting to those new to the computer keyboard. Scores of little programs are lovingly saved on cassette tapes — most of them centred around the FOR/NEXT loop. Typical programs include printing out "HELLO" 47 times, filling the screen with nine-digit columns of sin(x) and −cos(x) or meaningless equations chosen primarily for their complexity. As many of these little morsels as possible are crammed on both sides of C60 (or in some cases even C120!) tapes. Frantic trips to purchase new supplies of blank cassettes are frequently made or, if the shops are shut, a previously loved recording of Beethoven's ninth is irreverently erased in order to make room for a program which generates the first 2000 primes (I often wonder what you do with primes after you generate them but they seem to offer solace to many).

Naming Names

But all things come to an end at some time or another. It gradually dawns on most people that their "collection" is in reality nothing more than a heap of rubbish. Most of what they have saved is useless, and the few that have some merit are buried between dozens of unwanted remnants.


THE GATEWAY GUIDE TO THE ZX81 AND ZX8
BY MA
The Gateway Guide to the ZX81 and ZX80
More than 70 Programs

```
 10 REM PRIME NUMBERS
 20 DIM Q(2000)
 30 LET Z=1
 35 SCROLL
 40 PRINT 2
 50 LET Q(1)=2
 60 FOR G=3 TO 2000 STEP 2
 70 FOR H=1 TO Z
 80 IF INT (G/Q(H))*Q(H)=G THEN
GOTO 500
100 NEXT H
150 LET Z=Z+1
200 LET Q(Z)=G
240 SCROLL
250 PRINT G
260 PAUSE 50
500 NEXT G
```

Organisation, The Key?

Any attempt to organise your computing life must begin with a simple rule . . . one program on a tape with a copy on the reverse side. Superficially, this appears to be a shocking waste of tape because, on the average, most of the tape will remain unused but in spite of this the rule is sound in human terms. It is better to waste a few feet of relatively inexpensive tape in return for the following benefits: no infuriating searches for programs "in the middle"; no need to name programs, therefore no need to memorise what you have named them; if you have to amend a program, there is no danger of the extra few bytes extending into the obliterating the beginning of the next program; if the tape is accidentally dropped into a plate of soup (or a similar household hazard degrades its performance) only one program is lost; if you lend a tape to a friend for copying purposes and it is returned a corrupted length of jargon, there is less danger of physical violence breaking out if only one program is spoilt.

Finally, we cannot entirely discard a psychological factor. Weeks, perhaps even months of programming work condensed onto one tape fails to impress the casual acquaintance. Spread out into twenty or so, neatly labelled cases with the whole resting in a partitioned "cabinet" will enhance your local reputation as an egghead.

Worthwhile Programs

"Worthwhile" in this sense means "is it worth saving on tape?" Consider the following as a reasonable set of criteria from which to start:

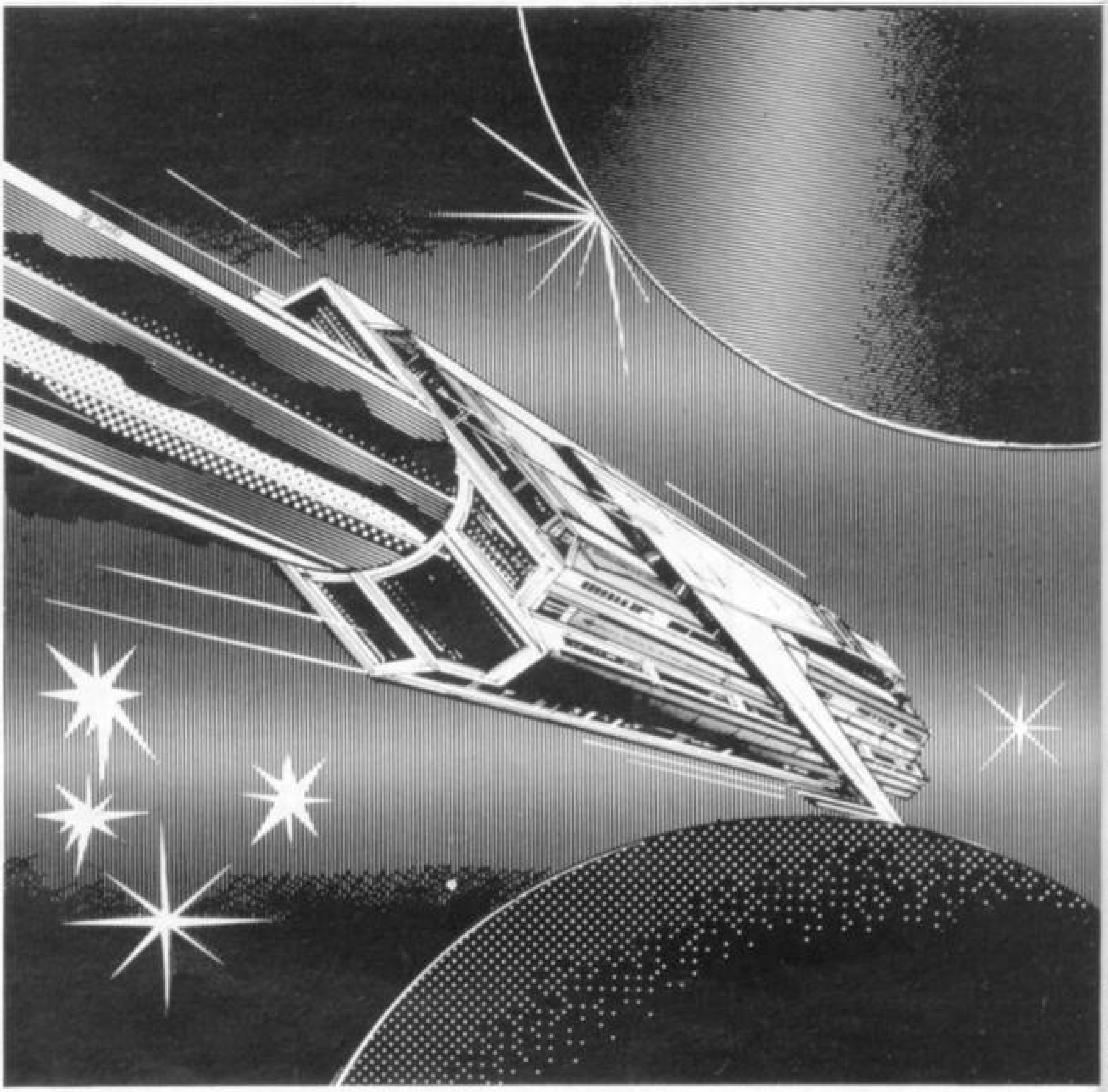
1) Has the program been tested for every conceivable input combination. For example, what happens if you input a "Ø" or a negative number or a number with umpteen digits in it? Nothing is more humiliating to a proud demonstrator than one of those sarcastic error messages which leap up from the bowels of the BASIC interpreter whenever it suffers the slightest confusion. Particularly if you are trying to impress.

2) Will the program check for ridiculous input? Remember that an input can be mathematically acceptable and free from syntax error but can still lack realism. For example, let us assume a program, which assists in the design of a signal amplifier, asks for the supply rail voltage. If the operator mistakenly keys in 2.6E4 instead of 2.6E-4 will the stupid machine accept this . . . or what is more to the point . . . will the stupid program accept it and go on to compute a recommended output current in the order of kiloamps? In short, does the program include full data input validation routines?

3) Is the program completely self-explanatory to the operator? Are there, for instance, full instructions on the VDU screen or does it mean searching for some scrap of paper somewhere which contains the gory details of the button-pressing routines? No accompanying document of any kind should be necessary because the VDU screen can tell all. There should also be a title page which defines clearly the purpose of the program. Remember that at the time of

writing, the purpose is all too clear but after a few weeks or months the memory fades.

4) Is the textual material on the VDU easy to understand and pleasantly arranged? There is no excuse for sloppy presentation and curt chunks of computer jargonese interspersed with abbreviations. Just because the computer has no soul or manners this is no excuse for omitting the pretence. A little care taken in presentation will give the pleasant illusion that lurking behind the cold rectangular sheet of glass is a "being" with a heart of gold . . . kindly and paternal when the occasion warrants it and yet hesitating to deliver streams of pure vitriol if its human operator enters silly figures or presses wrong buttons. In other words, give your computer a personality. Space out the text in a readable manner. Nothing is more tiresome than a page full of closely spaced reading matter, particularly if it is composed entirely of capitals. There is no need to stuff everything on one VDU page but never allow the pages to scroll. Text creeping up from the bottom and disappearing at the top should never be tolerated; it is unpleasant to read and amateurish.



Expansion

5) Is the program planned with the idea of future expansion or improvement in mind? No program can ever be perfect and equally true, no program can ever be absolutely complete. There will always be the nagging doubt, particularly when it is re-run a few weeks later, that some extra facility or twist should have been added. In many cases however, this can be a difficult or even impossible task. In the first case, the program may be utterly incomprehensible when LISTed if several weeks have elapsed since it was written. Juggling with obstinate statements, temper, frustration and the other multitude of ills popular during program construction eventually leads to a transient state of euphoria when the beast finally decides to work. There is a mad rush to "get it on to tape" and indulge in a satisfying bout of self-congratulation.

It takes a little while to appreciate the value of the REM statement because at the time, it seems unnecessary. In fact some of us deliberately leave out remarks in order to prevent other people understanding how our masterpiece works. This attitude can be self-destructive because the writer of the program may eventually become the victim. Another obstacle to future amendment is a poorly structured original and close-packed line numbers. Never start a program with line number less than 100 in case some extra stuff may have to be squeezed in at the head. Be methodical in the choice of subroutine line numbers. Stick them all together well down the bottom, say at line 9,000 onwards. In this way, you will avoid the ugly embarrassment of having to leap frog over them with a wasted GOTO statement when the lines start to creep down further than the original estimate allowed.

The term "program structure" of course means a lot more than the mere organisation of line numbers. It means laying out a program in neat little modules, each capable of being individually tested in its own right. In fact there is a specific programming philosophy with many little rules and regulations resting beneath the blanket title of "STRUCTURED PROGRAMMING". This is worth detailed study if only to know when to break some of the rules.

Programs To Write

Advice on what programs to write is about as difficult as advising on the best length for a piece of string. An overall piece of advice is simply to walk before you run. Don't attempt to write wildly ambitious programs unless you are quite certain you understand the full implications of the task ahead. Unfortunately, it takes some experience to know in advance whether or not a certain programming task is likely to be easy or horribly difficult; computers are odd things.

For example, if someone came and asked me to write a program to print out a table of the singular solutions of a second order differential equation I would take the money in advance and probably deliver the goods (suitably tarted up in accordance with the previous advice) the next day. This is not because maths and physics is my strong point (I might pass O-level maths with difficulty) but because the actual maths details must reside in some text book equation somewhere or other. It would just be a case of letting the faithful old BASIC interpreter handle the sordid details once the correct sequence of brackets and operators have been entered from the text book to the VDU.

Such programs are elementary number crunching exercises, impressive but routine. On the other hand, a request for "a little program to sort and classify my butterfly collection'' could turn out to be a nightmare. The following is a crude attempt to group the classes of programs which can be written and appropriate remarks on their respective difficulty factors.

Numbercrunching

These follow a relatively simple pattern; inputting the required parameters, fitting them into the ''equation line'' and displaying the results in a clear manner.

Two subroutines should be considered almost indispensable to number-crunching activities, one to round off numerical results to a desirable number of decimal places and the other to line up the decimal points. Answers like 34.5689302 inches or £67.24578945 lack realism and the sight of a VDU screen full of figure groups zigzagging from top to bottom is not only difficult to read, it is quite revolting in appearance. Always use TAB(n) to position columns, the semicolon as a delimiter encourages zigzagging.

Quizzes

Many sophisticated programs have been written under the general title of Computer-Aided-Teaching or Computer-Aided-Learning. Less ambitious but surprisingly useful programs are relatively easy to write (and certainly worth saving) based on questions and answers.

An extra twist is to incorporate random selection of the pairs to stop the operator using a sequence. There is, however, an element of danger in this type of program. It tends to breed quiz addicts. Tape after tape is saved on all possible subjects until the entire household takes on the appearance of a Bamber Gascoigne Show.

Games

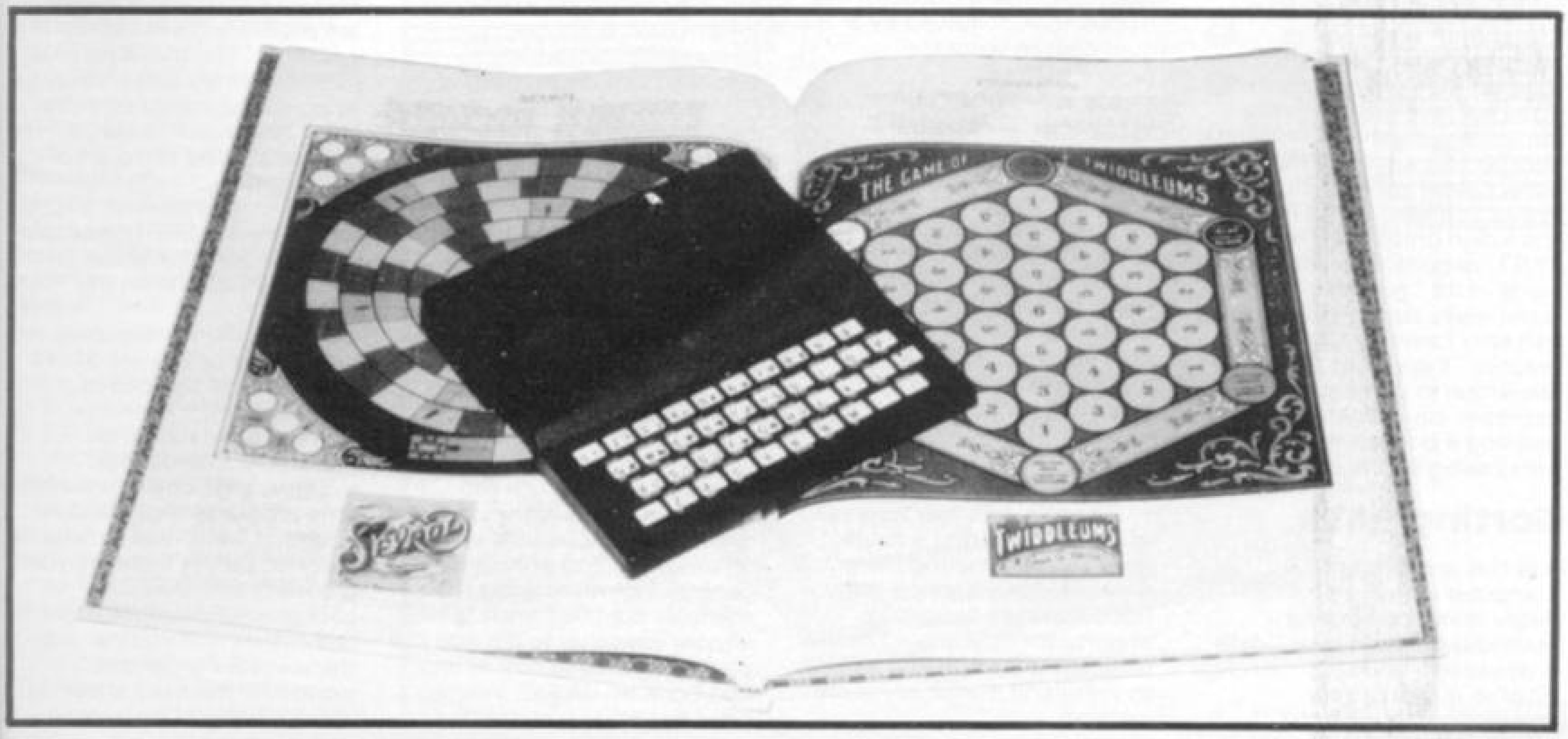
This area is undoubtedly popular and it cannot be denied that senior programmers in the professional classes devote many hours to thinking up new games or introducing new twists to existing ones. Unfortunately, a game program, unless particularly novel and interspersed with exciting animation takes a disproportionate time to program in relation to the subsequent playing time. As programming exercises they are superb. Whether many of them are really worth the tape storage is debatable. Consider for example the class of games which could be covered by the classification ''Moon Landing''. They all follow the same well-worn path . . . you are in some dangerous James Kirk situation . . . too much throttle and you run out of something or other . . . too little and you crack the surface of the moon or Mars or whatever particular member of the galactic regions happens to fit the title. They will all contain a couple of equations from the Newtonian tables, suitably embellished to fit the game. The most awkward thing to get right in programming such a game is the difficulty-factor. Too hard and the player is frustrated; too easy and the game is described as boring.

Enthusiasm

The behaviour pattern of the players, however much care is taken with the programming details, is distressingly familiar. Great enthusiasm at first but declining exponentially towards complete apathy. For those who have a genuine love for game programming the following little tips may be found useful:

Explain the rules concisely in the title page.

Display as much animation



as your skills in programming allow.

Don't allow the computer to respond "too instantaneously". An apparent immediate response does not impress the player.

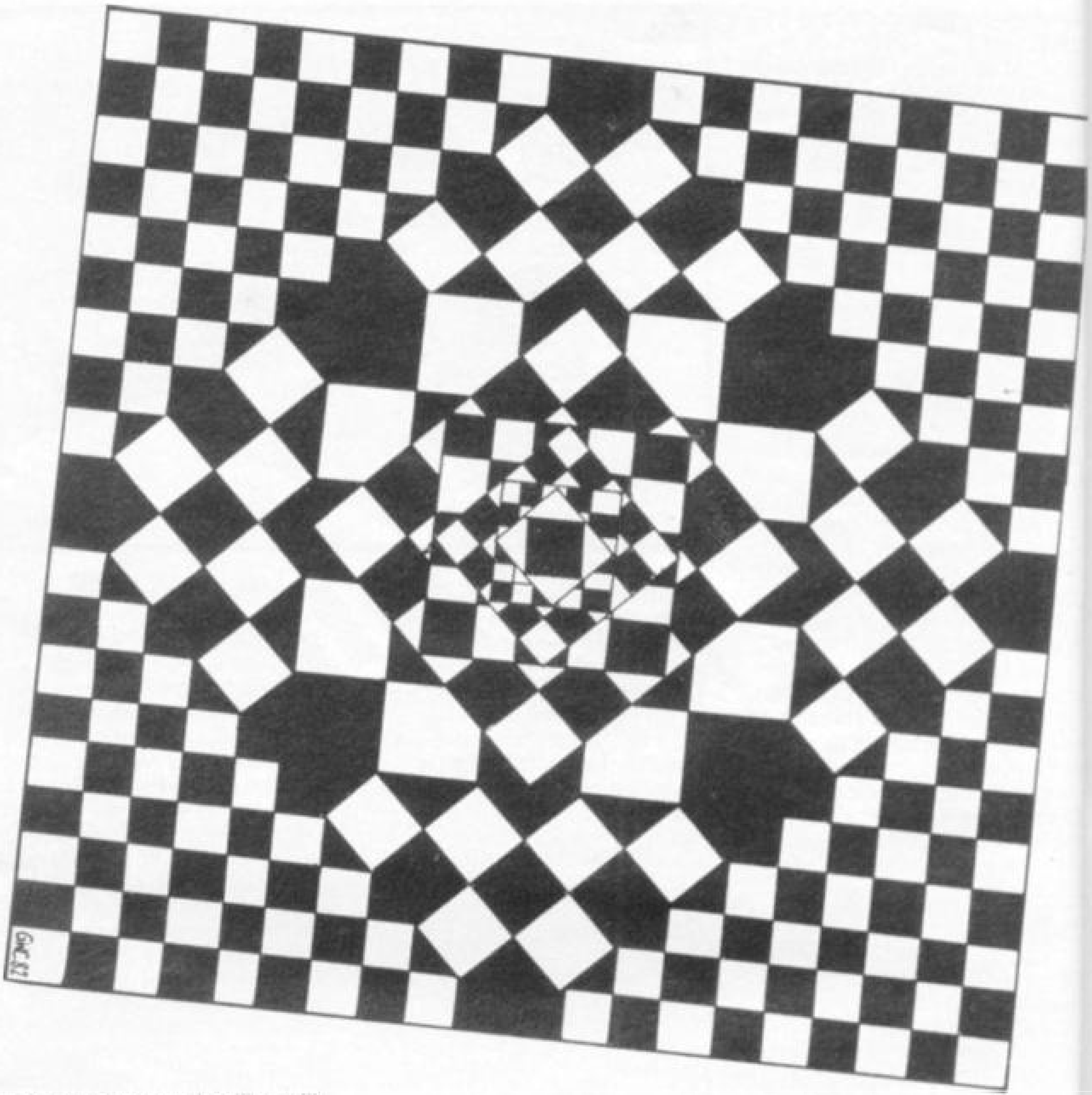
Choose your GO-BACK-TO destinations carefully. It is pleasant for the ego when the computer asks for your name and instantly promotes you to "CAPTAIN . . ." but it soon becomes an irritating chore if this ritual has to be repeated on each replay.

Take particular care to make programs crash-proof. There are some who, finding themselves in an irretrievable position, would crash the program rather than suffer the humilation of being beaten by "some damned machine".

Try and add a few original twists. For example, allow a few loop holes for cheating but make the computer respond with something like,

"We noticed your pathetic attempt at subterfuge three lines ago, but in view of your obvious immaturity, we decided to overlook the matter. Should it occur again you will be disqualified."

Note the use of the royal "WE" above . . . very useful little dodge to create an air of omnipotence, although don't overdo it by using phrases like "My RAMS and I . . . . . ";



Dynamic Art

Providing the world "art" is not taken too literally, some quite astonising moving patterns can be generated on most of the home computers. They are however far more impressive if you are fortunate enough to own an APPLE or other model which includes colour combined with high-resolution graphics. The ZX81, despite the great play made of its "graphics facility" is not really suited to the job. It certainly has very useful graphic "keys" but the resolution in general is pathetic; equivalent to painting a portrait with a ten inch ceiling brush.

Sorting DATA

It is this area that the computer is truly at home. Every home computing enthusiast should take "data processing" seriously. Strange how so many writers attempting to teach this subject use examples like milk bills to start off with. Milk is of course a delightful source of health giving energy but the compilation of milk bills is not likely to cause a flutter of excitement, followed by a mad rush to write the program.

My wife would look at me in sheer astonisment if I suggested she used my ZX81 each month. She would probably write it out on the back of an envelope in ten seconds flat, certainly before I would have time to fumble round for the ON/OFF switch. It is appreciated of course that such simple examples are typical weapons of the educationalists, based on the principle "teach from the know to the unknown", "use homely analogies" etc etc.

There is a danger however of de-glamourising a subject and underestimating the public mood and intelligence. Why not substitute plutonium imports for milk bills? The program would be just as easy to write and marginally more exciting.

Tape books

Sales brouchures often draw attention to advantages of storing useful day to day information on home computers, recipes etc. General purpose reference "books" can certainly be very useful on tape, providing there is a title selection page or pages. Once the tape is loaded (the most annoying stage), it is quicker to get at a given page by pressing a number key than turning the pages of a paper book.

Programming

It is difficult to say anything original on this subject. Literally hundreds of books have been written on the BASIC language alone, besides the thousands written on programming principles in general. However good the manuals supplied are it is almost essential to dip into the pocket again and buy at least one book on BASIC. Which one? For what it is worth, I have been impressed (and educated) by "BASIC AND THE PERSONAL COMPUTER" by Thomas A. Dwyer and Margot Critchfield but there are probably dozens of equally as useful. The following little snippets of wisdom (?) may be of some assistance to those who, like myself, have no *natural* abilities in the art of programming.

1) Buy a good book on BASIC and carry out *EVERY* example of it. It's not a bit of use just "reading" a book on this subject.

2) Buy as many magazines on computing as you can afford in addition to this one of course.

3) Keep a notebook, or preferably a card index system, and copy down every little programming "module" or dodge which has general purpose use. In this way you gradually acquire a background in fundamental techniques and you can slip them in your programs whenever the need arises. Is this cheating? Depends on

how you define cheating. There is little point in reinventing the wheel on every possible occasion.

Issac Newton, not renowned for his modesty, once repled to a remark by an admirer, "If I have seen a little further than most, it is because I have stood on the shoulders of giants." To copy down a complete program and pass it off as one of your own is of course a different matter. Ethics apart (not particularly fashionable nowadays anyway) some one else might have read the same magazine and bang goes your reputation! The sort of modules worth saving for future and continous use include, lining up decimal points, rounding to n significant digits, sorting numbers into ascending or descending order, sorting names into alphabetic order, etc etc.

A word of warning regarding program modules or indeed full programs printed in magazines. Some of them don't work! The usual cause is a misprint some where along the line and readers, to judge from the rather acidic tone of their letters, express surprise that "the Editor doesn't proof read them before printing". Proof reading costs time and but to proof read computer programs to guarantee 100% error free would probably treble the cost of a magazine. In any case, if they don't work then make them work . . . it's good practice anyway and the mistake is often the trival omission or incorrect insertion of a comma or quote or perhaps an unmatched parenthesis.

Join a local computer club. They tend to be friendly gatherings all anxious to learn from each other and refreshing free from professional snobbery of any kind. The home computer addict tends to be thought of as slightly weird by "normal" people, a kind of mutation. It is comforting to spend a few hours in the evening with other mutants. The great thing is to join soon while the hobby is still young.

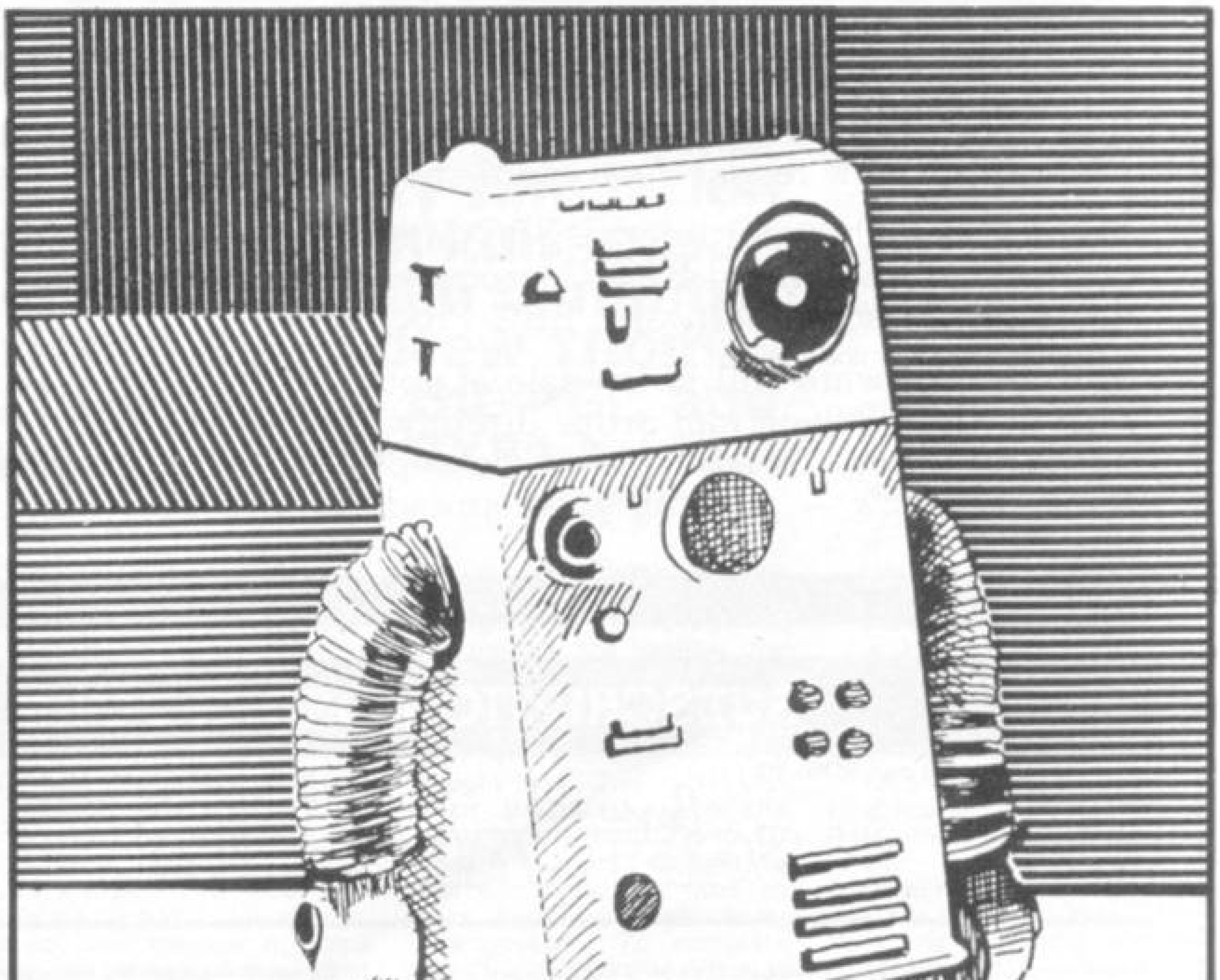
As the numbers of these clubs grow and the membership expands to excessive limits, the character may change. It could reach a state like that which exists in the so-called "exclusive" golf clubs, questionnaires on various aspects of the applicants background. Perhaps, God forbid, they may even require that supreme emblem of respectability the club tie!

The Final Words

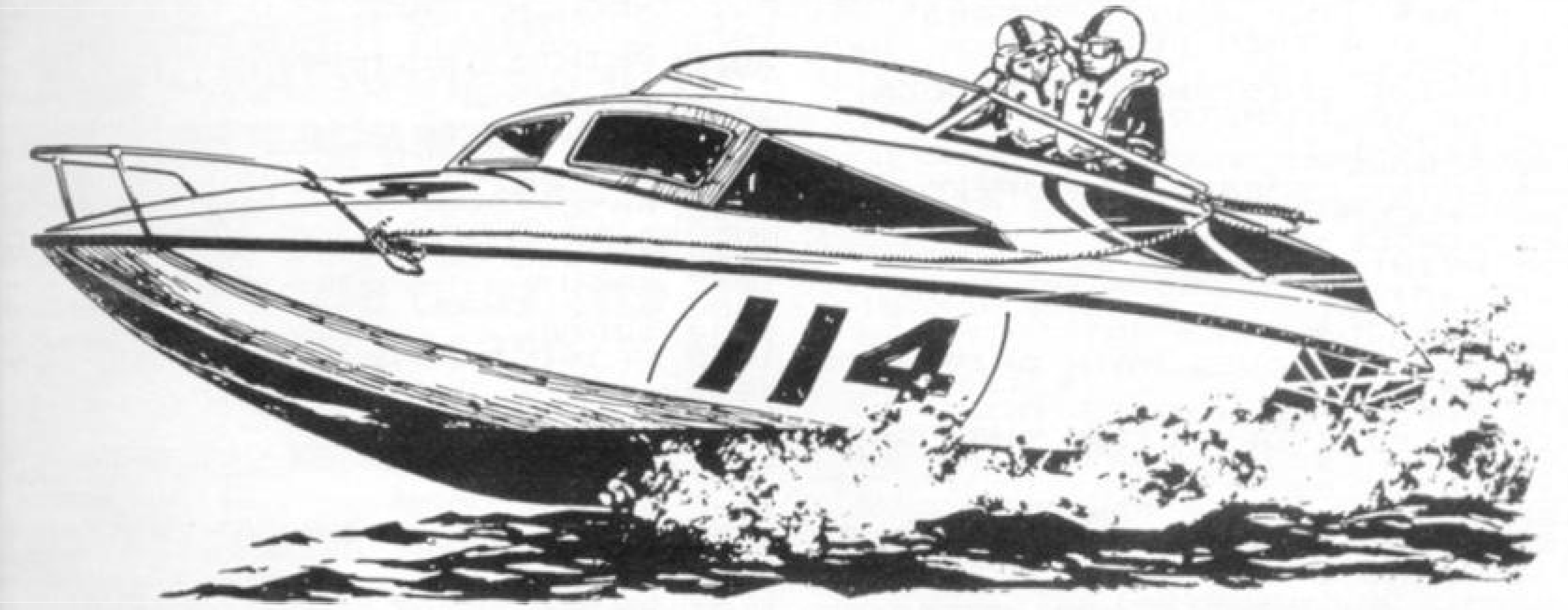
In conclusion, it is worth examining some advice given in the manuals concerning the art of programming. Apparently, it is a cardinal sin to compose at the keyboard . . . . it is called "winging it". We are instructed by the tribe elders to write the complete program on paper before approaching the keyboard; at least every separate module. This disipline came into being because of two non-related influences. Firstly, the influence of the academic purists who insist on a carefully thought out logical approach on paper first. The second influence was that of practical necessity. Prior to the micro-processor and high density integration of semiconductor memory, computing was very expensive, VDUs were non-existent or rare, every response was spewed out on reams of expensive paper and, above all, the cost per minute precluded the luxury of idle doodling.

The position with the home computer is different. Very few of us can afford printers anyway . . . at least not in the first year of ownership. The VDU wastes nothing. It is a perfect doodling pad and unlike paper, can be used over and over again. It is, however, a good idea to draw out a rough plan of campaign in the form of an outline flowchart, prior to operating the keys.

Another discipline carried over from the past is an obsession with memory economy. It seems pointless to prune a program (that works) down to the last byte unless there is a real danger of running out of memory. If you have say, a 16K memory and your unpruned program takes 6K why fiddle about with it. To increase execution speed just for the sake of it is another pointless operation. If your program works and it is reasonably "tidy" leave it alone and get on with another. In this way your tape library will grow much quicker and be just as useful as those of your fusspot colleagues.



Power Boat



If you're desperate to try out your sea legs, this little speedy simulation of a power boat race on Ruislip Lido should start the adrenalin pumping. It was written by Andrew Thomas and Gareth Callister for the PET, and converted for the ZX81 by Tim Hartnell.

You're in control of an extremely powerful, 16K power boat, with a state-of-the-art on-board computer (looking somewhat like a small black thing from Cambridge), racing around the Lido against other top champs. The course and weather conditions vary in each game, although the weather doesn't alter during the course of a game (the reliability of the climate in West Ruislip is well-known). The course is marked out with buoys to give you the direction and maximum speed for that section. There are 20 sections in all. Even finishing the race is an achievement, as you'll see when you run the program.

You must enter the amount of RUDDER you need to make a turn. Too much and you'll cut the corner, too little and you'll overshoot the corner. Slightly cutting the corner, if you can manage it, helps you get ahead of the other drivers. However, bad driving on the corners will get you disqualified. The lower the number entered for the rudder, the smaller the turning circle and hence the smaller the radius. Entering zero will make your craft continue in a straight line. The ratio between the rudder and the radius varies from one to 10 when stationary, to one to five at top speed, just to make a difficult game nearly impossible. To complete the course, you must also regulate your speed to a value near to, or slightly above, the maximum speed for the section.

The more power you supply to the drive, the faster the boat will go. If you enter too much power, you run the risk of a most dramatic flip-over.

Variables used in this program are:

C is the difference between actual and rated turning circles.
D is the distance to the leader.
E is the previous 'power to the main drive'.
E1 is the present 'power to the main drive'.
F is the amount of fuel remaining.
M is the maximum speed for the section.
P is your position.
R is the rated turning circle.
S is your present speed.
T is the rudder position.
W is the prevailing weather condition.
X is the section number.

```
  10 REM *POWER BOAT*
  20 REM *ANDREW THOMAS*
  30 REM *GARETH CALLISTER*
  40 REM *TIM HARTNELL*
  90 LET E=50
 100 DIM A$(6,10)
 110 LET A$(1)="CALM"
 120 LET A$(2)="MILD"
 130 LET A$(3)="CHOPPY"
 140 LET A$(4)="ROUGH"
 150 LET A$(5)="VERY ROUGH"
 160 LET A$(6)="STORMY"
 230 LET W=INT (RND*6)+1
 235 LET S=0
 237 SCROLL
 240 PRINT "CONDITIONS ARE ";A$(
W)
 250 LET F=1000
 260 LET P=5
 270 LET D=10
 280 LET E=0
 300 FOR X=1 TO 20
 305 SCROLL
 310 IF RND<0.3 THEN GOTO 320
 311 LET R=0
 312 LET M=55+INT (RND*55)-W*2
 315 GOTO 340
 320 LET R=INT (RND*90)+10
 330 LET M=INT (30+RND*20+R/3)-W
*2
```

```
 340 SCROLL
 345 PRINT "SECTION ";X
 347 SCROLL
 350 PRINT "POSITION ";P
 352 SCROLL
 355 IF D<0 THEN PRINT "YOU LEAD
 BY ";ABS (INT (D));" METRES"
 356 IF D<0 THEN GOTO 365
 357 SCROLL
 360 PRINT "DISTANCE TO LEADER I
S ";INT (D);" METRES"
 365 SCROLL
 370 SCROLL
 380 PRINT "SPEED  MAX SPEED  RA
DIUS  FUEL"
 385 SCROLL
 390 PRINT "  ";INT (S);"
";INT (M);"           ";INT (R);"
   ";INT (F)
 400 SCROLL
 410 PRINT "POWER MAIN DRIVE? (1
 TO 100)";
 415 INPUT E1
 416 IF E1<1 OR E1>100 THEN GOTO
 415
 417 PRINT E1
 420 SCROLL
 425 PRINT "RUDDER POSITION? (0
TO 9) ";
 426 INPUT T
 437 IF T<0 OR T>9 THEN GOTO 426
 438 PRINT T
 445 SCROLL
 450 IF E1-E>50-W*2 THEN GOTO 10
00
 460 LET C=(S/100+1)*10*T-R
 470 IF ABS (C)>18-W THEN GOTO 1
020
 480 IF C<6 THEN GOTO 510
 485 SCROLL
 490 PRINT "YOU TOOK CORNER FAR
TOO WIDE"
 500 LET C=12
 510 IF C>-6 THEN GOTO 540
 520 SCROLL
 525 PRINT "YOU CUT CORNER TOO C
LOSE"
 530 LET C=18
 540 LET E=E1
 550 LET S=S/3+E/1.2-(T/10)
 560 IF S>1.1*M+10-W THEN GOTO 1
080
 570 LET D=D+M-S+C/2
 580 LET P=INT (D/10+RND*2+2)
 585 IF P<2 THEN LET P=2
 590 IF D<=0 THEN LET P=1
 600 LET F=F-((E/10)**2+E)/2
 610 IF F<0 THEN GOTO 1000
 620 NEXT X
 630 IF P>1 THEN GOTO 670
 635 SCROLL
 640 PRINT "CONGRATULATIONS - YO
U WON"
 660 GOTO 1200
 670 IF D>30 THEN GOTO 710
 675 SCROLL
 680 PRINT "YOU FINISHED...AND N
OT TOO"
 681 SCROLL
 682 PRINT "FAR BEHIND THE LEADE
R"
 690 GOTO 730
 700 SCROLL
 710 SCROLL
 715 PRINT "WELL, AT LEAST YOU F
INISHED"
 730 SCROLL
 740 PRINT "YOU FINISHED ";P
 745 SCROLL
 750 PRINT "DISTANCE TO LEADER W
AS ";INT (D);" M"
 760 GOTO 1200
1000 SCROLL
1001 PRINT "YOUR BOAT COLLAPSED
FROM"
1002 SCROLL
1003 PRINT "TOO MUCH ACCELERATIO
N"
1010 GOTO 1200
1020 SCROLL
1025 PRINT "CRASH!!, YOUR BOAT W
ENT THROUGH"
1026 SCROLL
1027 PRINT "THE MARKERS. YOU HAV
E BEEN"
1028 SCROLL
1029 PRINT ,"DISQUALIFIED"
1030 IF C>0 THEN GOTO 1060
1040 SCROLL
1045 PRINT "YOU SHOULDNT TAKE CO
RNERS"
1046 SCROLL
1047 PRINT ,"SO WIDE"
1050 GOTO 1200
1060 SCROLL
1065 PRINT "YOU SHOULDNT CUT COR
NERS"
1070 GOTO 1200
1080 SCROLL
1085 PRINT "CRACK!!, YOU DAMAGED
 THE BOAT"
1086 SCROLL
1087 PRINT "BY GOING SO FAST"
1090 GOTO 1200
1100 SCROLL
1101 PRINT "PHUT...OUT OF FUEL"
1200 SCROLL
1210 PRINT "ANOTHER GAME?"
1220 INPUT U$
1225 CLS
1230 IF CODE (U$)<>CODE "N" THEN
 RUN
1240 SCROLL
1250 PRINT "OK,BYE"
```

```
CONDITIONS ARE STORMY

SECTION 1
POSITION 5

DISTANCE TO LEADER IS 10 METRES

SPEED  MAX SPEED  RADIUS  FUEL
  0       50        0     1000
POWER MAIN DRIVE? (1 TO 100)4
RUDDER POSITION? (0 TO 9) 0
SECTION 2
POSITION 7

DISTANCE TO LEADER IS 56 METRES

SPEED  MAX SPEED  RADIUS  FUEL
  3       74        0      997
POWER MAIN DRIVE? (1 TO 100)5
RUDDER POSITION? (0 TO 9) 0

SECTION 3
POSITION 15
DISTANCE TO LEADER IS 125 METRE
SPEED  MAX SPEED  RADIUS  FUEL
  5       62        0      995
POWER MAIN DRIVE? (1 TO 100)7
RUDDER POSITION? (0 TO 9) 0

SECTION 4
POSITION 20
DISTANCE TO LEADER IS 179 METRE
SPEED  MAX SPEED  RADIUS  FUEL
  7       59        69     991
POWER MAIN DRIVE? (1 TO 100)7
RUDDER POSITION? (0 TO 9) 5
CRASH!!, YOUR BOAT WENT THROUGH
THE MARKERS. YOU HAVE BEEN
                DISQUALIFIED
YOU SHOULDNT TAKE CORNERS
                SO WIDE
ANOTHER GAME?
```

Zap

Paul Gausden from Kingswood brings a little machine code into play to put you in the cockpit of a biplane engaged in a deadly battle over the Straits of Mallacash.

Paul writes to *ZX Computing*: "A short while ago I was writing a 'space war' type of program with a starry sky, and moving the space ships by printing black squares over them which meant that as the game progressed all the stars disappeared.

"I was trying to find a way around this when a friend lent me his "Programming the Z80", by Rodnay Zaks. This helped me to write the following machine code program:

```
ld hl,7530h (30000)
ld de, (d-file)
inc de
ld bc, 02D6 (726)
LDIR
Ret
```

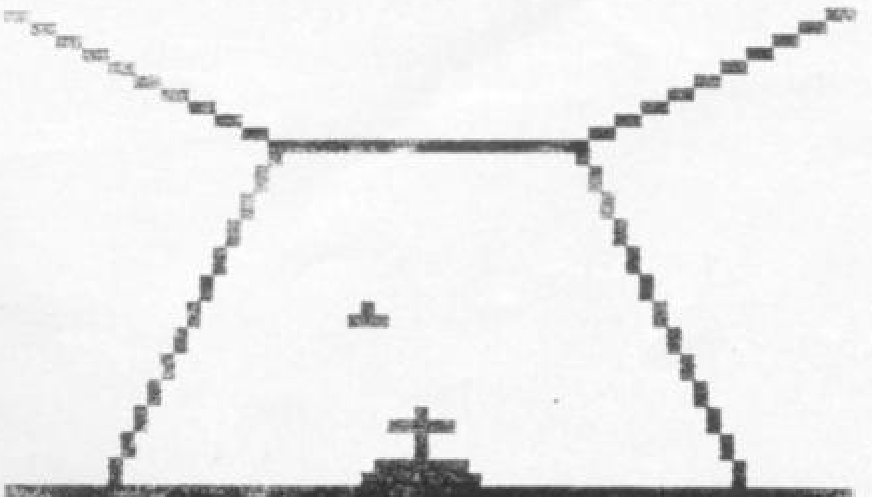
"This program loads what is above memory location 30000 into the display, and by changing over the de and hl registers it saves the display above 30000. This meant that when I printed out my sky I used one routine to save it, then used the other as a type of CLS, but recalling the same background each time, so now the stars stayed where they were!

"One of the programs I've written using this routine is ZAP. It uses the routines to save the foreground (the interior of the cockpit) and *recalls* it each time. You are chasing after an enemy aircraft and you have to remember the controls work in reverse.

"To get the GOSUBs in the REM statements, type THEN GOSUB and rub out the THEN."

As you can see from the screen printout, the enemy plane is the little thing you can see through the cockpit. It moves randomly around, and you must use 5, 6, 7 and 8 to move yourself into a position where the enemy plane will be on the cross of your sight. As Paul pointed out, you have to remember that you are, in effect, moving backwards, swinging the view of the enemy against the sky. This may sound complicated, but it becomes very clear once you RUN the program, and gives a good impression of a dogfight.

```
   1 REM 5K? GOSUB ?£RND(▀CHR$ ▀
 GOSUB ▒TAN
   2 REM E£RND)K?▀CHR$ ▀7 GOSUB
▒TAN
ENTER THE FOLLOWING AS
DIRECT COMMANDS:

POKE 16516,117
POKE 16518,91
POKE 16539,117

   1 REM 5K? GOSUB ?£RND(▀CHR$ ▀
 GOSUB ▒TAN
   2 REM E£RND)K?▀CHR$ ▀7 GOSUB
▒TAN
   3 LET SCR=16514
   4 LET CPY=16534
  10 FOR F=0 TO 4
  20 PRINT TAB F*2;"▀▄";TAB 30-F
*2;"▄▀"
  30 NEXT F
  40 PRINT TAB 10;"▛▀▀▀▀▀▀▀▀▀▀▜"
  50 FOR F=0 TO 5
  60 PRINT TAB 9-F;"▐";TAB 22+F;
"▌"
  70 PRINT TAB 9-F;"▐";TAB 22+F;
"▌"
  80 NEXT F
  90 FOR F=0 TO 3
 100 PRINT "▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
▒▒▒▒▒▒▒▒▒▒▒▒"
 110 NEXT F
 120 PRINT AT 15,14;"▄█▄";AT 16,
15;"▌";AT 17,13;"▄▄▄▄▄"
 130 LET L=USR CPY
 140 LET S=45
 150 LET K=0
 160 LET X=INT (RND*32)
 170 LET Y=INT (RND*18)
 160 LET P=USR SCR
 190 PRINT AT Y,X;"▄█ "
 195 IF S=0 THEN STOP
 200 IF INKEY$="0" THEN GOTO 300
 210 LET X=X+INT (RND*3)-1+(INKE
Y$="5")-(INKEY$="8")
 220 LET X=X+2*((X<0)-(X>30))
 230 LET Y=Y+INT (RND*3)-1+(INKE
Y$="7")-(INKEY$="6")
 240 LET Y=Y+2*((Y<0)-(Y>17))
 250 GOTO 180
 300 PRINT AT 15,14;"▒▒▒"
 310 LET S=S-1
 320 IF X<17 AND X>13 AND Y=14 T
HEN GOTO 340
 330 GOTO 210
 340 LET K=K+1
 350 POKE 30643,K+156
 360 GOTO 160
```

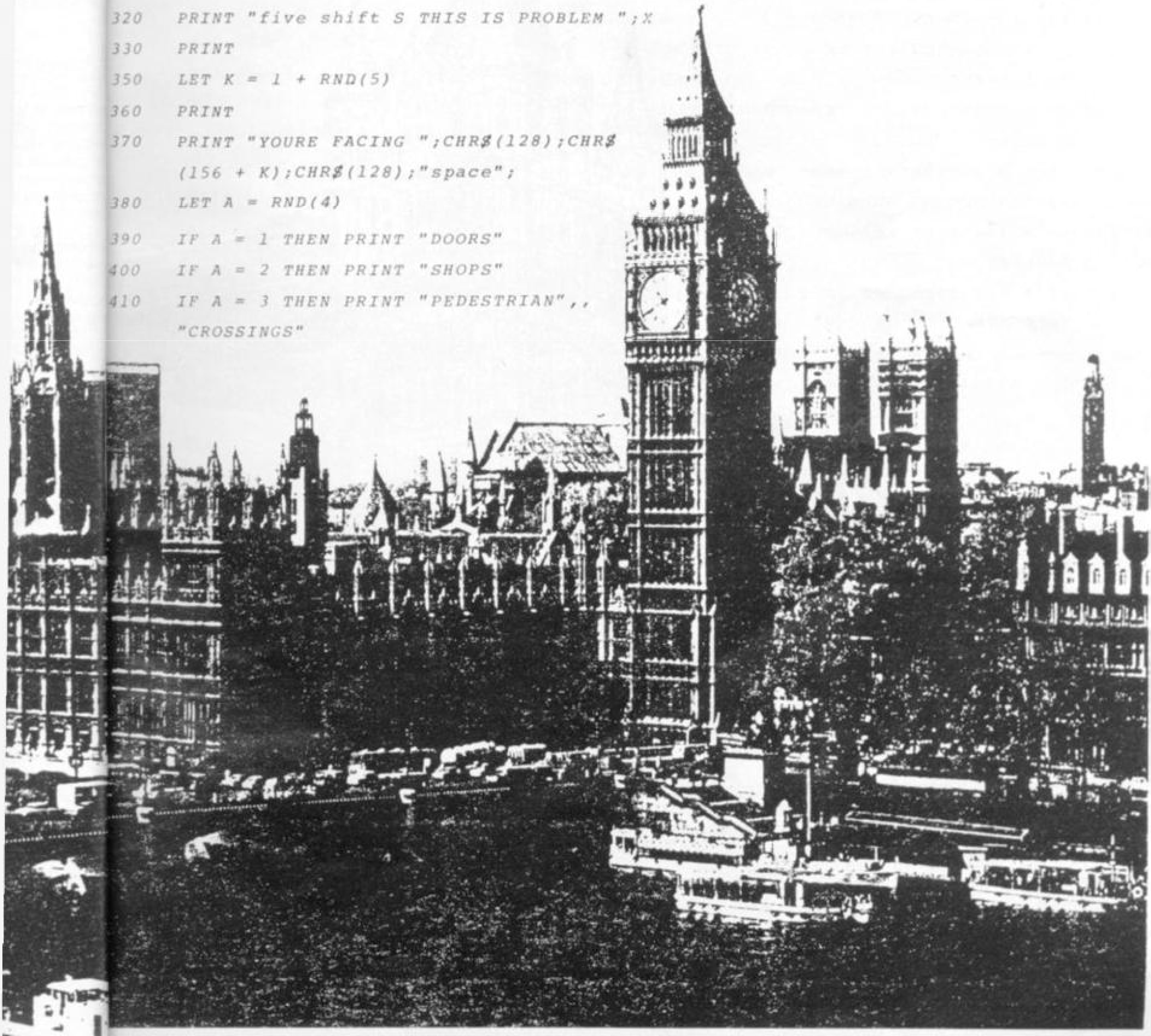


TOURIST TRAP

TOURIST TRAP is a 4K ZX80 "adventure" type program which can be easily converted (essentially by changing the way the random numbers are generated) for the ZX81. It is very, very light-hearted but is fun to play, and gives you a framework upon which to construct your own ADVENTURE. You might like to try changing this program slightly so that it asks for, and uses, your name from time to time.

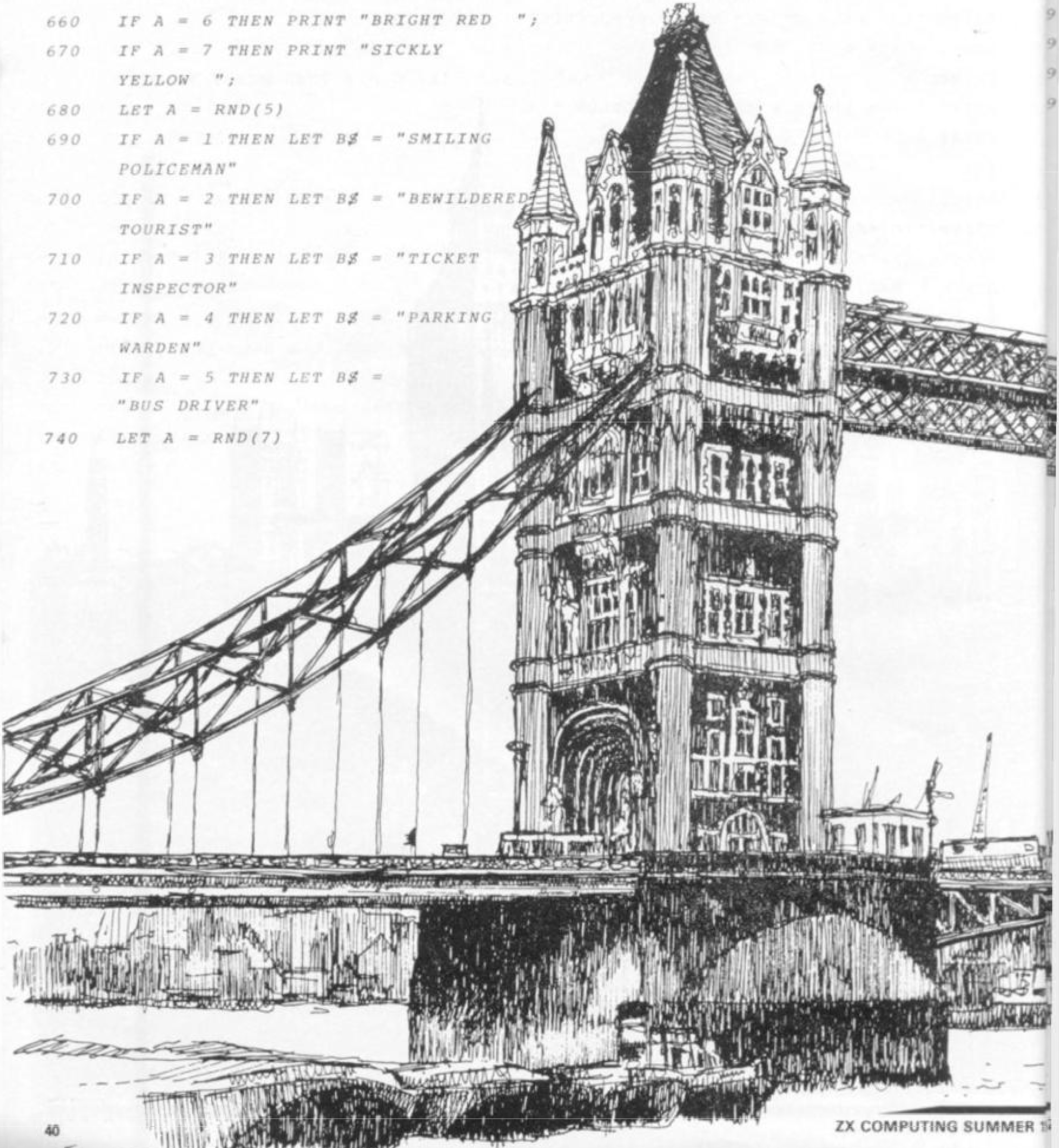
```
1      RANDOMISE
10     PRINT ,"TOURIST TRAP"
20     PRINT ,"12 shift G"
30     GOSUB 1630
50     LET X = 0
60     LET  S = 30
70     LET W = 1
80     PRINT "HERE WE GO ON A TRIP THROUGH"
90     PRINT ,"LONDON TOWN..."
100    PRINT
110    PRINT "YOU HAVE $30, AND YOURE
       TRYING"
120    PRINT "TO GET TO BUCKINGHAM PALACE"
130    PRINT "(THE PALACE IS ON FOOTPATH 10)"
135    PRINT
140    PRINT "IF YOU GET THERE YOULL NEED
150    PRINT "   $35 TO BRIBE YOUR WAY IN
160    PRINT
170    PRINT ,,"PRESS NEWLINE"
180    INPUT A$
190    IF NOT A$ = "" THEN STOP
200    GOSUB 1610
```

```
220 IF W < 1 AND RND(10) > 8 THEN GOTO 1730
225 IF W < 1 THEN LET W = 1
230 PRINT "twospaceTHIS IS FOOTPATH ";W
250 IF W = 10 THEN GOTO 1690
270 PRINT "SO YOU ARE ";CHR$(128);CHR$
    (166 - W);CHR$(128);" FROM THE PALACE"
280 PRINT
285 IF S < 1 THEN LET S = RND(5)
290 PRINT "YOU HAVE $";S;" IN YOUR POCKETS"
300 LET X = X + 1
310 PRINT
320 PRINT "five shift S THIS IS PROBLEM ";X
330 PRINT
350 LET K = 1 + RND(5)
360 PRINT
370 PRINT "YOURE FACING ";CHR$(128);CHR$
    (156 + K);CHR$(128);"space";
380 LET A = RND(4)
390 IF A = 1 THEN PRINT "DOORS"
400 IF A = 2 THEN PRINT "SHOPS"
410 IF A = 3 THEN PRINT "PEDESTRIAN",,
    "CROSSINGS"
420 IF A = 4 THEN PRINT "TUBE STATION",,
    "ENTRANCES"
450 PRINT
460 PRINT "WHICH ONE WILL YOU USE?"
470 INPUT B
475 IF B > K OR B < 1 THEN GOTO 470
480 GOSUB 1610
500 IF RND(10) < 4 OR B = A THEN GOSUB
    1170
530 LET C = RND(4)
540 IF C = 1 THEN PRINT "FOOL";
```



```
550    IF C = 2 THEN PRINT "IDIOT";
560    IF C = 3 THEN PRINT "HELP";
570    IF C = 4 THEN PRINT "SURPRISE";
620    PRINT "YOURE FACE TO FACE",,
       "WITH A ";
630    LET A = RND(7)
640    IF A = 4 THEN PRINT "GREEN  ";
650    IF A = 5 THEN PRINT "BLUE  ";
660    IF A = 6 THEN PRINT "BRIGHT RED  ";
670    IF A = 7 THEN PRINT "SICKLY
       YELLOW  ";
680    LET A = RND(5)
690    IF A = 1 THEN LET B$ = "SMILING
       POLICEMAN"
700    IF A = 2 THEN LET B$ = "BEWILDERED
       TOURIST"
710    IF A = 3 THEN LET B$ = "TICKET
       INSPECTOR"
720    IF A = 4 THEN LET B$ = "PARKING
       WARDEN"
730    IF A = 5 THEN LET B$ =
       "BUS DRIVER"
740    LET A = RND(7)
```

```
750    IF A = 1 THEN LET C$ = "PARKING
       TICKET"
810    IF A = 2 THEN LET C$ = "PASS TO TH
       TOWER"
820    IF A = 3 THEN LET C$ = "POSTCARD O
       TRAFALGAR SQUARE"
830    IF A = 4 THEN LET C$ = "MAP OF
       CAMBRIDGE"
```



```
840   IF A = 5 THEN LET C$ = "MONKEY ON
      A STICK"
850   IF A = 6 THEN LET C$ = "PACK OF FISH
      AND CHIPS"
860   IF A = 7 THEN LET C$ = "RINGSIDE SEAT
      AT OXFORD CIRCUS"
900   PRINT B$,"WITH A"
910   PRINT C$
920   PRINT
930   PRINT "HOW DO YOU REACT?"
940   PRINT
950   PRINT"SPIT(1)","SWEAR(2)","SCREAM
      FOR HELP(3)",,"POKE AT IT WITH A
      STICK(4)","SAY YOU DONT SPEAK ENGLISH
      (5)","SAY YOU ARE A STRANGER IN TOWN
      (6)nospaceBOOK A
     FLIGHT TO PARIS(7)",
    "HAIL A CAB(8)",,,
    "BUY A RIOT SHIELD
      (9)",,"POINT THE OTHER WAY(10)?"
1000  INPUT A
1005  IF A > 10 OR A < 1 THEN GOTO 1000
1010  LET B = RND(10)
1020  GOSUB 1610
1030  IF RND(5) = 1 OR A < B THEN LET A = B
1040  IF A = B THEN PRINT "YOU FOOLED
      THE ";B$
1050  IF A = B THEN LET S = S + RND(10)
1060  IF A = B THEN PRINT "AND NOW HAVE
      $";S
1070  IF A = B THEN LET W = W + RND(4)
1080  IF W > 10 THEN LET W = 10
1090  PRINT
1100  IF NOT A = B THEN PRINT "THE ";B$;
      " OUTWITTED","YOU"
1120  IF NOT A = B THEN LET S = S - RND(5)
1130  IF W < 10 THEN PRINT "YOU ARE CLOSE
      TO",,"FOOTPATH ";CHR$(128);CHR$(156 +
      CHR$(128)
1140  GOTO 160
1170  LET A = RND(4)
1180  GOSUB -1240*(A = 1) - 1340*(A = 2)
      - 1380*(A = 3) - 1510*(A = 4)
1190  GOTO 160
1240  PRINT ,"YOUVE FALLEN DOWN"
1250  LET B = RND(4)
1260  IF B = 1 THEN PRINT "  IN FRONT OF
      BIG BEN"
1270  IF B = 2 THEN PRINT "   IN THE STRAND'
1280  IF B = 3 THEN PRINT "IN FRONT OF THE
      TRAFFIC","IN PICCADILLY CIRCUS"
1290  IF B = 4 THEN PRINT RND(99);" STEPS
      TO THE THAMES"
1310  LET W = W - 1
1320  LET S = S - RND(2)
1330  RETURN
1340  PRINT "fivespaceYOU ARE CAUGHT RIDING
      ON A","fivespaceNO. ";RND(99);" BUS
      WITHOUT A TICKET"
1350  LET W = W - 1
1360  LET S = S - RND(2)
1370  RETURN
1380  PRINT ,"YOU GAIN A"
1390  LET D = RND(6)
```

```
1400  IF D = 1 THEN PRINT ,"STALE SALAD R
1410  IF`D = 2 THEN PRINT "TICKET TO THE
      PLANETARIUM"
1420  IF D = 3 THEN PRINT "GOOD SEAT AT
      COVENT GARDEN"
1430  IF D = 4 THEN PRINT "TUBE TICKET TO
      SHEPHERDS","BUSH"
1440  IF D = 5 THEN PRINT "SEAT UPSTAIRS
      ON A ";RND(99);" BUS"
1450  IF D = 6 THEN PRINT "BLACK EYE FROM
      A RUDE LOCAL"
1470  LET S = S + RND(5)
1480  LET W = W + RND(3)
1490  IF W > 10 THEN LET W = 10
1500  RETURN
1510  PRINT "YOU FIND A MAP OF LONDON"
1520  PRINT
1530  PRINT
1540  PRINT "CHOOSE YOUR BONUS (UP TO $5)
1550  INPUT A
1570  LET S = S - A*(A < 6)
1580  LET W = W - A/2
1590  RETURN
1610  CLS
1615  FOR M = 1 TO 25*RND(5)
1620  NEXT M
1630  FOR M = 1 TO 5
1640  PRINT
1650  NEXT M
1660  RETURN
1690  PRINT "YOU ARE AT THE PALACE GATE"
1700  PRINT "DO YOU HAVE $35 TO BRIBE YOU
1710  PRINT "WAY IN? PRESS N/L TO FIND OU
1720  INPUT A$
1730  GOSUB 1610
1740  IF S < 35 AND W > 5 THEN PRINT "YOU
      HAD ONLY $";S;"  SO YOUVE","BEEN
      DEPORTED..."
1745  IF W < 1 THEN PRINT "YOUVE FAILED A
      A TOURIST...AND BEEN BEHEADED...";
1750  IF S < 35 THEN GOTO 1740
1760  PRINT "YES, YOU HAVE $";S
1770  PRINT "THE RED CARPET AWAITS YOU...
1780  RUN 1770
```

Double your RAM

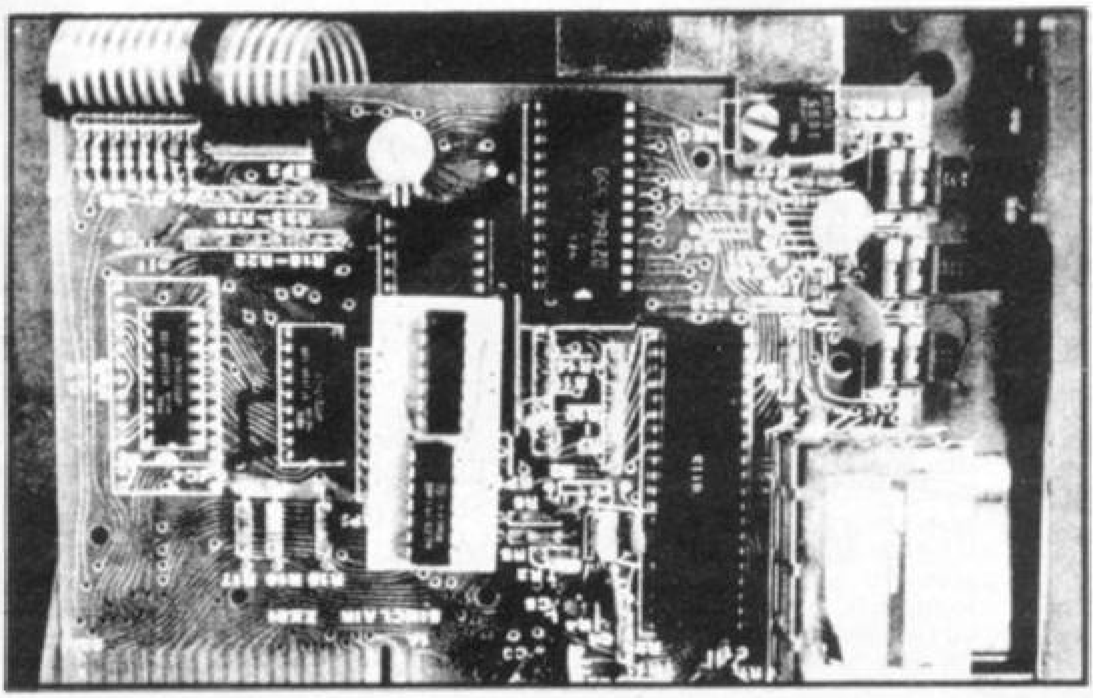
The limits of the 1K supplied on-board with the standard ZX81 become frustratingly obvious once you start to write programs. You probably spend more of your time trying to save memory with little tricks than you do in improving the program. Here's one solution. Stephen Adams, author of '20 Simple Electronic Projects for the ZX81' explains how you can double the memory of your computer. Even if you've never touched a soldering iron before, you should be able to carry out the simple modifications necessary to get some workable RAM space. This RAM addition gives not only a full 24 line screen, but also gives twice as much program space. The 1K extra RAM described in this article is cheaper to install than Sinclair's own suggestion of replacing the 1K chip supplied with a 2K (4816) RAM chip.

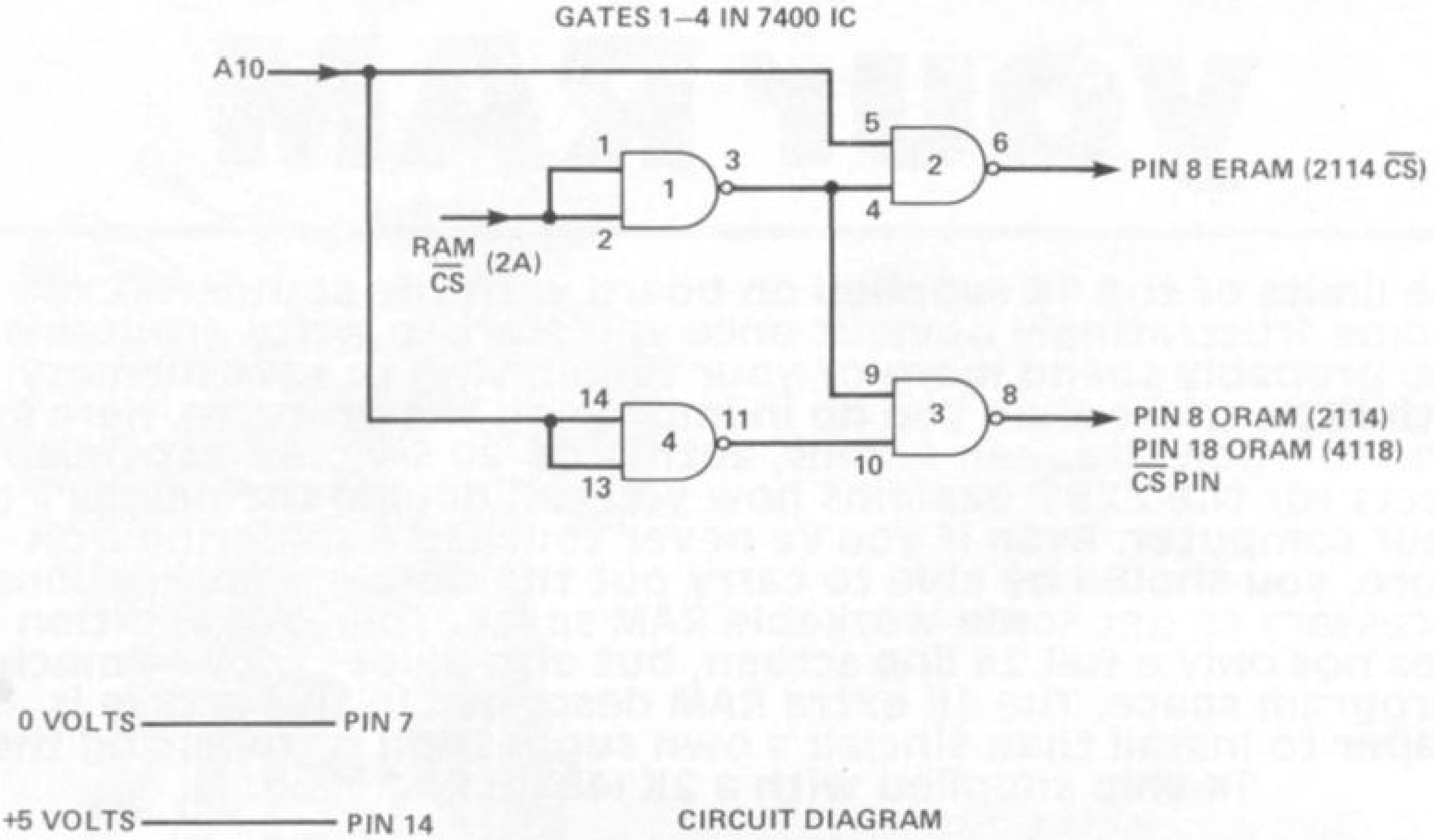
The 1K of original RAM (ORAM) can come in two forms, two 2114s or a single 4118. The single chip is identified as IC4 in the circuit diagram and two 2114s as IC4a and 4b. Both are connected up to the same data and address lines. They are operated by the same signal, RAM CS, which is generated by the control chip known as the Sinclair Control Logic or ULA. This chip decodes the upper two address lines A14 and A15, so that when the correct address for the RAM (16384-32767) is on the address lines, the RAM CS line to the RAM chips is lowered to binary 0.

This line is then used to turn on the RAM chips, so that the other address lines using A0 to A9 can detect which byte is required. The data is transferred into, or out of, the RAM chip on the data bus. As the decoding of A15/A14 provides only a 16K section in memory, the RAM chip appears again and again, at every 1K within that 16K section. In order to add another 1K of RAM, we have to divide this 16K space into larger units or else both chips will appear at the same address. The line we will use is A10, which will provide us with 2K sections instead of 1K ones. In other words, it will only repeat every 2K within the 16K section.

The 7400 IC Decoder

The address decoding is done by a 7400 NAND gate IC, which contains four such gates. A NAND gate will only change its output to binary 0





CIRCUIT DIAGRAM

when both inputs are binary 1, otherwise the output is binary 1. Figure 1 shows the connections to the IC.

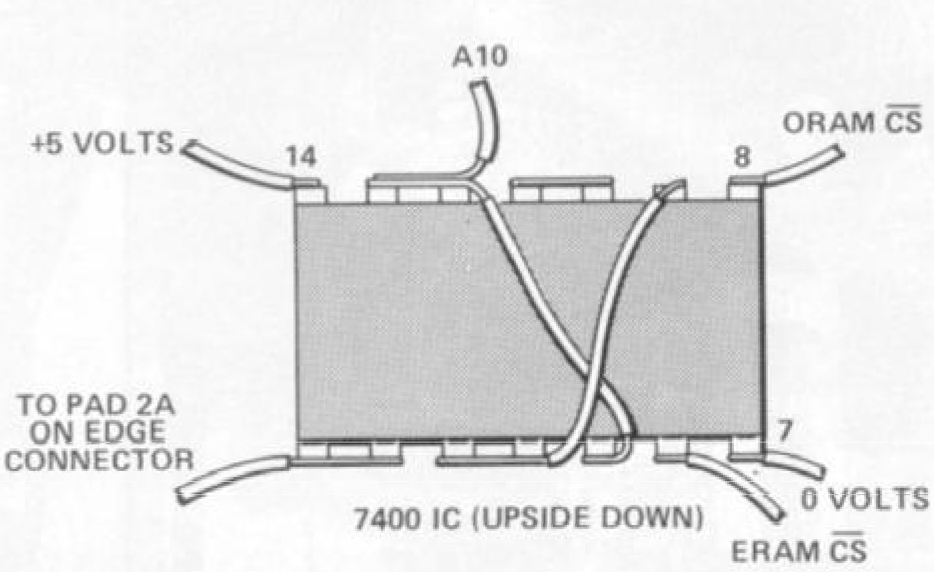
The RAM CS line is connected to both inputs of gate one, so that its output of gate one is constantly binary 0, unless the correct 16K section is decoded by the ULA. If the RAM CS line is changed to binary 0 by the ULA, one of the inputs to gates two and three is binary 1. The other input to the gate depends on A10. It must be binary 1 to operate gate two, and binary 0 for gate three to operate.

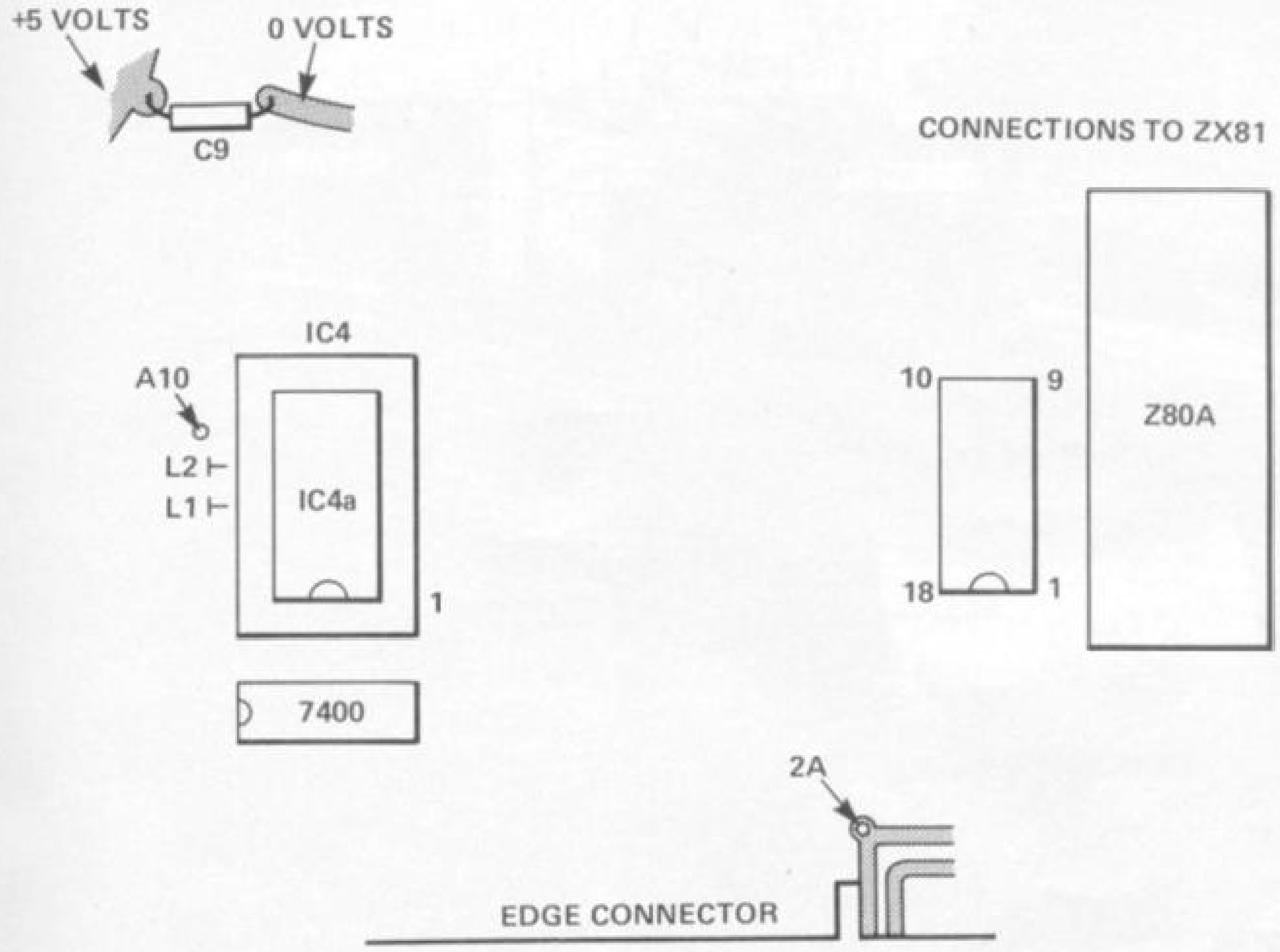
As A10 changes every 1K, starting as binary 0, then changing to binary 1, then back to binary 0 again, the ORAM will be first turned on by gate three going to binary 0, and then the ERAM (Extra RAM). The next RAM to come on in the third change will be ORAM, as A10 is now binary 0 again. This sequence will repeat itself all the way through the 16K section.

Construction and Alterations

The first step is to wire up the 7400 IC as shown in Fig. 2, remembering that it is shown upside down. This is because we are going to mount it on th ZX81's PCB using a piece o Blu-tack or a double-side sticky pad, so the legs of the I are sticking upwards, and d not make contact with th ZX81's PCB. The only connec tions that are made are to th PCB at the points shown. Th connection to the ORAM C lines is made underneath th PCB.

The RAM CS line appears o the edge connector pad 2A. W can solder our connection t the 7400 IC (pins one and tw to the hole connected to th edge connector pad 2A. Th printed circuit track leadin from this pad to pin eight o IC4b must be cut. Pick a sp half-way along the track an make two small cuts about a eighth of an inch apart, using craft knife. Then, still using th craft knife, scrape away all the track in between, so there no connection between th two. Attach the wire from p eight of the 7400 IC to pin eig of IC4b, whether fitted or no Keep all the wire to the 7400 as short as possible withou touching any other track. A tach the 0 volt and + 5 volt wi to either side of capacitor C making sure you get them th right way round. The only wi





left is the ERAM CS line. The position you fix this will depend on what sort of ORAM you have. Stick the 7400 IC (legs upwards) onto the printed circuit board in the position shown. The A10 connection can be made by pushing a wire into the hole next to L2 and soldering it.

Two 221's as ORAM

In this case, remove the two ORAM ICs and mount two more 2114s on top of them. ERAM 2114s should have their pin eights cut short, so that when they are mounted on the ORAM the two pin eights do not touch. Solder this 'piggy back' arrangement together, except for the pin eights. The top ERAM's pin eight should then be soldered to a piece of wire making sure that it does not touch the lower ORAM's pin eight. These pins are the RAM's CS pins.

Put the 2114s back into their sockets, making sure the ICs are the correct way round, as shown in Fig. 3. The top and bottom RAMs should be the same way round. Now join the two pin eights of the ERAMs together, and solder the ERAMs together. Solder the ERAM CS wire to one of them, making sure it does not touch the ORAM's pin eight.

One 4118 as ORAM

This is a bit more difficult as the ORAM covers up one of the ERAM sockets. An IC socket should be soldered into position onthe IC4b, then a 2114 IC placed in it with its pin eight left outside the IC socket. The other 2114 must be mounted underneath the IC4. This is done by laying the top of the 2114 against the PCB over the pin connections for IC4a. Now solder wires from the IC4a connections to the 2114 RAM IC, making sure it is the right way round, so that the cutaway of the IC is nearest the edge connector. Do not solder a wire to pin eight. Join the two ERAM's pin eights together using a piece of wire and then solder the ERAM CS to pin eight of the ERAM in socket IC4b.

Finishing touches

Before applying any power to the ZX81, check all the connections for bridges of solder between tracks on the PCB and on the ICs. Check that the connections to the PCB are not likely to wander about and touch anything else. On re-assembling the ZX81 and applying 9 volts to it, you will find that PRINT PEEK 16389 gives a result of 72. If not, or the inverse 'K' does not appear, check all your connections again.

The reason for all this work is that 2114s only cost 99p plus VAT, whereas the cheapest 4816 is £7.00 plus VAT or more. If — despite the cost — you want to install a 4816, place it the same way round in the socket of the 4118 and cut the strap from L1 to L2. A new strap must then be inserted between L2 and A10. This modification will not stop you using a 16K pack, as the RAM CS connection on the edge connector still works.

Shopping list

There are at least three different ·2114 1K RAM chips available. The L version takes less current and so is useful if you are using your machine with extra boards. The other two types are differentiated by their speeds. One is 450 nS and the other 200 nS. This is the time (in nanoseconds) taken to operate the chip after the CS line is operated. The chips needed for this project can be up to 450 nS. The best thing is to look at the price of the chip, not forgetting that you will require two of them for the extra 1K of RAM.

The 7400 also comes in two versions. The original, cheaper 7400 and the more expensive 74LS00. The 74LS has more to recommend its use, however, as it requires less power to operate and has protection diodes built into the input of the gate. It is therefore best to use, as it only costs a few pence more. The 7400 and the 74LS00 use the same pin numbers.

First steps in Machine Code

Machine code may seem bewildering, but Toni Baker — author of Mastering Machine Code on Your ZX81 or ZX80 — is convinced the whole subject is a piece of cake. Here, Toni attempts to convince you, with a clever screen-handling routine and the raw ingredients of a BREAKOUT game.

There seems to be a myth surrounding ZX81 machine code. For some strange reason it is a common belief that programming in machine code is not easy. Indeed there are even people who would go as far as to call it difficult. Suffice to say this is not the case for, while it is true that the use of machine code was omitted from the Sinclair Manual, all you need is one little (BASIC) program taken from the book, Mastering Machine Code On Your ZX81, to get you started. Here it is . . .

```
10  INPUT X
20  LET A$ = "  "
30  IF A$ = "  " THEN INPUT A$
40  IF A$ = "S" THEN STOP
50  POKE X,16*CODE A$ +
    CODE A$(2) - 476
60  LET X = X + 1
70  LET A$ = A$(3 TO) note
    there is nothing between
    "TO" and ")"
80  GOTO 30
```

Now obviously there isn't enough room in this article for me to teach the whole of machine code from scratch — that would be like trying to squash Sinclair's BASIC Manual into two or three pages — so instead I'll give you one or two little machine code routines which you can use in your own BASIC programs.

Try this one. Load the above program, then add the following line:

```
1 REM 123456789023456789
```

Now RUN the program and input:

```
16514                    Machine
                         code address
"2A0C40"                 LD HL,
                         (D.FILE)
"0618"                   LD B,
                         twentyfour
"23"             LOOP:   INC HL
"7E"                     LD A,(HL)
"EE80"                   XOR 80h
"FEF6"                   CP F6h
"2803"                   HR Z,EXIT
"77"                     LD (HL),A
"18F5"                   JR LOOP
"10F3"           EXIT:   DJNZ LOOP
"C9"                     RET
```

Now please note — you do *not* need to understand what I've written in the right-hand column. All you need to do is enter what's written in the left column. When you've done that type "S" to stop the program.

Now add these lines:

```
100 IF INKEY$ < > "  " THEN
    GOTO 100
105 IF INKEY$ = "  " THEN GOTO
    105
110 RAND USR 16514
120 GOTO 100
```

Now type RUN 90 and just see what happens when you touch any of the keys. Incidentally, lines 10 to 80 can actually be deleted altogether now, and only line one is needed. From now on, any time the machine encounters the statement RAND USR 16514 the machine code magic will work.

Try this little gem. Delete all the lines except line one. Now add the following lines:

```
10  FOR I = 1 TO 100
20  PRINT "three spaces
      followed by three inverse
      spaces";
30  NEXT I
40  FOR I = 1 TO 50
50  NEXT I
60  RAND USR 16514
70  GOTO 40
```

Run this and stand back in amazement.

For something a little more complex, here is a routine which will move a ball around a screen, bouncing it off walls and bricks.

You can develop a kind of BREAKOUT-type game using this routine. Load the machine code loader at the start of the article and add one extra line:

```
1 REM 12345678901234567 8
      901234567890123456
      789012345678901234
      567890123123456789
      012345678901234567
      890123
```

Now RUN the program and input the following (counting "/" as "newline"):

```
16516/0101/2A8240/3600/
3A8440/3D/2002/23/23/2B/7E/
FE80/200B/2A8240/4A8440
ED44/328440/228240/3A8540
/3D/2006/11DFFF/19/1804/
112100/19/7E/FE80/200B
2A8240/3A8540/ED44/328540
/010000/7E/FE08/2009/03/
3A8540/ED44/328540/228240
/3634/C9/S
FE80/200B/2A8240/3A8440
```

You can now delete lines 10 onwards.

To demonstrate how this works just add the following BASIC:

```
10  PRINT "thirty-two inverse
      spaces"
20  PRINT "inverse space thirty
      spaces inverse space"
30  as 20
40  PRINT "inverse space thirty
      graphic A inverse space"
50  as 40
60  FOR I = 1 TO 10
70  PRINT "inverse space thirty
      spaces inverse space"
75  NEXT I
76  PRINT "thirty-two inverse
      spaces"
80  LET X = PEEK 16396 + 256*
      PEEK 16397 + 200
90  POKE 16514,X - 256*INT
      (X/256)
100 POKE 16515,INT (X/256)
110 LET A = USR 16518
120 GOTO 110
```

Lines 10 to 76 just print the board. You can replace these by any board-printing routine you like. Lines 80 to 100 are absolutely vital for the machine code to work, (although the +200 can be more or less +anything — it just determines the position of the start of the ball), *and all the fun happens at* 110 and 120. The instruction LET A = USR 16518 just moves the ball one square along. If you PRINT A afterwards you'll find it's always zero unless the ball hits a brick, in which case it will be one.

So you see, the loop 110 to 120 is now really the whole of the 'game'. It's up to you to improve it, but as long as you don't alter the machine code and just play around in BASIC, remembering that LET A = USR 16518 means "move the ball one square", it should be no trouble at all.

HILDERBAY LTD
PROFESSIONAL SOFTWARE

If you don't believe that the Sinclair ZX81 can do real work, come and see us at the Computer Fair Exhibition, Earls Court (23rd-25th April) and the ZX Micro Fair, Central Hall, Westminster, London SW1 (30th April-1st May).

ZX81 & 16K SOFTWARE: Reviewed in "Your Computer" (Feb. 1982).

- ★ **Payroll** for up to 30 employees. Meets all regulations £25.
- ★ **Stock Control** Fast, big, and versatile £25.
- ★ **Budget** £15.
- ★ **Critical Path Analysis** Up to 500 activities £15.
- ★ **VAT & Mortgage & Loan** £8.
- ★ **GOLD**: A really good adventure; together with the intriguing "Pick a Word". £8.
- ★ RELIABLE 16K RAM Pack for the ZX81 £42.
- ★ **BEAMSCAN**: Beam analysis for architects and engineers £25.
- ★ **APPLE II** Payroll: £35.
- ★ **HEWLETT PACKARD 9845B**: Superspline (general purpose interpolation and tabulation).
- ★ **MEMOTECH** 64K Memopak £79.

DO YOU HAVE DIFFICULTY LOADING TAPES! We have the answers; and for well under £10.
Contact us for more details.

CROFTON PROFESSIONAL KEYBOARDS, MONITORS, VDU's
Prices include VAT, Post & Packing. COD Orders £2 extra.

TAPE HINT: Don't use the first 20 seconds of your cassettes — most tape problems are found near the ends of the tape.

HILDERBAY LTD
8/10 PARKWAY, REGENTS PARK
LONDON NW1 7AA

Enquiries for Hilderbay Ltd. will now be handled by HOLDCO LTD. Tel. 01-251 3090

ALSO ON ZX81 + 16K

Financial Budget/ Cash Flow Forecast: The program allows thirteen columns (12 months + total). The forecast can be changed and a "WHAT IF?" possibility can be examined on the screen. £25.

Sales Day Book: keeps a record of all invoices, with nett, VAT, zero rated and total. £18.

Purchase Day Book: keeps records of all purchases and VAT. £18.

Petty Cash Book: Keeps a record of all petty cash expenses and allows for VAT to be calculated from total if unknown. £18.

Cash Sales Day Book: as sales day book but with the additional "Method of payment" feature, incl. cash, cheque, postal order, Access, Barclays, sundries and unpaid. £18.

Cassette Recorders: individually tested for ZX81 and supplied with a test tape for loading. £22.

Barcharts for Architects: a simplified version, takes away the tedious task of manually handling the problem £25.

Bank Account: calculate your bank charges whilst keeping your account £5.

Architects Programs: Barchart with S-curve/Dewpoint calculations N.B.S. on computer/Structural programs/etc. etc.

All programs now also available for ATARI and SHARPE

HOLDCO LTD.
14, BRITTON STREET
LONDON EC1M 5NQ
Tel: 01-251 3090

MICRO GEN
QUALITY PRODUCTS

ZX81 A/D CONVERTER BOARD

This 4 Channel analogue to digital converter, originally developed for joystick control, can be used for such applications as measurment of voltage, temperature, light intensity etc. The board fits in between the RAM pack and the ZX81. (No skill is required to make this connection, and it actually improves the stability of the RAM pack).
Price now only £18.50.

JOYSTICKS FOR THE ZX81, ONLY £9.60 EACH

★ The most exiting add-on ever for the ZX81, free yourself of that dead, unresponsive keyboard.
★ 1 or 2 joysticks may be connected via our A/D board.
★ Turns your ZX81 into a true programmable games machine.
★ Extends the capability of the ZX81, imagine the tremendous varity of games an applications that now become possible.
★ Details supplied on how to use the joysticks in your own programs.

Please note that you cannot connect conventional analogue joysticks directly to the digital input ports found on most I/O boards, and A/D convertor such as ours is required.

A free copy of ZX AMAZE plus any one of the games listed below when ordering a joystick and an A/D board.

PROGRAMS AVAILABLE

ZX SPACE INVADERS. You've tried the rest, now try the BEST.
This program has many features including an ever increasing rate of play, (they'll get you in the end).
only £3.95

ZX BREAKOUT. Quite simply the best breakout on the market.
Features seven bat angles, (you won't find this one easy).
only £3.95

ZX NEW YORK. A very addictive arcade game. Bomb and shoot your way out of trouble, otherwise you are doomed to crash.
Generates a different pattern, for a different game each time you play. On the reverse of the cassette is ZX REFLEX, find out how fast you really are.
now only £3.95

ZX CHESS. The original and still the best.
Graphic display of Chessboard 6 levels of play. Displays record of your moves and the computers. Board can be set up to any position. Has ability to change sides or level in mid-game. PLUS CHESS CLOCK on reverse side, records time taken by each player. Resetable function. Single key entry.
NOW ONLY £6.50

All our games are written in machine code, and can be used with joysticks or keyboard, (except chess, keyboard only). Supplied on cassette with library case.

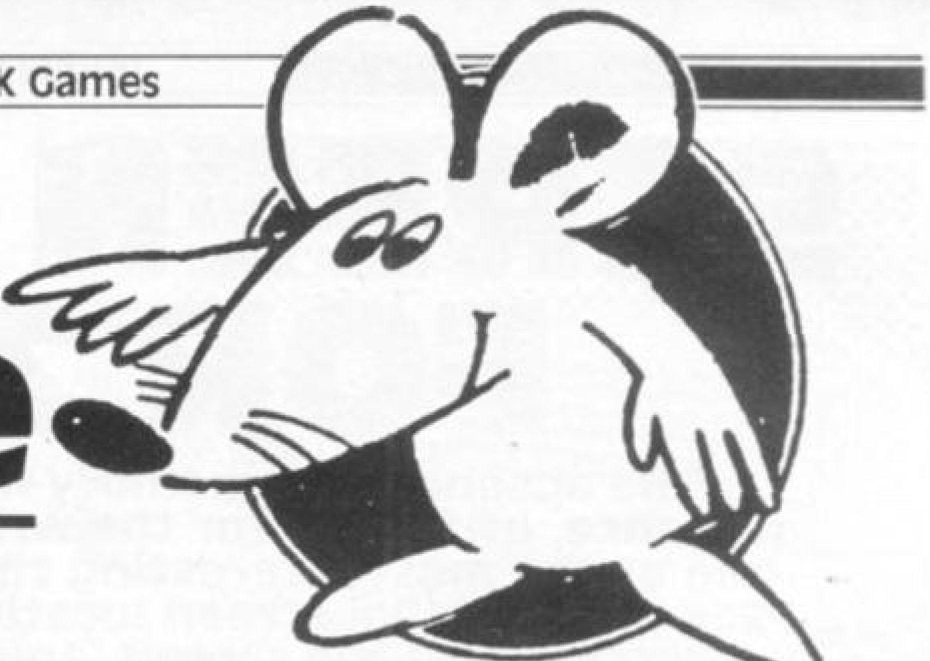
To allow you to prove to yourself that our products are second to none, MICRO GEN offers the following terms to our customers.

14 DAYS FREE APPROVAL
ON ALL PRODUCTS

All money will be refunded if goods are returned in good condition within fourteen days of despatch.

**Postal Orders payable to:
MICRO GEN Dept ZXC
24 Agar Crescent, Bracknell, Berks.
TEL: (0344) 27317**

Micro-Mouse



This program is fascinating to watch, and makes a good demonstration of the ZX81. A 'mouse' (an inverse asterisk) starts off in a random position somewhere near the top left hand corner of the screen. It is aiming for the bottom right hand corner. There is a solid black 'frame' on the screen, and a number of obstacles are placed randomly within the frame. When the mouse finally makes it to the end, the number of moves it took is shown.

The obstacles change position, the mouse flashes off and on a few times, and then begins again.

The mouse cannot get itself into a trap it cannot get out of, unless it happens to land in one at the very beginning, so have patience, no matter how long it seems to be taking.

```
   1 REM MICROMOUSE 16K
   5 LET G=16398
   6 LET H=G+1
  10 PRINT AT 0,0;"■■■■■■■■■■■■■■■■
■■■■■■■■■■■■■■■■"
  20 PRINT AT 21,0;"■■■■■■■■■■■■■■■■
■■■■■■■■■■■■■■■■"
  30 FOR A=1 TO 20
  35 PRINT AT 2+RND*18,1+RND*27;
"  "
  36 PRINT AT 2+RND*18,1+RND*27;
"  "
  37 PRINT AT 2+RND*18,1+RND*27;
"  "
  42 PRINT AT A,0;"■"
  43 PRINT AT 3+RND*15,2+RND*22;
"■ ■"
  44 PRINT AT 2+RND*18,1+RND*27;
"  "
  45 PRINT AT 2+RND*18,2+RND*24;
"■"
  46 PRINT AT 3+RND*15,2+RND*27;
"▒"
  47 PRINT AT 2+RND*18,2+RND*24;
" ▒"
  50 PRINT AT A,31;"■"
  57 PRINT AT 2+RND*18,2+RND*26;
"  "
  60 NEXT A
  61 FOR Z=1 TO 13
  62 PRINT AT 20,30;" "
  63 PRINT AT 20,30;"■"
  64 PRINT AT 20,30;"■"
  65 PRINT AT 20,30;" "
  66 PRINT AT 20,30;"■"
  67 PRINT AT 20,30;" "
  68 NEXT Z
  70 LET A=INT (RND*6+1)
  75 LET Q=0
  80 LET B=INT (RND*15+1)
  85 PRINT AT 20,30;" "
  90 LET E=A
  95 LET Q=Q+1
 100 LET F=B
 101 IF A=20 AND B=30 THEN GOTO
2000
 105 LET T=0
 106 IF RND>.2476 THEN GOTO 120
 110 LET Y=INT (RND*7)+1
 111 IF Y=1 THEN GOTO 120
 112 IF Y=6 THEN GOTO 169
 113 IF Y=3 THEN GOTO 200
 114 IF Y=4 THEN GOTO 250
 115 IF Y=5 THEN GOTO 290
 116 IF Y=2 THEN GOTO 154
 117 IF Y=7 THEN GOTO 330
 120 PRINT AT A+1,B;
 130 IF PEEK (PEEK G+256*PEEK H)
=0 THEN LET T=1
 140 IF T=1 THEN LET A=A+1
 150 IF T=1 THEN GOTO 1000
 152 IF RND>.2 THEN GOTO 169
 154 IF A=0 OR B=30 THEN GOTO 16
9
 155 PRINT AT A-1,B+1;
 156 IF PEEK (PEEK G+256*PEEK H)
=0 THEN LET T=1
 157 IF T=1 THEN LET B=B+1
 158 IF T=1 THEN LET A=A-1
 159 IF T=1 THEN GOTO 1000
 165 IF RND<.2 THEN GOTO 110
 169 PRINT AT A,B+1;
 170 IF PEEK (PEEK G+256*PEEK H)
=0 THEN LET T=1
 180 IF T=1 THEN LET B=B+1
 190 IF T=1 THEN GOTO 1000
 195 IF RND<.6 THEN GOTO 290
 200 PRINT AT A+1,B+1;
 210 IF PEEK (PEEK G+256*PEEK H)
=0 THEN LET T=1
 220 IF T=1 THEN LET A=A+1
 230 IF T=1 THEN LET B=B+1
 240 IF T=1 THEN GOTO 1000
 245 IF RND<.1 THEN GOTO 110
 250 PRINT AT A-1,B;
 260 IF PEEK (PEEK G+256*PEEK H)
=0 THEN LET T=1
 270 IF T=1 AND A>0 THEN LET .A=A
-1
 280 IF T=1 THEN GOTO 1000
 290 PRINT AT A,B-1;
 300 IF PEEK (PEEK G+256*PEEK H)
=0 THEN LET T=1
 310 IF T=1 AND B>0 THEN LET B=B
-1
 320 IF T=1 THEN GOTO 1000
 330 IF B=0 OR A=0 THEN GOTO 110
 340 PRINT AT A-1,B-1;
 350 IF PEEK (PEEK G+256*PEEK H)
=0 THEN LET T=1
 360 IF T=1 THEN LET A=A-1
 370 IF T=1 THEN LET B=B-1
 380 IF T=1 THEN GOTO 1000
 390 GOTO 110
1000 PRINT AT E,F;" "
1010 PRINT AT A,B;"■"
1020 GOTO 90
2000 PRINT AT 0,15;"■";Q;"■"
2010 FOR N=1 TO 50
2020 NEXT N
2030 PRINT AT 0,15;"■■■■■■■"
2040 GOTO 30
```

SCREEN POKES FOR ZX80

The absence of a memory-mapped display can be a nuisance, especially for the writer of games programs, as one of the most interesting things one is able to do is to PEEK at individual screen locations and to POKE characters directly on to the screen. Animated graphics, of course, depend on this facility but they are definitelyout with the ZX80 because the screen would remain blank while the action was being computed. On the other hand using POKE to put characters onto the screen is feasible and is potentially a useful feature.

Filing A Display

With a memory-mapped display there is no problem because the display file is contained within a fixed amount of RAM. The screen can be considered to consist of a matrix of locations (number of lines by number of characters per line) with the memory address of each one fixed and known. To make a character appear at any desired point on the screen it is simply a matter of POKEing the code for that character at the relevant location address.

On the ZX80 things are rather different. The display-file uses a variable amount of RAM depending on the quantity of data to be displayed. The addresses of the various locations on the screen also vary according to the length of the program. In addition the location addresses change during the running of a program whenever data is input for the first time or variables are assigned.

The computer, of course, knows where the display-file is in the RAM at any time and the address of the start of the display-file is recorded as a two-byte record at address 16396. By PEEKing at that address we can locate the display-file and then calculate the addresses where we need to POKE to get characters on to the screen.

Character By Character

The first character in the display-file is a "newline" character so that if we call the address of the start of the display-file W then the first visible character location (top left) is at W + 1. Each line consists of up to 32 visible characters with a new line character at column 33. By adding the appropriate multiple of 33 plus the column number to W we can get the address of any character location on the screen. If we call the row number A and the column number B then the address formula is W + (A −1)*33 + B.

Of course the display-file has to exist before we can start PEEKing and POKEing at it. If we wish to POKE onto a blank screen then it is first necessary to create a display-file full of spaces. Unfortunately a succession of PRINT statements will not achieve this and although a FOR . . . NEXT loop PRINTing individual spaces will, it is very cumbersome. Luckily PRINT , , , , creates a line full of spaces so a short loop can be used to produce the required number of screen lines. Obviously characters can be used as well as spaces to create a display-file. Up to 23 lines can be printed in this way.

Having ensured that we

have a display-file we can now take a PEEK at its starting address. The following subroutine achieves this and it is used in all subsequent listings:-

Of course the display-file has to exist before we can start PEEKing and POKEing at it. If we wish toi POKE onto a blank screen then it is first necessary to create a display-file full of spaces. Unfortunately a succession of PRINT statements will not achieve this and although a FOR . . . NEXT loop PRINTing individual spaces will, it is very cumbersome. Luckily PRINT . . . creates a line full of spaces so a short loop can be used to produce the required number of screen lines. Obviously characters can be used as well as spaces to create a display-file. Up to 23 lines can be printed in this way.

Having ensured that we have a display-file we can now take a PEEK at its starting address. The following subroutine achieves this and it is used in all subsequent listings:-

```
500 LET P = PEEK(16397)
510 IF P>127
    THEN LET P = P - 256
520 LET W = PEEK(16396)
    + P*256
530 RETURN
```

It should now be obvious how we can use this address to POKE a character onto the screen. The following program establishes a blank display-file, inputs a row and column number, POKEs character code 148 (inverse asterisk) at the relevant address and then inputs another "grid reference". When the program is run, inverse asterisks appear at your bidding anywhere on the screen:-

```
 10 LET P = 0
 20 LET W = 0
 30 FOR A = 1 TO 22
 40 PRINT , , , ,
 50 NEXT A
 60 INPUT A
 70 INPUT B
 80 IF A>22 OR B> 32
    THEN GO TO 60
 90 LET Y = (A-1)*33 + B
100 GOSUB 500
110 POKE W + Y,148
120 GOTO 60
500 LET P = PEEK(16397)
510 IF P>127 THEN LET
    P = P - 256
520 LET W = PEEK(16396)
    + P*256
530 RETURN
```

The following two alterations to the listing extend this simple program:-
Specify character to be POKEd:-

```
 84 INPUT C
110 POKE W + Y,C
 (C is relevant character code)
```

POKE character taken from the keyboard:-

```
 84 INPUT C$
 86 LET X = CODE (C$)
 88 IF X> 191
    THEN GOTO 84
110 POKE W + Y,X
```

It will be noticed that the programs above assign variables P and W before the first PEEK. This is because, as mentioned before, any variable assignment or initial input will alter the location of the display file. If you write any screen-POKE programs and find that the characters are displaced it will almost certainly be because a variable in either PEEK or POKE has not been previously assigned. A similar case is where an initial input or an assignment is made after a previous PEEK or POKE, when it will be necessary to take another PEEK at W before POKEing again.

Careful POKEs

Another thing worth remembering is that POKEing can be a hazardous occupation if you happen to POKE in the wrong place or even if you POKE an inappropriate character code in the right place. Care should therefore be taken when writing programs to ensure that characters are not POKEd outside the boundaries of the display-file. Usually such characters seem to disappear without trace but sometimes they can find their way into your program, invariably with unpleasant consequences Some bad POKEs can cause havoc with the video control. The codes for all statements, tokens and operators should *definitely* be avoided (ie codes > 191).

A more subtle problem is that any extensive use of screen space is very expensive in terms of memory. A 23 line "blank" screen will occupy 760 bytes of RAM, which does not leave much for the program if you are using the basic model ZX80 with 1K of memory. You therefore need to think hard about the balance of memory requirements when writing screen-POKE programs if you have no memory expansion.

Having grasped the principles involved in defining and locating the display-file it is relatively simple to manipulate it. Existing characters on the screen can be replaced by POKEing an alternative code at the same address. If this is the code for a space (0) then the character already on the screen disappears. By PEEKing at the address you plan to POKE to you can see what character already occupies that location, thus opening up the possibility of a conditional response. All the relevant character codes are identified in the ZX80 handbook.

Graphic Example

Finally, here is a simple games program that demonstrates the features discussed and which just fits onto the 1K ZX80. The computer prints up a display consisting of black and grey squares in a pattern determined by a number input at the beginning of each series of games. The object of the game is to get the "woodworm" (an asterisk), which first appears at line 8 column 1, to eat its way across the screen to column 32 in the least number of moves. The snag is that the black squares represent a particularly tough kind of wood and each time one is eaten a penalty of 5 moves is incurred. Numerals 6, 7 or 8 are input as pseudo-cursor controls to move the insect down, up, or forwards respectively. The computer keeps track of the number of moves taken to reach column 32 and displays the total at the end of each game together with the best performance in the present series. Pressing NEWLINE after a game sets up another game in the same series. Entering a character starts a new series.

```
  2 LET Y = 32000                 No. of moves — best so
                                  far!
  4 INPUT R                       Seed for random number
                                  generator
  8 LET P = 0
 10 LET W = 0                     Assign variables prior to
                                  PEEK
 12 LET B = 1                     and POKE
 14 LET A = 8
 16 LET Z = -1
 18 LET M = 0
 20 RANDOMISE R                   Set seed for random
                                  number generator
 22 FOR N = 1 TO 352
 24 LET D = 9
 26 LET X = RND(2)
 28 IF X = 1 THEN LET D = 128     Print eleven lines with
 30 PRINT CHR$(D);                black and grey squares at
                                  random. Pattern
                                  determined by R.
 32 NEXT N
 34 GOSUB 500                     Locate display-file
 36 POKE W + 232, 20              Insect in initial position
 38 LET Z = Z + 1                 Count No. of moves
 40 INPUT C                       Which way?
 42 GOSUB 500                     Locate display-file
 44 LET M = W + (A-1)*33 +
    B
 46 POKE M, 0                     Put a space where insect
 48 IF C = 6 and A > 11           Set A and
    OR C  = 7 AND A>1 THEN        make sure we don't
    LET A=A-2*C+13                POKE off-screen
 50 IF C = 8 THEN LET B = B+1
 52 LET M = W + (A-1)*33 +        Set M to next insect
    B                             location address
 54 IF PEEK(M) = 128 THEN LET     If there's a black square in
    Z = Z + 5                     the way, add penalty
 56 POKE M, 20                    Put insect in next location
 58 IF B = 32 THEN GOTO 62        Watch for end of game
 60 GOTO 38                       Next move
 62 IF Z> Y THEN Y = Z            Set Y to best so far
 64 PRINT "END OF GAME IN
    "Z," MOVES"
 66 PRINT "BEST SO FAR", "Y,"
    MOVES"
 68 INPUT X$
 70 CLS
 72 IF X$ = '' ''THEN GOTO 8      NEWLINE for another
                                  game
 74 RUN                           Any character for another
                                  series
500 LET P = PEEK(16397)
510 P>127 THEN LET P = P-256      Subroutine for setting W
520 LET W = PEEK(16396)           to address of start of
    + P*256                       display-file
530 RETURN
```

AND FOR THE '81

While there are many similarities between the ZX80 and the ZX81 in the way the display is handled, there are some significant differences.

oves

ile

ere insect is

on't
n

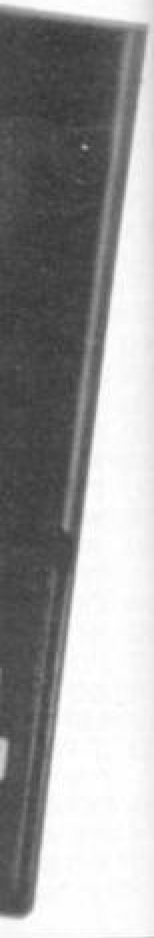
nsect
s
k square in
enalty
xt location
of game

far

nother

for another

setting W
tart of



The display file is a block of addresses that contain the CHR$ codes that form the display on the TV screen. At the beginning of the file we find a Newline CHR$, and there are also Newline CHR$ at the end of each line. As we can print up to 32 CHR$ per line, every 33rd CHR$ = a Newline CHR$. These are used by the ZX81 ROM as references and should not be altered.

To find the position of the display file we can use: 5 LET Z=1+ PEEK 16396 + 256 * PEEK 16397. "Z" will now be the address of the first print position on the screen. If we POKE Z with any CHR$ code then this CHR$ will be printed n the first position on the position, and so on, as shown on the chart.

To prevent printing, or should I say POKEing, to the 33rd CHR$ there are generally three different approaches.

1. To use a dedicated POKE routine that can never work out to hit a 33rd CHR$.
2. To peek at the position on the screen before POKEing, ie if PEEK (position) = 118 (Newline CHR$) then don't POKE here.
3. To use a formula that will automatically skip every 33rd CHR$, ie let us say X is the position we wish to poke then we can use POKE (Z+X+INT(X/32),(CHR$ code) this will automatically add one after every 32nd CHR$ making 33=34 effectively, and so on for every line. A simple program using this formula:-

```
 5 LET Z=1+PEEK 16396
    +256*PEEK 16397
10 FOR X=* TO 763 STEP 7
20 POKE (Z+X+INT(X/32)),8
30 NEXT X
40 STOP
```

(Note:- That this formula automatically adds one to every 33rd position. The memory map chart must be read as (position) — (line number on left) to give the correct number per position).

ANOTHER SHORT PROGRAM

```
 5 LET Z=1+PEEK 16396
    +256*PEEK 16397
10 LET A=0
20 LET B=33
25 IF A<0 OR A>767
   THEN GOTO 35
30 POKE Z+A+INT/A/32),
   128
35 LET A=A+B
40 IF A>0 THEN LET B=33
50 IF A>767
   THEN LET B=-31
60 IF A=27 THEN CLS
70 GOTO 25
```

One thing to remember is that the display file is using the memory immediately after the BASIC program. This means that if the BASIC program is altered then "run" is required to find the new value for "Z", or at least the new value for "Z" must be calculated.

An idea of a simple program may be as follows. We have a gun at the bottom left-hand side of the screen. Pressing any key from 0 to 9 will fire the gun at a target at the to line on the screen. "0" to fire straight up, "9" to fire farthest right, we will also need to PEEK at the top line in the relevant place to see if we have hit the target.

Lookinġ at this idea we can see that system 1 can be used as we don't intend to POKE over the edges of the screen, and the plotting involved has a limited path.

After the normal line 5 to find "Z" we then POKE Z+726,128 to put a square block for the gun. Then to fire straight up to the top line is easy, we simply POKE Z+693,23 then successively −33 from this, poking all the way up to 0. Taking the top line to equal 32 CHR$ wide and the number of lines down to the gun being 22 we must shift to the right 32+22 = 1.5 positions per line to hit the top right-hand corner. Considering we have 9 shifted to the right options (0 to 9) if we use 1.5+9 = 0.166* (the INKEY$ number) this will give us the right amount of shift per line to plot our gun fire. I've rounded down to 0.16 so there is no chance of over-stepping the corner if 9 is selected. So the basic program is shown as, an INKEYS input, followed by the fire routine.

(Note:- The line numbering should be followed to leave room for the rest of the program).

BASIC PROGRAM

```
  5  LET Z=1+PEEK 16396
      +256*PEEK 16397
100  POKE Z+726,128
150  LET AS=INKEY$
160  IF A$<"0" OR A$>
      "9" THEN GOTO 10
170  LET A=VAL A$
180  FOR B=693 TO 0 STEP
      -33
200  POKE Z+B,23
210  LET B=B+A*0.16
220  NEXT B
320  GOTO 110
```

(Run using KEYS 0 to 9 to fire) Now we have the problem of rubbing out the stars (*) after each shot. Try:- 300 CLS 320 GOTO 100. This rubs out the "fire" line backwards, not such a good idea. Try:- LIST 180, EDIT, change line number to 230, N/L (new line). LIST 200, EDIT, change line number to 240 and change 23 (CHR$ being poked) to 0, N/L. LIST 210, EDIT, change line number to 250, N/L. LIST 220, EDIT, change line number to 260, N/L. Then add:- 320 GOTO 150 AND 300 N/L.

Running the program, now plots the gun fire, followed immediately by plotting blank spaces in the same positions, using the same routine. This solution is acceptable though in my opinion it makes the whole firing routine too long, doubling up the time involved per shot.

It was therefore decided to try blanking out the fire one step behind leaving only one "*" printed at any one time. Try this:- 230, N/L, 240, N/L, 250, N/L, 260 N/L, (to rub out the last effort). Add lines:-190 POKE (B<693)*Z+B+33−A*0.16,0 followed by N/L. Then EDIT, change the line number to 230 and rub out (B<693) * so the line reads 230 POKE Z+B+33−A**.16,0 then N/L. This formula should reverse the plotting for one step and blank out the "*" one step behind.

This works well, but it tends to give one the impression of a snow flake drifting rather than a gun fire shot. It was decided to try blanking out three or four steps behind, to form a short train and thus killing the snow flake effect. EDIT to change line 190 to;— 190 POKE (B<594)*Z+B+132−A*0.64,* and line 230 = CLS. 320 = GOTO 100. This again I decided wasn't the effect I required.

Beginning to run out of ideas now, I decided to use the above with different CHR$ making line 200 POKE Z+B,24. You will see the difference is astounding. Finally, learning the effect of different CHR$, I settled for the flake drift effect using CHR$ 2. Because this is only a quarter size it gives the impression it's moving twice as fast (very useful). The correct final program then is:-

```
190  POKE (<693)*Z+B+
      33−A*0.16,0
200  POKE Z+B,2
230  POKE Z+B+33−A*
      0.16,0
320  GOTO 110
```

All the rest as BASIC program.

Now for the easy bit — a moving target. The easiest way is by filling up a string with 32 CHR$ and printing it at line 0,0. then shifting it along by letting the string = the string (2 TO) + the string (1) and reprinting at 0,0 creating a circular shift movement. Add lines:-

```
 60  LET BS="          "
 70  FOR B=1 TO 16
 80  LET BS=BS+CHR$(INT
      (RND*11))+"        "
 90  NEXT B
120  LET B$=B$(2
      TO)+B$(1)
130  PRINT AT 0,0;B$
```

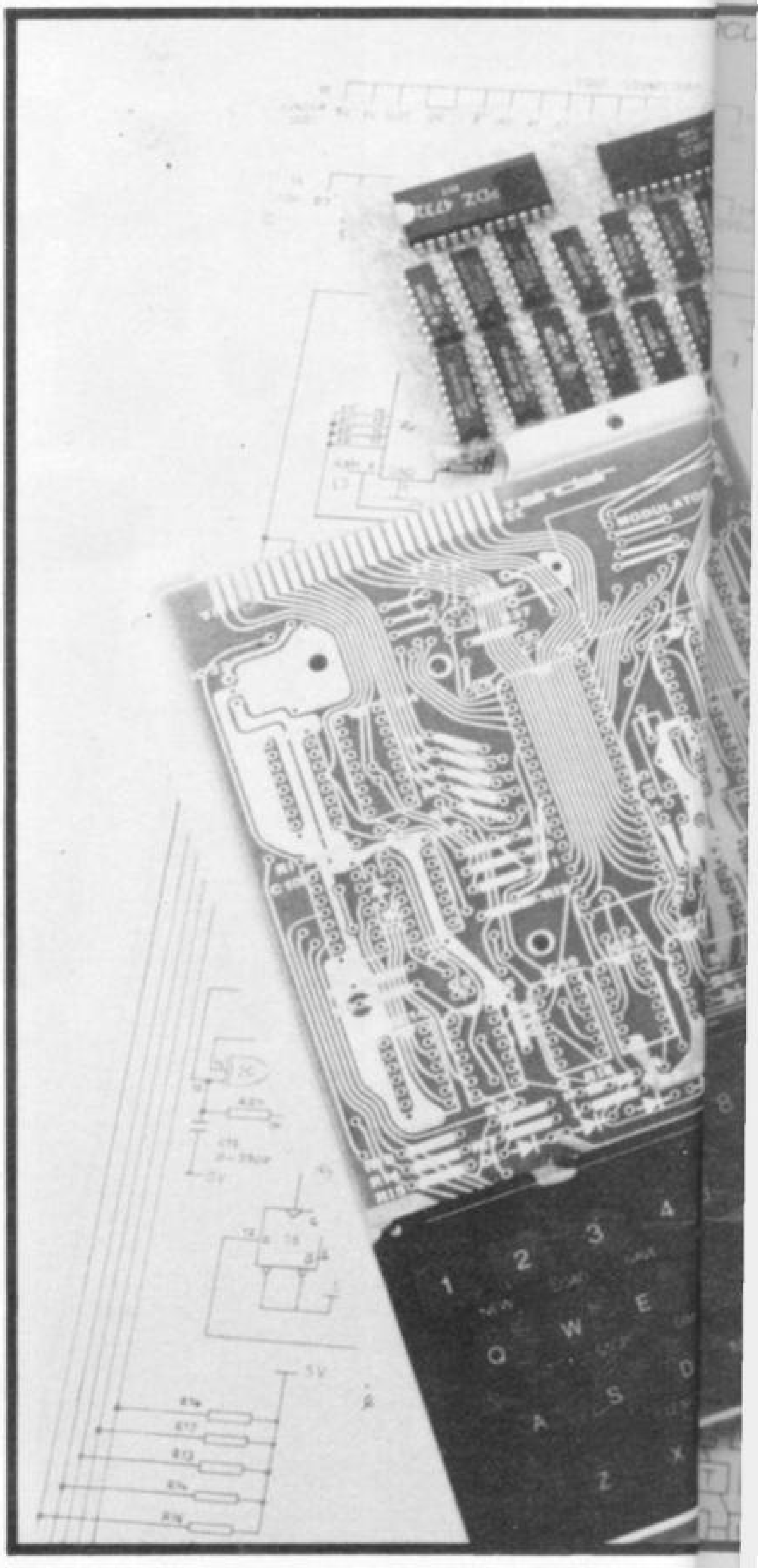
Running this, you will notice that although you bit the target, nothing happens so far. The use of CHR$ codes 0 to 10 for the targets simplifies the scoring where we PEEK, then * 10, to give us possible scores of 0 to 100.

We now need to alter the system slightly as rather than poke onto the top line we need to PEEK at the location firstly, then alter the BS accordingly. Add to main program:-

```
180  FOR B=693 TO =
      STEP −33
240  LET C=PEEK (Z+B)
250  POKE Z+B,23
```

If shots are fired from all ten positions they hit B$ at (1), (4), (8), (11), (14), (18), (21), (25), (28), (31). As these numbers don't work out conveniently the best solution is to store these numbers in a string (C$), and use the record of the INKEY$ (A) to give us a pointer to pick out the number required from this string.

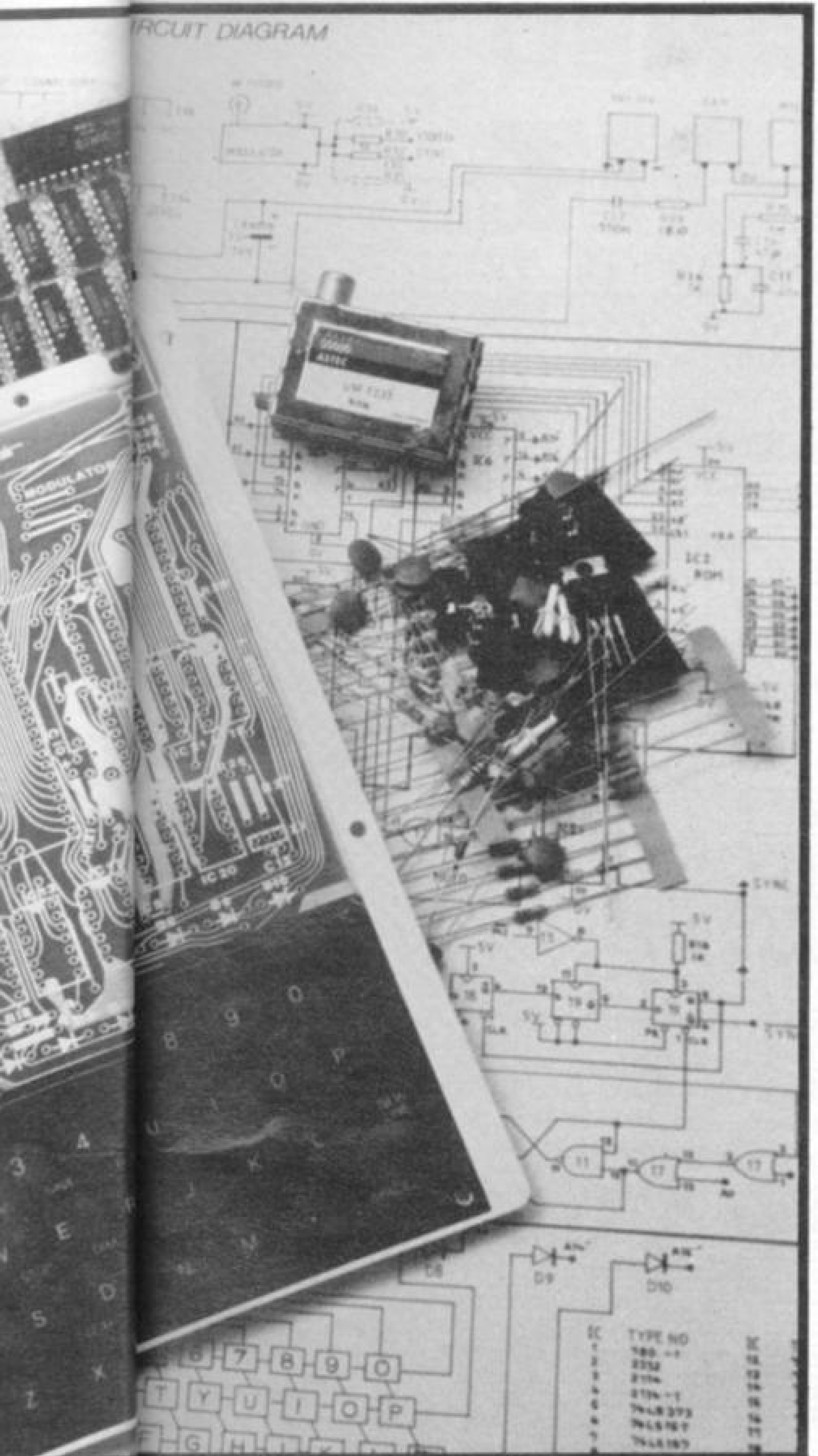
To save us worrying about single and double figure numbers we simply use CHR$ codes. Rather than entering



graphic symbols in the C$ string, I've up'ed the numbers by 27 to keep the entries simple, the 27 must be deducted from the code before use. The C$, then, is rather like a data statement that holds the numbers used to alter the B$ (targets). We also need to display the score (line 260) and blank it out before the next shot (line 110). A line of blanks is automatically produced for us in line 10. Add to main program:-

```
 10  DIM E$ (1,32)
 30  LET C$="037ADHK
      RU"
110  PRINT AT 2,0;E$(1)
260  PRINT AT 2,12;
      "MISSED" AND C=0
      "HIT" AND C>0;"=
      C*10
290  LET B$(CODE C$(A+
      −27)=" "
```

To complete the program w just put in a few subtleties.



limiting the firing to ten shots, one on each key to keep it fair. The shots information is held by S$. We must now fire in the order 0,1,2,3 – 9. Add to main program:-

```
 20 LET S$ = "Ø"
160 IF A$<>S$ THEN
    GOTO 120
300 LET S$=STR$(VAL
    S$+1)
310 IF S$="10" THEN
    GOTO 330
```

We also wish to display the overall "hit points" = H and each time the target moves we clock up "TIME" = T. Add to main program:-

```
 40 LET H=0
 50 LET T=0
140 LET T=T+1
270 LET H = H+C*10
280 PRINT AT 15,22;
    "TIME="; T; AT
    17,22;"SHOT=";S$;
    AT 19,18;"HIT PTS.="
    ;H
```

```
330 PRINT AT 0,0;E$(1)
340 PRINT AT 0,6;"HIT
    POINTS (–) TIME"
350 PRINT AT 2,6;
    "OVERALL SCORE OF"
360 PRINT AT 4,13;H-T
370 IF H-T>300 THEN
    PRINT AT 8,10;"WELL
    DONE"
380 IF H-T>400 THEN
    PRINT AT 8,10;"VERY
    GOOD"
400 IF INKEY$<>""THEN
    GOTO 400
410 IF INKEY$=""THEN
    GOTO 410
420 CLS
430 RUN
```

The object is now to try and beat the 500 score using the keys in order 0 - 9. To complete the programme a title and instructions should be written in, at lines 440 upwards and end with a GOTO 400 instruction.

Rather than fill a string full of blanks why not DIM (1,32), this makes the machine do the work for us. Then use String (1).

To extend on this idea why not DIM a string (1,352) if we now print this at 0,0; we will clear the top half of the screen. Also if we print this String (1) at 11,0; we will clear the bottom half of the screen.

A simple way to print a picture quickly is to print it via a String. The easiest way is to draw this picture on squared paper. Then type in a direct command FAST. Then line number for example:-

```
10 LET A$ = "
```

this is now followed by the rest of line being blanks.

So the first line is filled with blanks then the picture can be drawn in the normal manner as seen on the screen. When the picture is complete end the quotes then new line. We can now edit this line, rub out the unwanted blanks at the beginning and new line to have our required picture in the correct positioning.

Try this simple program:-

```
 10 FAST
 20 LET A$=" "
 30 LET B$=" "
 40 LET C$=" "
 50 FOR X=1 TO 666
 60 LET A$=A$+CHR$
    ((RND<0.1)*24)
 70 LET B$=B$+CHR$
    ((RND<0.4)*24)
 80 LET C$=C$+CHR$
    (CODE B$(X)+128)
 90 NEXT X
100 SLOW
100 PRINT AT 0,6;A$; AT
    0,4;A$; AT 0,2;A$; AT
    0,0;A$; AT0,6;B$; AT
    0,4;B$; AT 0,2;B$; AT
    0,0;B$; AT 0,4;C$; AT
    0,2;B$; AT 0,0;C$
120 CLS
```

```
  5 LET Z=1+PEEK 16396
    +256*PEEK 16397
```

ORDINARY PRINTING

```
 20 LET B=2
 30 FOR A=0 TO 21
 40 PRINT AT A,A+B; " "
 50 NEXT A
```

FIRST METHOD POKE

```
 60 LET B=34
 70 FOR A=0 TO 21
 80 POKE Z+A*B,8
 90 NEXT A
```

SECOND METHOD POKE
Same program as above but insert this line:-

```
 75 IF PEEK(Z+A*B)=118
    THEN GOTO 90
```

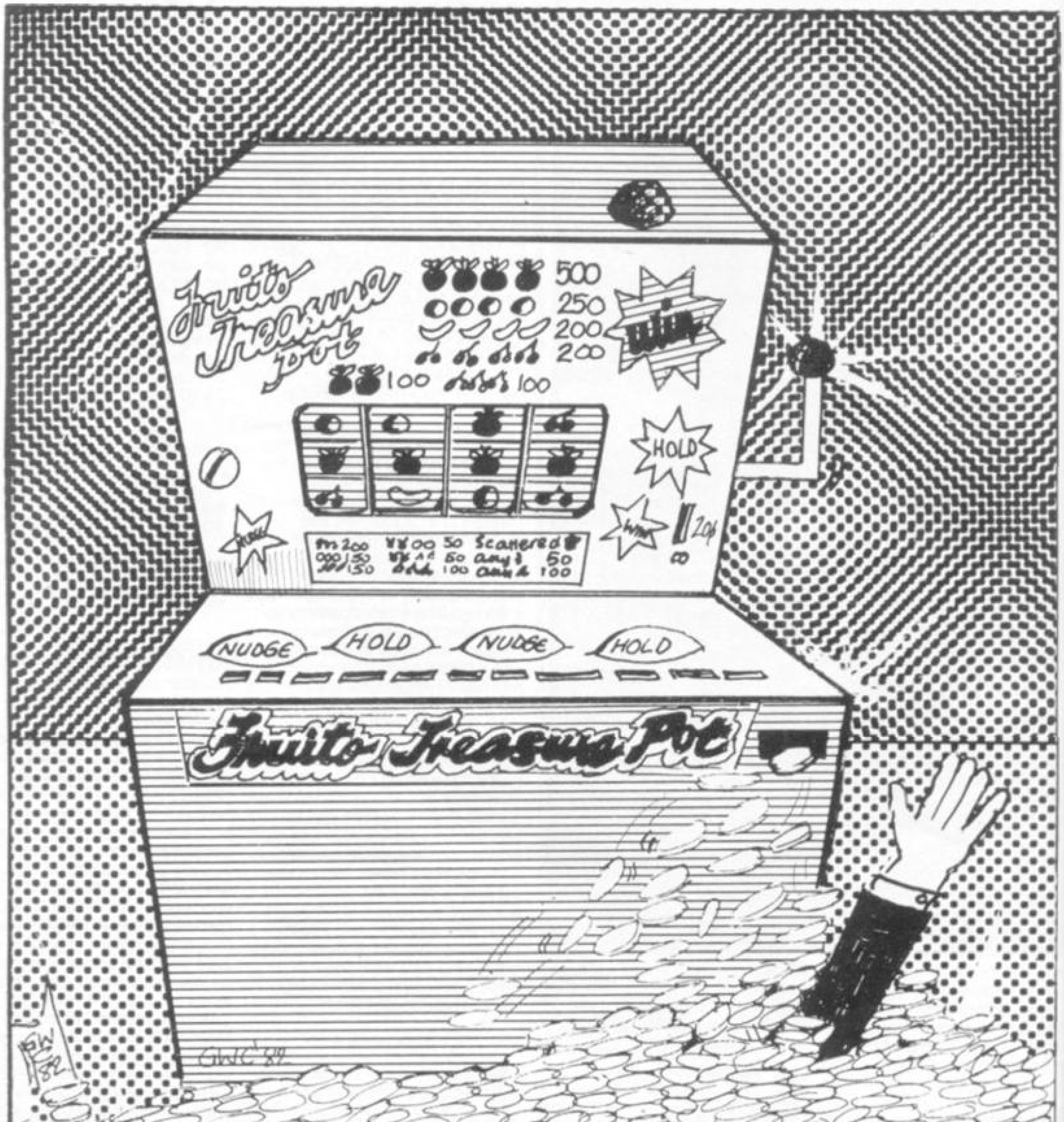
THIRD METHOD POKE
Lines 5 to 50 as above then:-

```
 60 LET B=33
 70 FOR A=0 TO 767 STEP
    B
 80 POKE Z+A+INT
    (A/32),8
 90 NEXT A
```

Conclusion

The first method of direct POKEing to the screen is faster than the "PRINT AT" by far. The only way to print faster would be to resort to extremes. A single line of "PRINT AT";"AT";"AT" to print the lot in one line or to print a string containing the pattern required, of course if you are prepared to do this, several lines of direct POKEs in succession will still work out fastest possible. The second method of looking to see if it is safe to POKE takes time and works out at about the same speed of the "PRINT AT". The third method using the formula, surprisingly enough, works out a little quicker in response to the "PRINT AT", as well as being safe. The second method in some cases may have an advantage even though it is slowest. This is that it detects the sides of the screen which can be used to bounce or deflect a moving object quite easily.

ONE ARMED BANDIT



Winnir

COIN
BELL
CASTLE
LEMON
CHERRY
ORANGE

COIN

BELL

CASTLE

LEMON

CHERRY

ORANGE

CHERRY

At random
will appear.
then choos
the barrels.
To hold bar
Press "Y"
To hold bar
Press "Y"
To hold bar
Press "Y"
e. To hold
barrel 3 Pr

List of
A(0) . . . .
A(1) . . . .
A(2) . . . .

W$ . . . . .
Y$ . . . . .
H$ . . . . .

S$ . . . . .
D$ . . . . .

This is a ZX80 version of the pub game. Three barrels are rolled on which are marked six symbols. According to the symbols displayed, different payments are awarded.

Winning Positions

COIN	COIN	COIN	66
BELL	BELL	BELL	55
CASTLE	CASTLE	CASTLE	44
LEMON	LEMON	LEMON	33
CHERRY	CHERRY	CHERRY	22
ORANGE	ORANGE	ORANGE	11
COIN	COIN	—	18
—	COIN	COIN	18
BELL	BELL	—	15
—	BELL	BELL	15
CASTLE	CASTLE	—	12
—	CASTLE	CASTLE	12
LEMON	LEMON	—	9
—	LEMON	LEMON	9
CHERRY	CHERRY	—	6
—	CHERRY	CHERRY	6
ORANGE	ORANGE	—	3
—	ORANGE	ORANGE	3
CHERRY	—	—	5

At random intervals, "HOLD" will appear. The player may then choose to hold any of the barrels.
To hold barrel 1
Press "Y"(else "N")
To hold barrel 2
Press "Y"(else "N")
To hold barrel 3
Press "Y"(else "N")
i.e. To hold barrel 1 and barrel 3 Press "YNY" N/L

List of Variables

A(0)	Result for barrel one.
A(1)	Result for barrel two.
A(2)	Result for barrel two.
C	Result for barrel one.
W$	String used for display.
V$	String used for display.
H$	String used for containing what is to be held 'held'.
G$	String used to check H$ is legal.
Q$	String used to stop programme.
I	Dummy variable.
J	Dummy variable.

```
 10 LET W$ = "                    "
 15 LET V$ = "      "
 20 LET C = 1000
 25 RANDOMISE
 30 DIM A(2)
 35 LET H$ = "NNN"
 40 INPUT Q$
 45 IF Q$ = "STOP" THEN STOP
 47 LET C = C-5
 50 FOR I = 0 TO 2
 55 IF CODE(H$) = 62 THEN GOTO 65
 60 LET A(I) = RND(RND(6))
 65 LET H$ = TL$(H$)
 70 NEXT I
 75 IF A(0) = A(1) OR A(1) = A(2) THEN LET C = C +
    3*A(1)
 80 IF A(0) = A(1) AND A(1) = A(2) THEN LET C = C +
    8*A(1)
 85 IF A(0) = 2 AND NOT A(1) = 2 THEN LET C = C + 5
 95 CLS
 98 PRINT "ONE ARMED BANDIT   M.R.HARRISON"
100 PRINT "      CREDIT $";C
105 PRINT
110 PRINT W$
115 FOR I = 0 TO 2
120 PRINT V$
125 IF A(I) = 1 THEN PRINT "ORANGE";
130 IF A(I) = 2 THEN PRINT "CHERRY";
135 IF A(I) = 3 THEN PRINT "LEMON";
140 IF A(I) = 4 THEN PRINT "CASTLE";
145 IF A(I) = 5 THEN PRINT "BELLE";
150 IF A(I) = 6 THEN PRINT "COIN";
155 NEXT I
160 PRINT V$,,W$
180 IF RND(5)<5 THEN GOTO 35
185 PRINT "      HOLD      "
195 INPUT H$
200 LET G$ = H$
205 FOR J = 0 TO 2
215 IF NOT (CODE(G$) = 62 OR CODE(G$) = 51) THEN
    GOTO 195
220 LET G$ = TL$(G$)
225 NEXT J
230 GOTO 47
330 STOP
```

OTHELLO

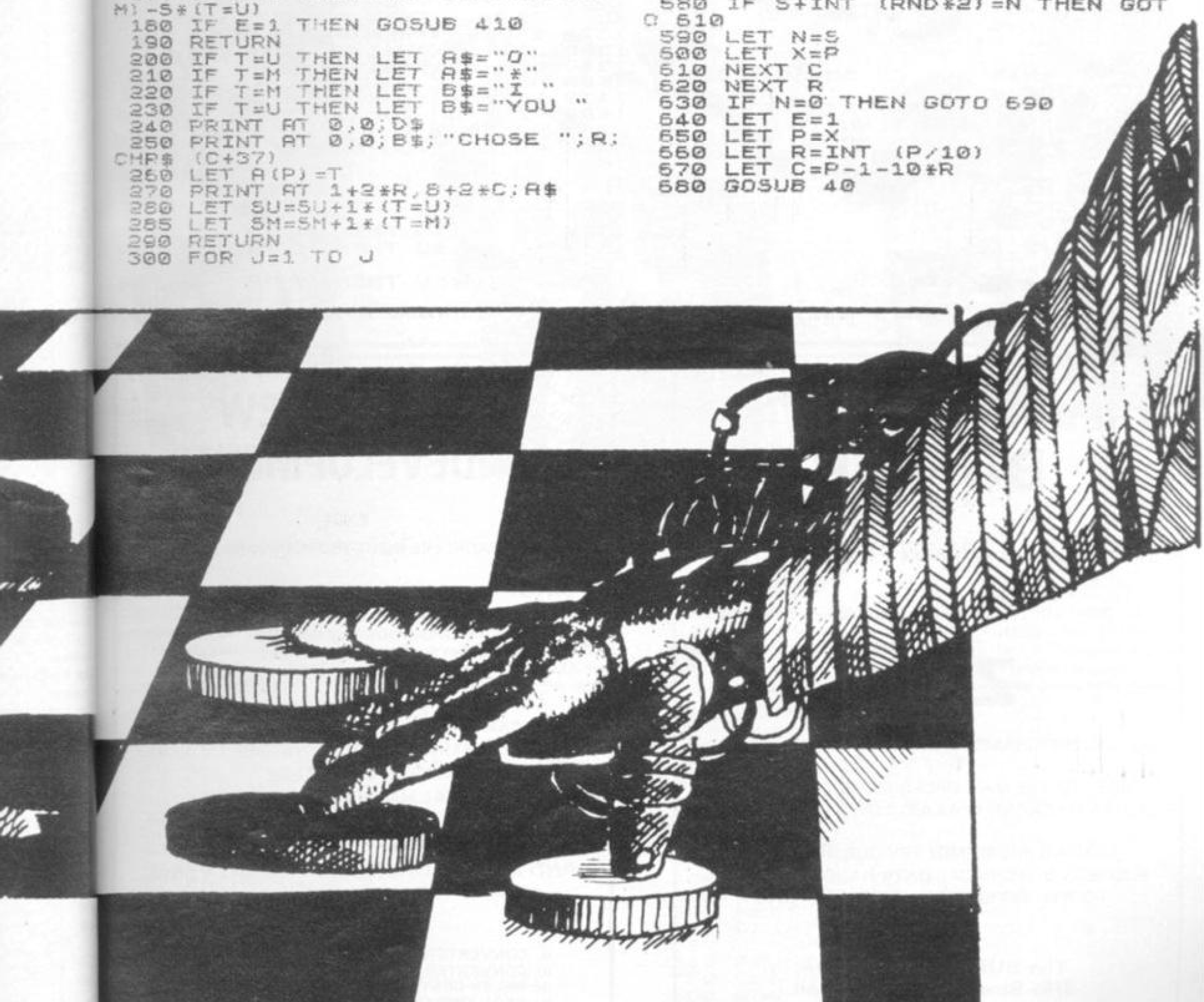
Challenge your ZX81 to this hundred-year old board game, in a program written by W P Davies of Bristol.

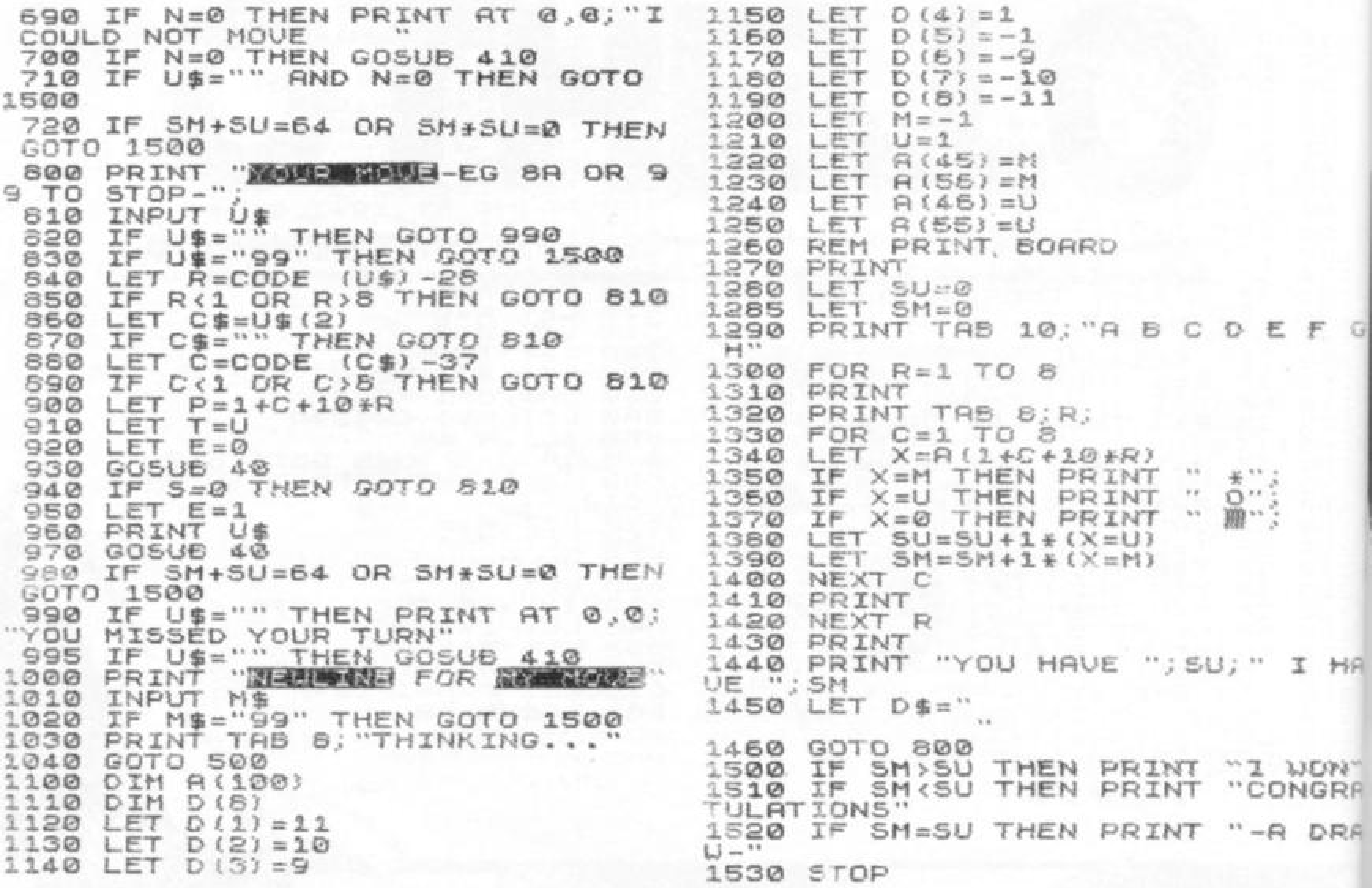
Othello was invented in the 1880s. The game is based on the older game Reversi, but has a restriction on the opening moves. Othello is played on a draughts board, with double-sided pieces. As you'll see at the beginning, there are four pieces on the board. You place your own piece so that at least one of the opponent's pieces is 'trapped' between a piece of yours and your new piece. The opponent's piece or pieces then flip over to become your pieces. The winner is the person with the most pieces on the board when the board is filled, or when neither of you can move.

The player's pieces are Os, and the computer's pieces are asterisks. The board is updated after each move, together with a confirmation of the move and the current score. There is also a cue for the next move, as well as an end-game routine.

If you want a faster game, add 675 SLOW, 1035 PAUSE 75, 1038 FAST. This modification blanks the screen while the clever ZX81 works out its next devastating move, reducing thinking time to around 15 seconds.

```
  10 REM "OTHELLO"
  20 SLOW
  30 GOTO 1100
  40 LET S=0
  50 IF NOT A(P)=0 THEN RETURN
  55 IF E=1 THEN GOSUB 200
  60 FOR I=1 TO 8
  70 LET Q=P
  80 LET J=1
  90 LET Z=D(I)
 100 LET Q=Q+Z
 110 IF A(Q)=0 THEN GOTO 170
 120 IF A(Q)=T THEN GOTO 150
 130 LET J=J+1
 140 GOTO 100
 150 LET S=S+J-1
 160 IF E=1 THEN GOSUB 300
 170 NEXT I
 172 IF E=1 THEN LET SU=SU+S*(T=
U)-S*(T=M)
 174 IF E=1 THEN LET SM=SM+S*(T=
M)-S*(T=U)
 180 IF E=1 THEN GOSUB 410
 190 RETURN
 200 IF T=U THEN LET A$="O"
 210 IF T=M THEN LET A$="*"
 220 IF T=M THEN LET B$="I "
 230 IF T=U THEN LET B$="YOU "
 240 PRINT AT 0,0;D$
 250 PRINT AT 0,0;B$;"CHOSE ";R;
CHR$ (C+37)
 260 LET A(P)=T
 270 PRINT AT 1+2*R,6+2*C;A$
 280 LET SU=SU+1*(T=U)
 285 LET SM=SM+1*(T=M)
 290 RETURN
 300 FOR J=1 TO J
 310 LET W=P+J*D(I)
 320 LET A(W)=T
 330 LET R=INT (W/10)
 340 LET C=W-1-10*R
 350 PRINT AT 1+2*R,5+2*C;A$
 380 NEXT J
 390 RETURN
 410 FOR I=0 TO 2
 420 PRINT AT 19+I,0;D$
 430 NEXT I
 440 PRINT AT 19,0;"YOU HAVE ";S
U;"   I HAVE ";SM
 450 RETURN
 500 LET N=0
 510 LET E=0
 520 LET T=M
 530 FOR R=1 TO 8
 540 FOR C=1 TO 8
 550 LET P=1+C+10*R
 560 GOSUB 40
 570 IF S<N THEN GOTO 610
 580 IF S+INT (RND*2)=N THEN GOT
O 610
 590 LET N=S
 600 LET X=P
 610 NEXT C
 620 NEXT R
 630 IF N=0 THEN GOTO 690
 640 LET E=1
 650 LET P=X
 660 LET R=INT (P/10)
 670 LET C=P-1-10*R
 680 GOSUB 40
```





```
 690 IF N=0 THEN PRINT AT 0,0;"I
 COULD NOT MOVE          "
 700 IF N=0 THEN GOSUB 410
 710 IF U$="" AND N=0 THEN GOTO
1500
 720 IF SM+SU=64 OR SM*SU=0 THEN
 GOTO 1500
 800 PRINT "YOUR MOVE-EG 8A OR 9
9 TO STOP-";
 810 INPUT U$
 820 IF U$="" THEN GOTO 990
 830 IF U$="99" THEN GOTO 1500
 840 LET R=CODE (U$)-28
 850 IF R<1 OR R>8 THEN GOTO 810
 860 LET C$=U$(2)
 870 IF C$="" THEN GOTO 810
 880 LET C=CODE (C$)-37
 890 IF C<1 OR C>8 THEN GOTO 810
 900 LET P=1+C+10*R
 910 LET T=U
 920 LET E=0
 930 GOSUB 40
 940 IF S=0 THEN GOTO 810
 950 LET E=1
 960 PRINT U$
 970 GOSUB 40
 980 IF SM+SU=64 OR SM*SU=0 THEN
 GOTO 1500
 990 IF U$="" THEN PRINT AT 0,0;
"YOU MISSED YOUR TURN"
 995 IF U$="" THEN GOSUB 410
1000 PRINT "NEWLINE FOR MY MOVE"
1010 INPUT M$
1020 IF M$="99" THEN GOTO 1500
1030 PRINT TAB 8;"THINKING..."
1040 GOTO 500
1100 DIM A(100)
1110 DIM D(8)
1120 LET D(1)=11
1130 LET D(2)=10
1140 LET D(3)=9
1150 LET D(4)=1
1160 LET D(5)=-1
1170 LET D(6)=-9
1180 LET D(7)=-10
1190 LET D(8)=-11
1200 LET M=-1
1210 LET U=1
1220 LET A(45)=M
1230 LET A(56)=M
1240 LET A(46)=U
1250 LET A(55)=U
1260 REM PRINT, BOARD
1270 PRINT
1280 LET SU=0
1285 LET SM=0
1290 PRINT TAB 10;"A B C D E F G
 H"
1300 FOR R=1 TO 8
1310 PRINT
1320 PRINT TAB 8;R;
1330 FOR C=1 TO 8
1340 LET X=A(1+C+10*R)
1350 IF X=M THEN PRINT " *";
1360 IF X=U THEN PRINT " O";
1370 IF X=0 THEN PRINT " ▒";
1380 LET SU=SU+1*(X=U)
1390 LET SM=SM+1*(X=M)
1400 NEXT C
1410 PRINT
1420 NEXT R
1430 PRINT
1440 PRINT "YOU HAVE ";SU;" I HA
VE ";SM
1450 LET D$="
          "
1460 GOTO 800
1500 IF SM>SU THEN PRINT "I WON"
1510 IF SM<SU THEN PRINT "CONGRA
TULATIONS"
1520 IF SM=SU THEN PRINT "-A DRA
W-"
1530 STOP
```

Swappo

Get your digits into order in the smallest number of moves.

This program was originally written by Don Scales, modified by H J Garwood to run on a TRS 80 and further modified by Tim Hartnell to make it run in Sinclair BASIC. Once you've got it working as listed, you might like to modify it so that instead of just stopping when you get the digits into order, the computer prints them out in inverse graphics, and gives you a congratulatory message.

A further modification would be a 'lowest score' feature. Pressing any key at the end will give you a new game.

```
  10 DIM A(9)
  20 FOR Z=1 TO 9
  30 LET A=INT (RND*9)+1
  40 IF Z=1 THEN GOTO 80
  50 FOR J=1 TO Z-1
  60 IF A(J)=A THEN GOTO 30
  70 NEXT J
  80 LET A(Z)=A
  90 NEXT Z
  95 LET B=0
  97 PRINT AT 6,6;
 100 FOR S=1 TO 9
 110 PRINT A(S);
 120 NEXT S
 130 PRINT
 200 PRINT
 205 PRINT "ENTER NUMBER TO REVE
RSE, MOVE ";B+1
 210 INPUT J
 215 PRINT AT 3,0;"
         "
 220 IF J<1 OR J>9 THEN GOTO 210
 260 LET K=(J+1)/2
 270 FOR Z=1 TO K
 280 LET A=A(Z)
 285 LET A(Z)=A(J+1-Z)
 287 LET A(J+1-Z)=A
 290 NEXT Z
 295 LET B=B+1
 300 FOR Z=1 TO 9
 310 IF A(Z)<>Z THEN GOTO 97
 320 NEXT Z
 330 PRINT "IT TOOK YOU ";B;" MO
VES"
 340 PAUSE 4E4
 350 CLS
 360 RUN
```

PEEK, POKE

Peek, Poke, Byte and RAM!

Phil Garratt looks at this flashy book, with cover art by the man who produced the cover of Clive's manual.

The cover of this well-presented book gives a clue as to the audience it is aiming at. Drawn by John Harris, who drew the ZX81 manual cover, it shows the same spaceship, but this time in flight. The book clearly intends to give further assistance to the new owner in getting his or her machine 'in flight' — or at least up and running.

The book assumes no previous knowledge of computing, and starts right from the beginning with setting up the ZX81 and adjusting your TV. This is all done in a light-hearted ("On no account use Channell No. 5, it stinks!") but thorough manner and avoids merely repeating the manual. The book goes on to cover arithmetic assignments, variables, loops, loading and saving (with photographs to show you what your telly should look like), graphics, plotting and debugging. With the exception of debugging, which is explained in some detail, all these areas are covered with just sufficient information to ease the beginner into writing his or her first programs. To help maintain interest and give blistered fingers a rest, the book includes lovely little "De Bugs" strip cartoons and a ZX crossword.

There are over 20 complete programs contained in the book, plus a number of useful routines. These programs, an the suggestions for programm ing exercises tend towards mathematical bias; factorials areas, solving equations an the like. There are a few games as well, such as Molehunt an Hangman, though nothing very substantial or new. The programs all fit into 1K, and all tha I tried were bug-free and wel documented.

This is a well-written and at tractively laid out book, which has deliberately (and success fully) avoided becoming ye another rehash of the manual. think it would be of grea assistance to the new Z owner, even though the boo would be outgrown in a coupl of months. To experienced Z users I would suggest that nex

BYTE and RAM

time you convince somebody to buy a ZX81, get them to buy the book as well . . . and then borrow it. The cartoons are great, and who knows, you may pick up a trick or two. "PEEK, POKE, BYTE and RAM", Shiva Publishing Ltd, ISBN 0 906812 178.

ZX81 BASIC Book

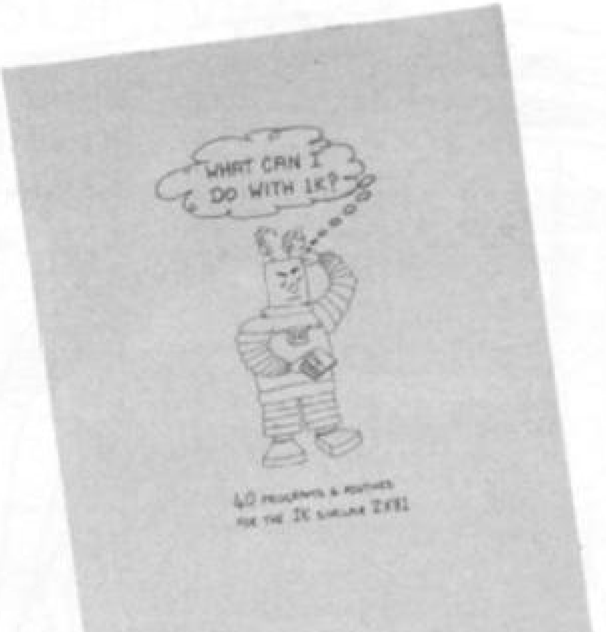
The ZX81 BASIC Book, published by Newnes Microcomputer Books, is more staid in approach than the others reviewed in this section of *ZX Computing*, and for that reason is sure to appeal to schools. Although the approach is fairly straight, the book is far from dull, with witty (?) chapter titles like "Gone out, bizzy, back soon" to introduce subroutines, and "Graphics ride again!"

The book methodically covers the ins and outs of the ZX81 starting with general information on what computers can do, followed by a short introduction to computer languages and binary arithmetic, and then a brief section on what a program is — using a sample 'program' describing certain actions by Mickey Mouse in a Walt Disney cartoon. Once you've traversed this ground, and worked out how to plug your computer into the telly, the book gets down to work. Direct input commands are covered, and — in a section sure to confuse newcomers who don't have a maths background, or don't want to get involved in such things at such an early stage — then the priority of mathematical operators (such as multiplication before addition) is discussed. Already I can sense newcomers flipping past this section in exasperation, looking for something a little more directly relevant to their needs.

We are already up to chapter eight (some of the chapters are only one or two pages long) before the first particularly useful information for first-time bewildered users is presented. The use of LET to assign values to variables is explained, followed (in subsequent chapters) by such things as the use of commas in PRINT formatting, the use of the EDIT function, trigonometrical functions (you can see this is just the sort of book your teachers would leap on), and GOTO and FOR/NEXT loops.

The book would be, I believe, fairly heavy going, despite its simplicity, for a person who has just picked up a computer for the first time, but when used as a text to guide pupils who have the benefits of a live teacher on the spot, would be very useful indeed. It is difficult to imagine why the material has been presented in this order, with GOTO assigned a lower priority than converting radians to degrees, and loading and saving programs is considered less important than finding natural logarithms of square roots.

If you are teaching a class of



14-year olds the rudiments of computing using a ZX81, and you want to do it 'by the book' this is the book to buy, but if you're not . . . have a good look at it in the shop before you decide. ZX81 BASIC Book, Newnes Technical Books, ISBN 0 408 01178 5.

The ZX81 Pocket Book

Trevor accurately sums up the thinking behind the book, and explains how it differs from the ZX80 Pocket Book, in his introduction: "This is our second book along these lines . . . although the style is hopefully less clinical . . . Newcomers to computing will hopefully find something more challenging than just another set of games to copy and run . . . I have . . . tried to expand on the 'useful subroutines', as I think these give a much clearer picture of how certain features can be highlighted and how you can stretch the ZX81 to its utmost . . ."

The programs in the book are good, and varied. They vary from thė good-for-a-weak-laugh PIN THE TAIL ON THE DONKEY to a TUNNELS AND TROLLS CHARACTER GENERATOR, with some maths (STANDARD DEVIATION) and machine code (DICE SIMULATION) along the way.

A number of additional 1K programs are scattered throughout the book. The approach of the book is, as Trevor said in his foreward, 'less clinical' than his earlier book, so string handling is up near the front, and clearly and simply explained.

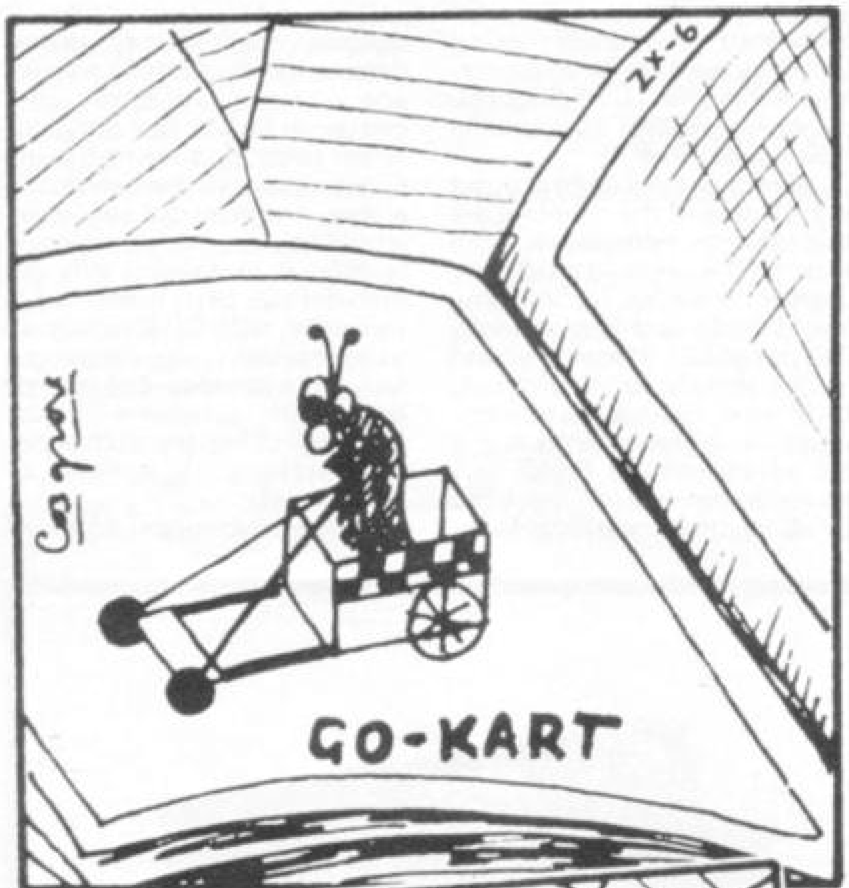
You'll pick up a number of useful ideas which you'll want to use in many programs. My favourite, and one which I see over and over again in other people's programs, showing how well the lesson has been learned, is the use of PAUSE 4E4 (which Trevor suggests you memorise as PAUSE forever) at the end of a game to freeze the action until any key is retouched to give you a replay. The ZX81 Pocket Book, Phipps Associates, ISBN 0 950 7302 2 X.

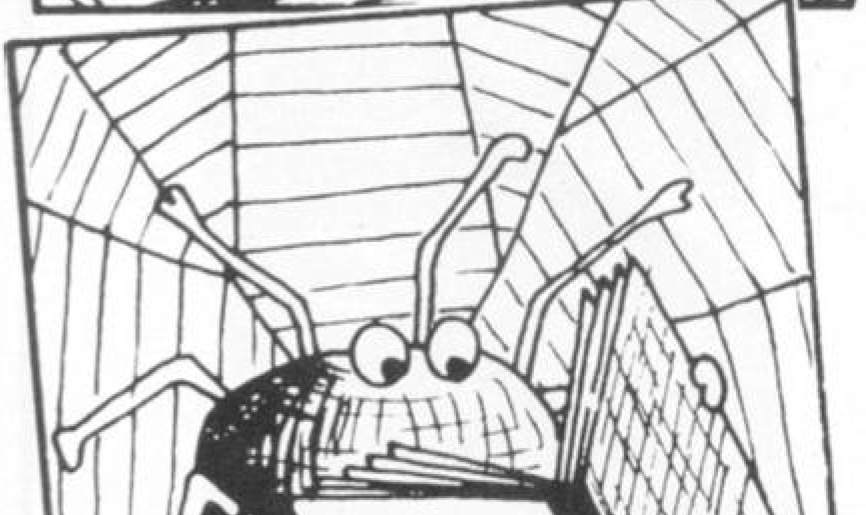
What Can I Do With 1K?

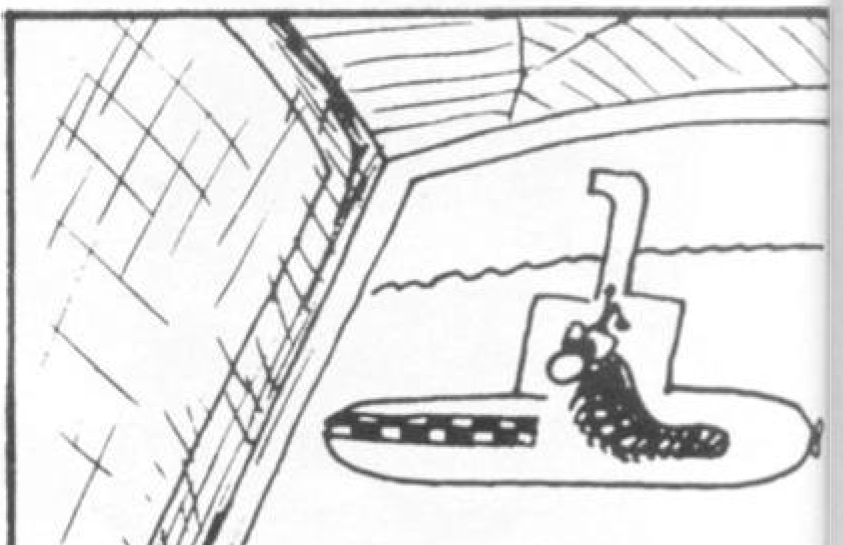
The ZX81 publishing bandwagon is one which a number of people — including myself — have jumped onto, with mixed results. Some of the books have a 'bad feel' to them, as they were produced just for the money, while others feel good and deliver the product their covers proclaim. This book is one that delivers.

When I first reviewed the ZX81, I was highly critical of the miniscule 1K provided on board, but along with many others, I have since then learned a number of tricks to squeeze the maximum into Clive's unforgiving RAM. Roger Valentine has also learned the lessons of program packing and some of the best of them are presented in this little volume.

The programs are divided into seven sections: Fate and fortune; Printing with more frills; Casino gambling; Data files; Business programs; Utilities and Games. The section titles do not even hint at how wide the program selection is in the book. Programs include the predictable standards (CRAPS, I CHING and SPACE RESCUE) as well as a number of the surprisingly effective and useful business programs — VAT, Bank Record, Credit Cards, Payroll, Test Data and Payslip. Some of the games are very good indeed, such as Roulette and there is even a very odd "computer dating" simulation which enables you to enter your intelligence as 'thick as two short planks'. All in all this is a splendid book, and one which will repay your investment time and time again. What Can I Do With 1K?, V & H Computer Services.







AREA CALCULATOR

The following program, produced for the ZX80 and using only the 1K of RAM, will calculate the area within a polygon. The computer plots the points whose co-ordinates have been entered, using a 36 symbol code, and the area is printed underneath. The sign of the area will be positive if traced anti-clockwise and negative if all, or any, of the figure is plotted clockwise. On the display the plotted points are displayed in the same code as the axes and are numbered according to the order of entry.

In the entry stage of line 80 and 90, inputs of greater value than 32 and 21 respectively will not be displayed although the area is still calculated. All the input co-ordinates are repeated after entry for checking, NEWLINE will cause the program to continue if they are correct.

```
 10 PRINT "HOW MANY VERTICES?"
 20 INPUT V
 30 CLS
 40 PRINT "GIVE CO-ORDINATES"
 50 DIM X(V-1)
 60 DIM Y(V-1)
 70 FOR N=0 TO V-1
 80 INPUT X(N)
 90 INPUT Y(N)
100 PRINT "(";X(N);",";Y(N);")"
110 NEXT N
120 INPUT A$
130 CLS
140 FOR N=1 TO 20
150 LET Y=21-N
160 PRINT CHR$(156+Y)
170 FOR P=0 TO V-1
180 IF Y=Y(P) THEN GOTO 210
190 NEXT P
200 GOTO 290
210 FOR X=1 TO 31
220 FOR Q=0 TO V-1
230 IF Y=Y(Q) AND X=X(Q) THEN GOTO 270
240 NEXT Q
250 PRINT " ";
260 GOTO 280
270 PRINT CHR$(Q+156);
280 NEXT X
290 PRINT
300 NEXT N
310 FOR N=0 TO 31
320 PRINT CHR$(156+N);
330 NEXT N
340 LET Z=0
350 FOR R=0 TO V-1
360 LET S=R+1-((R+1)/V)*V
370 LET T=R+2-((R+2)/V)*V
380 LET Z=Z+X(S)*(Y(T)-Y(R))
390 NEXT R
400 PRINT "AREA IS";Z/2;
410 IF NOT (Z/2)*2=Z THEN PRINT "1/2";
420 PRINT "SQUARE UNITS"
```

Clive sets the pace

Uncle Clive will be donning his sweaty tracksuit to set the pace in a half-marathon to be held in July.

The half-marathon, to be included in this year's Cambridge Festival, will be sponsored for £5000, by Clive's company — Sinclair Research — which is, of course, based in Cambridge. The race is expected to attract a number of top-class runners to the city. Clive will be among an expected 2000 competitors following a three-lap course through the city's historic streets. Starting in King's Parade, close to Sinclair's offices, it includes a riverside stretch along Chesterton Lane, the 'Backs', and finishes in the Market Square at the Guildhall Steps.

Co-organised for the Festival by Cambridge City's recreation department and the Cambridge and Coleridge Athletic Club, the event is open both to club-based runners and general enthusiasts. Entries have already been received from as far afield as Wales and Durham.

Uncle Clive told us his company sees its sponsorship of the marathon as part of its commitment to supporting and developing cultural life of the city. It is also sponsoring, as part of the Festival, a concert in King's College on July 31.

● *Clive Sinclair*

Sinclair miss the bus

Sinclair and Timex, were most conspicuous by their absence at the giant West Coast Computer Faire held in San Francisco at the end of March.

The booth listed in the catalogue as being the Sinclair one was unmanned throughout the show, the most important computer show in the world. Some 40,000 people crammed into a show with over 600 stands, to be confronted at the alleged Sinclair booth with a tiny, handwritten notice saying that Clive's people would not be there.

I found a disconsolate set of four ZX81 owners — who are very much in a minority among computer users in America — sitting sadly in the booth swapping 'Zee-X81' stories.

One enterprising ZX81 owner in the States, Eric Reiter of 16th Avenue, San Francisco, had a tiny, one-yard wide booth, in which he was showing his expansion board, which is suitable for ZX80s. ZX81s and the ZX80-lookalike, the MicroAce. The expansion board was a complicated, spaghetti-junction type motherboard which was controlling little lights and squawkers.

A series of seminars were held throughout the show. One talk was given by the president of Mindware, Michael Levy. It was fascinating to hear questions from those present at the talk who had never heard before of the ZX81. Michael tried to explain the excitement the ZX81 was generating in the UK, but I could see his words were received with some scepticism. When he told the seminar about the crushing crowds at ZX Microfairs, the disbelief reached fever pitch.

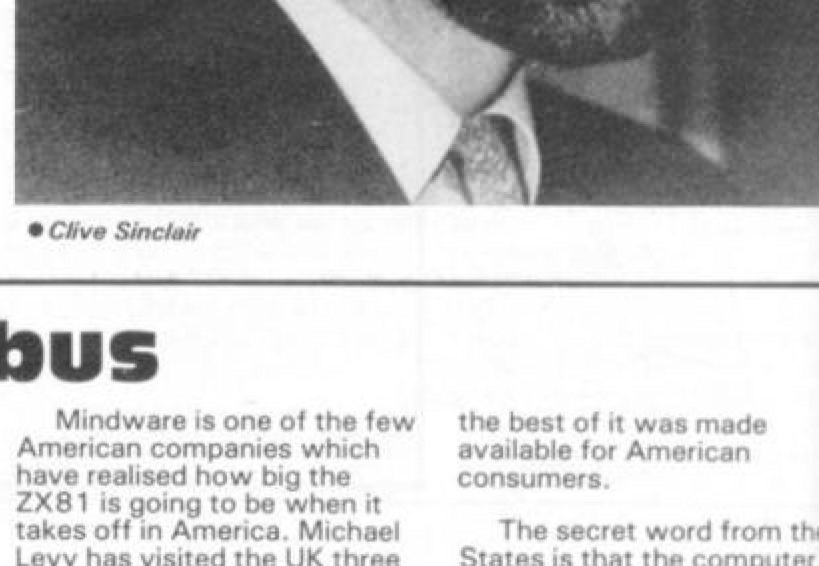
Mindware is one of the few American companies which have realised how big the ZX81 is going to be when it takes off in America. Michael Levy has visited the UK three times in the last few months to sign rights and distribution deals for UK products. He says that the standard of UK software, hardware and publications is very high, and thus was anxious to ensure the best of it was made available for American consumers.

The secret word from the States is that the computer will not be known as the ZX81 when it is launched there by Timex. The most likely name is 'Timex 1000' or if, as appears possible, the computer is sold with 2K on board, the 'Timex 2000'.

Schools lap it up

More than 2500 UK secondary schools have bought ZX81s to give their pupils hands-on computing experience.

Last May, Sinclair Research worked out a deal with the educational distributors Griffin and George to sell ZX81s at a cut price to schools. The scheme, which is now closed, aimed to provide a wider and more economic choice of equipment than was available under a government-assisted scheme for the purchase of computers. Although the government thought it was wonderful for schools to buy micros, it was not prepared to give cash for the purpose to schools that insisted on buying a ZX81.



Putting it diplomatically, Uncle Clive said: "Although welcoming the government initiative, we felt it did not fully account for the needs of all schools."

"We believe that the success of our scheme in bringing microcomputers to many schools vindicates our approach as both practical and economic."

●*Lem Tarshis, Executive Vice-President in charge of operations, Timex, at the press conference announcing his company's tie-up with Sinclair Research*

Timex buys it up

Sinclair Research have signed a deal with Timex in America to get the ZX81 sold through some 171,000 retails outlets in the States. Very shortly, lucky Americans will be able to buy ZX81s from the same shops which sell jewellery, perfume and Timex watches.

Uncle Clive will get a royalty on all sales under the new contract, which covers current and future Sinclair personal computer products, and Timex's own development of Sinclair computer technology.

It also includes the use of Uncle C's version of BASIC, although the pound sign on the ZX keyboard has raised a few eyebrows in the States. Apparently the Yanks were wondering what in hell this squiggly little symbol meant. The best guess that it was a new form of mathematical symbol which the British understood, but which hadn't yet been taught in progressive American schools.

Sinclair Research told ZX COMPUTING that Timex's marketing expertise meant that the ZX81 would make much greater inroads into the US market than has been the case in the past. And, with a high-tech background — ranging from cameras to gyroscopes — Timex reckon they'll quickly become a viable competitor in the personal computer field.

For the time being, Sinclair's Boston-based US subsidiary will continue to sell the '81 by mail order until Timex's own sales reach agreed levels. The Boston group will then concentrate on selling the new flat-screen telly. This is expected to be available just before Christmas.

Users' clubs

A network of users' clubs for the ZX81 has sprung up in the UK. The largest group, the National ZX80 and ZX81 Users' Club, produces a monthly magazine Interface, acts as an 'umbrella' club to publicise local groups, and assists in the presentation of the ZX Microfairs. The national club is at 44-46 Earls Court Road, London, W8 6EJ, and you can get a sample issue of Interface, along with details of the club, by sending £1.00 to the Earls Court Road address.

The North London Hobby Computer Club has a ZX80/81 users' group meeting each Monday night from 6-9 p.m. It is held at the North London Polytechnic, Holloway Road, London, N7 (diagonally opposite Holloway Road tube station).

Most local computer clubs have a large percentage of ZX81 owners, so you are sure to meet fellow enthusiasts if you get in touch with your local club. If you'd like to start a local club, write to the National ZX80 and ZX81 Users' Club so they can publicise your address. It will also be listed in future issues of ZX Computing.

Local clubs that we know about include:

- ● EZUG (Educational ZX80/81 Users' Group), Eric Deeson, Highgate School, Birmingham 12. Send a large, stamped, addressed envelope for details. EZUG also caters for the BBC Microcomputer.
- ● Roger Pyatt, 23 Arundel Drive, Orpington, Kent (66) 20281.
- ● Austin Knott, 269 Telegraph Road, DEal, CT14 9EJ.
- ● Christoph Moeller, Gross Kurfurstenstrasse 41a, 4800 Bielefeld 1, Germany.
- ● Danmarks Nationale ZX80 og ZX81 Club, Skovmosvej 6, 4200 Slageise Dk Denmark.
- ● Steve Brumby, 38 Eastfield Road, Messingham, Scunthorpe, Sth. Humberside.
- ● Ken Knight, 22 Mount Street, Aylesbury, Bucks. HP20 2SE (0296 5181).
- ● David Blagden, PO Box 159, Kingston upon Thames, Surrey, KT2 5YQ.
- ● Anthony Quinn, Heckenrosenweg 6, 3170 Gifhorn, W. Germany.
- ● Conrad Roe, 25 Cherry Tree Avenue, Walsall, WS5 4LH.
- ● Ian Watt, 107 Greenwood Road, Clarkeston, Glasgow.
- ● J. Palmer, 56 Meadowfield Drive, Edinburgh (031-661 3181).
- ● Leeds Microcomputer Users Group. Meets fortnightly on Thurs eve in Leeds, new members welcome. Contact: Paul O'Higgins, 20 Brudenell Mt, Leeds 6, tel: (0532) 742347 after 6.
- ● Brunel Computer Club: meets alternate Mondays 1900-2200 hrs at St Werburgh's Community Centre. Contact: Mr R Sampson, 4 The Coots, Stockwood.
- ● Worle Computer Club: meets alternate Mondays 1900-22.30 at Woodsprings Inn Function Rooms. Contact: S Rabone, 18 Castle Rd, Worle, Weston-Super-Mare, Avon, tel: 0934 513068.
- ● P Compton, 29 North Marine Road, Scarborough, Nth Yorks, YO12 7EY.
- ● Jonathan Meyer, Vanspaen Straat 22, 6524 H.N. Nymegan, Holland.
- ● Royston H Wallis, 22 Mallard Crescent, Pagham, Bognor Regis, West Sussex, PO21 4UU.
- ● Raymond Betx, Chemin du Moulin 38, 1328 Ohain, Belgium.

Users crowd it out

The ZX Microfairs, organised by Mike Johnston, are going from strength to strength.

The first one, last September, attracted about 4000 restless ZX81 owners, who surged seven-deep around exhibitors' stands in one half of one hall at the Central Westminster Hall. The three-hour queue — in the rain — infuriated many would-be attenders, so Mike decided to double the space for the second show, held at the end of January.

He needed to. The *New Scientist*, commenting on the second show, said that in terms of crowd concentration, the show gave the Royal Wedding a run for its money. Some 12,000 people came the second time, again there was a three-hour queue, and again the crowds were seven or so deep around the stands, even though there was double the floor-space, and the show was open for three hours longer than the first one had been.

For the third show on Friday, April 30 and Saturday, May 1, Mike is determined to avoid the queues, and make sure all those attending have a good time.

The main attractions at the second show — add-on hardware bits to get ZX81s to sing and dance, to draw real SPACE INVADERS and user-defined graphics — will be out in force at the third show, as well as part of the rapidly expanding output of books, software and selection of books and software.

Based on the catalogue of the second show, it seems that a new ZX-based company has evolved every day and a half since the launch of the little wonder.

DCP Microdevelopments' Packs

Stephen Adams takes a peek and a poke inside this system which is based on a single input/output port and intended for a wide range of educational and control applications. It can be used — with the other packs available to plug into the port — to monitor heat, light and voltage, as well as activating alarms and motors, and making sounds.

The Basic 'P' Pack

This is 4K of dynamic RAM (2 x 6116s) to add to the basic 1K provided with the machine. This gives a total of 5K, the maximum amount of memory allowed by the system. It also provides separate input and output ports at the back of the 'P' pack which are memory mapped. Because the port is memory mapped, it can be PEEKed and POKEd to get information to and from the computer. It makes no difference if the computer is a ZX80 or a ZX81 and it will work with a 4K or 8K ROM. It is not possible to connect anything else to the computer without using a motherboard as the only other connectors on the 'P' pack are 0.1 molex pins for the port. These pins are about ¼ inch long and there are 10 for input and 10 for output. The pins are connected to the eight data lines and also provide + 5 and 0 volts. They can only drive one TTL gate from each output pin.

The ports address is anywhere from 21504 to 22527 as the single port replaces 1K of RAM. The plug connections used for the port are readily available from other sources than DCP.

The port can be demonstrated by the use of switches and LEDs. A suitable circuit diagram is included in the leaflet that comes with the port. It is very easy to use and comes in a robust black plastic box 3½ x 3¼ x 1¾ inches. Demonstration programs are given, and with the accompanying information any user should find it easy.

This is a basic building block for DCP's control system and although it provides only 5K of memory, this should be more than enough to control most devices. The port connectors on the back are designed to be used by two other packs 'A' and 'C' to provide extra facilities, but only one pack can be used at a time.

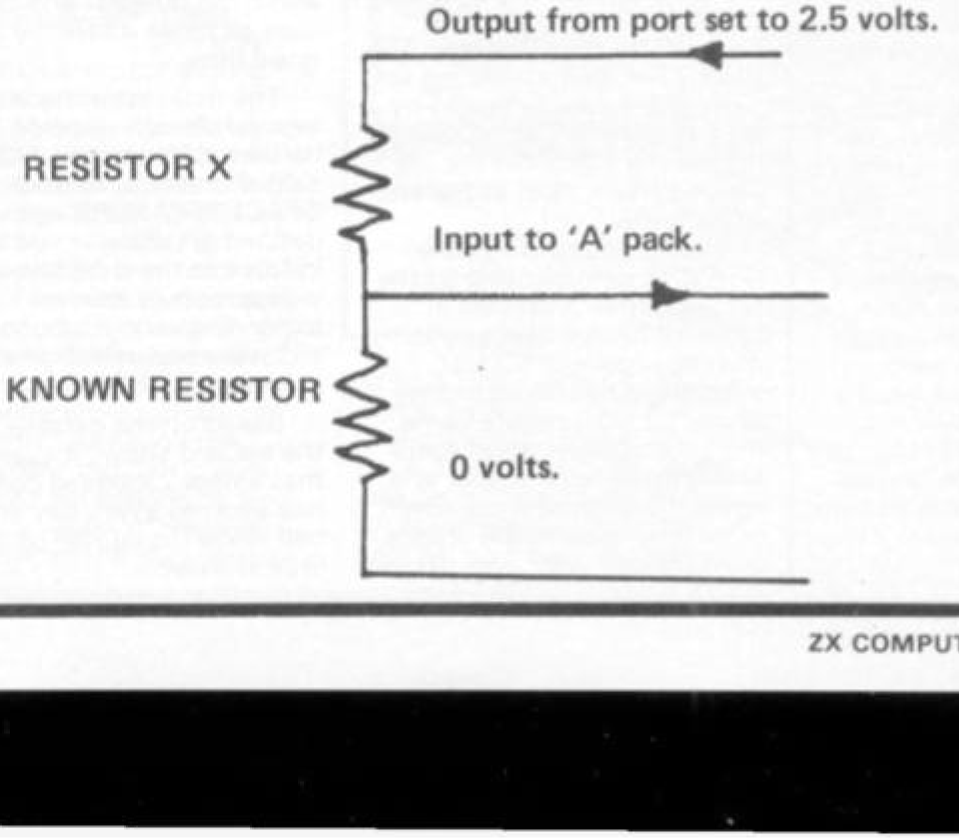
The 'A' Pack

The 'A' stands for analogue and the pack provides an interface between the digital world of the computer and the real world we live in.

The 'A' pack requires a 'P' pack or another port to work as it cannot plug directly into the expansion port of the computer. It is in a tough black box like the 'P' pack, but with the input and output sockets 2 mm wander sockets. These wander sockets are the same as used on some multimeters. They also take the connection plugs used by schools and colleges to connect up science experiments.

The 'A' pack is used for measuring a voltage between 0

Fig. 1



and 2.5 volts. This will produce a number between 0 and 255 on PEEKing at the INPUT from the port. There are +5 volts and 0 volts connections from the pack, so these can be used to create the voltage required. These should be used with care as they may cause the crash of the computer if too much current is taken.

The output of the Digital to Analogue converter will convert the number between 0 and 255 into a voltage between 0 and 2.55 volts. This can be adjusted within the port, if it is not correct, and a complete circuit diagram is given. The accuracy is ±5% on both the analogue input and output.

The resistor could be measured using this device by connecting them as shown in Fig. 1. By knowing the voltage across the known resistor and the voltage output by the analogue port, the resistance can be calculated. As the resistance of the unknown resistance rises, the measured voltage across the known resistor will fall. The computer can then be used to work out the unknown resistance.

The 'C' Pack

The 'C' pack again requires the use of a 'P' pack or another port. The 'C' pack can plug directly onto the connectors at the back of the 'P'.

The 'C' pack provides eight relays which can be controlled by the output of the port. The relays are turned on by making any of the eight bits Binary 0. The outputs from the back are by the same 2 mm sockets as used on the 'A' pack. The outputs consist of eight single contact switches, which connect to a common wire. They are not connected to the computer at all. The contacts are normally closed and can carry 12 volts at 1 amp maximum. Therefore when the computer is switched on and the output from the port is 255, all the switches are closed. I would have thought this would be a disadvantage in a controlled situation, as all the devices would be turned on under no control from the computer.

The inputs consist of a set of eight resistors connecting each of the input connectors to 0 volts. This means that when PEEKing the port with nothing connected the user will see 0. There is no buffering between these input pins and the input to the port, so that the application of any voltage above +5 volts could not only damage the 'C' pack, but also burn out the 'P' pack as well.

These devices are intended for use in educational or control purposes. Games could be another possible use, but the price of these packs might put that out of bounds. The limit of only 5K of RAM and only one port might prove restrictive in use, but up to 128 different 1 bit devices can be controlled.

The boxes are robust and could be used in schools without needing a set of expensive plugs and sockets. I have doubts, however, over the inputs to the 'C' pack. If any voltage over +5 volts is used, some form of protection will be required to stop the port blowing up.

The documentation that comes with the port is very clear and simple to understand. The leaflet included with each pack details its uses and gives a program to demonstrate its capabilities. The current drawn by these packs is shown in table 1. DCP packs cost £37.95 for the price of the 'P' pack, £29.95 for the 'A' pack and £19.95 for the 'C' pack. Both 'A' and 'C' packs must be used with a 'P' pack or other ports.

Table 1.

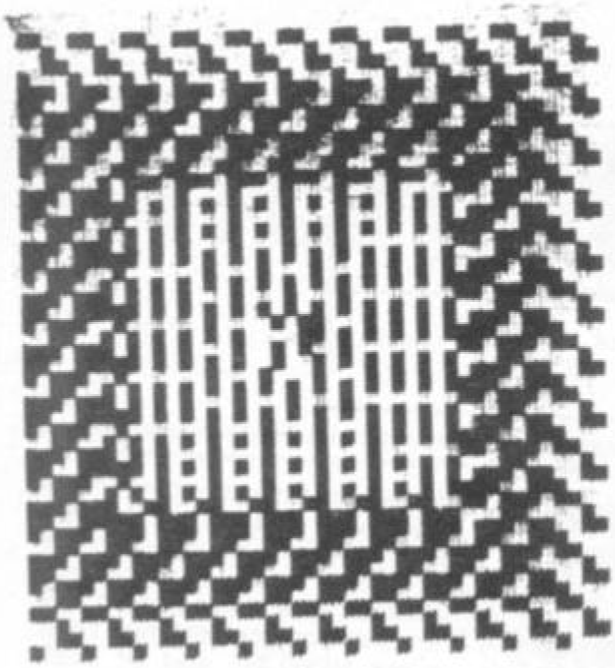
Port 'P' pack	45 ma.
'A' pack	80 ma.
'C' pack	775 ma. (Max.)

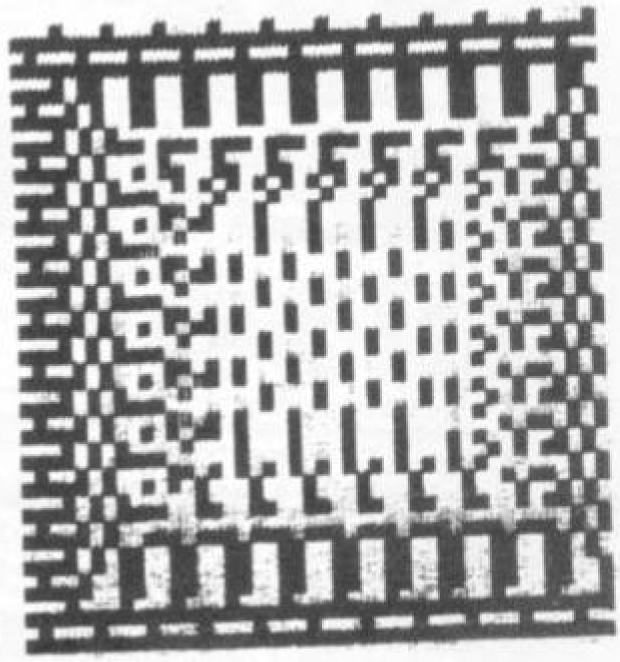
Solving Equations

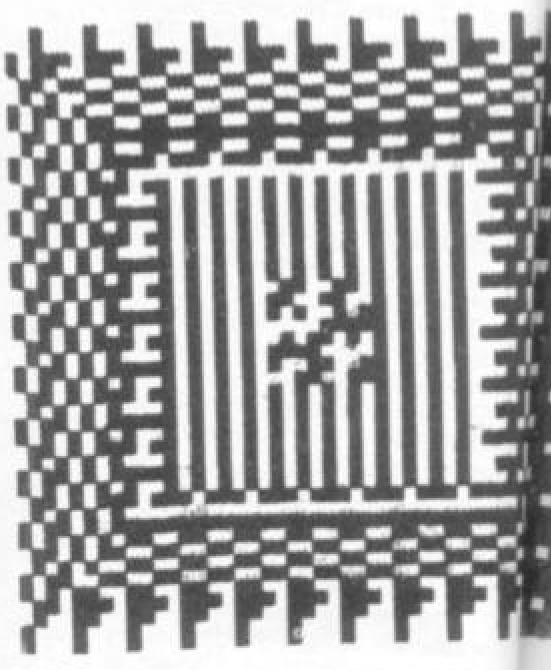
This clever routine, written by Jeremy Ruston, uses Newton's method for solving equations. Enter the equation you want solved for X when prompted by line 10, then — in reply to the prompt from line 40 — enter a starting position for the ZX81 to work from. This should be either an answer somewhere near what you believe the correct answer to be, or — if there is more than one correct answer — a number near the answer you are seeking. Then press NEWLINE and sit back and watch the fun as the computer verges towards the answer.

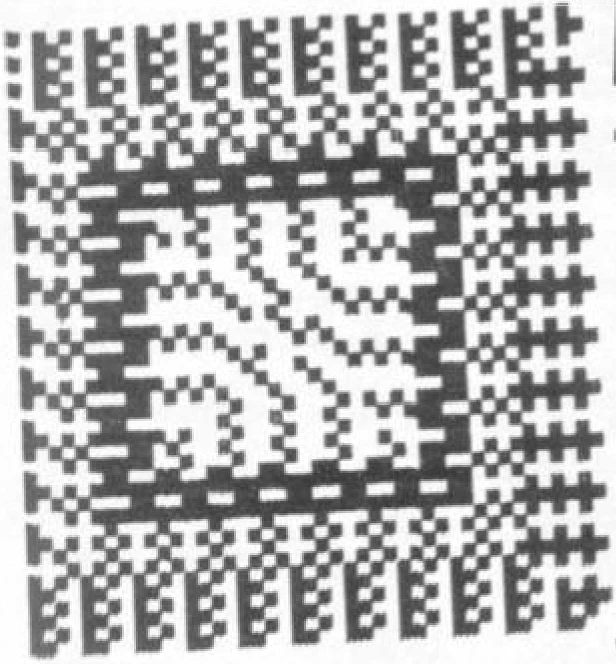
To try it out, enter X*X−5 (to find the square root of five) or X**3−27.6 to find the cube root of 27.6.

```
   5 REM NEWTONS METHOD FOR
         SOLVING EQUATIONS
   6 REM BY JEREMY RUSTON
  10 PRINT "ENTER A FUNCTION ";
  20 INPUT F$
  30 PRINT F$
  40 PRINT "ENTER A STARTING POI
NT ";
  50 INPUT S
  60 PRINT S
  70 PRINT "INPUT MAXIMUM ERROR
";
  80 INPUT ERR
  90 PRINT ERR
 100 PRINT AT 10,10;S
 110 LET X=S
 120 IF ABS (VAL (F$))<ERR THEN
STOP
 130 LET T=VAL (F$)
 140 LET X=X+0.00001
 150 LET B=(VAL (F$)-T)/0.00001
 160 LET S=S-T/B
 170 GOTO 100
```







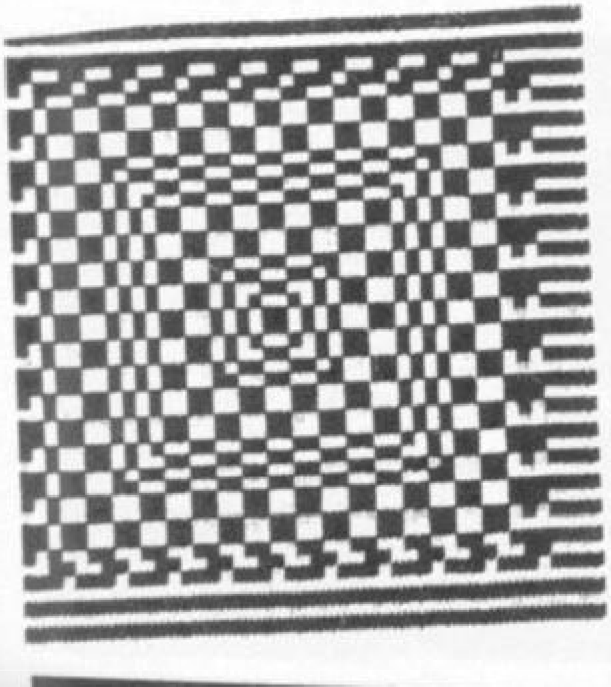


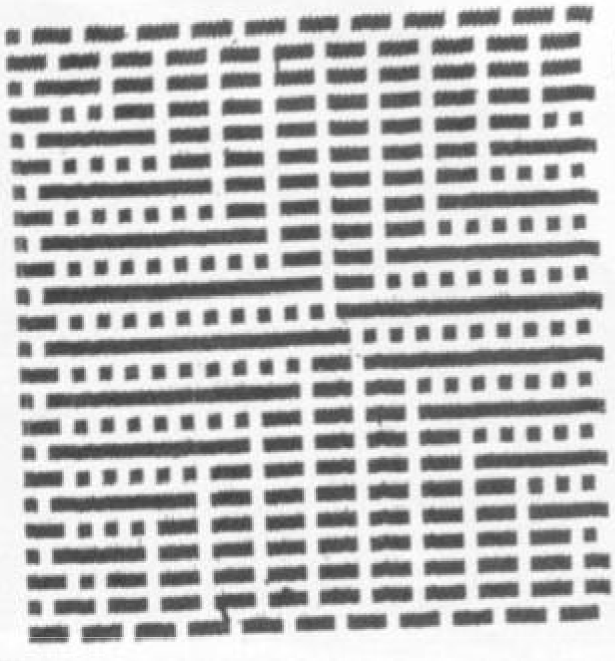
Hypno

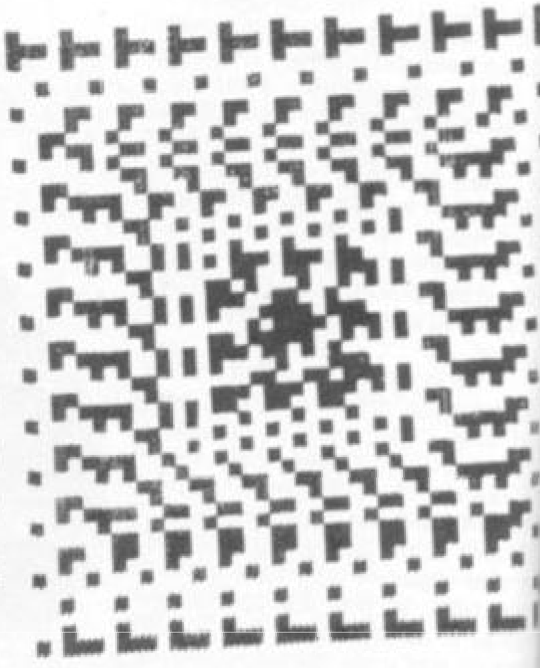
Tim Johns from Birmingham has produced a routine which shows off the graphics capability of the ZX81 to good effect, creating a constantly changing square pattern on the TV screen.

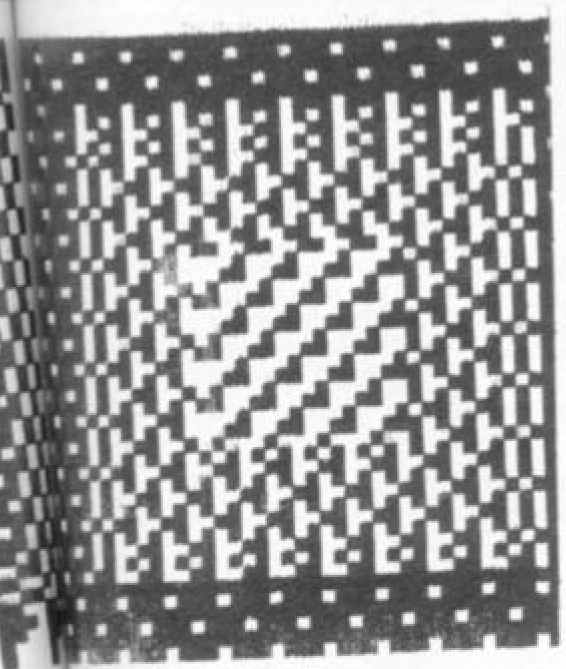
This program, instead of using the obvious technique of PRINT, makes use of POKE, putting the code values of the characters directly into the display file, the start of which is identified in line 10. By doing this, you get a 24 x 24 display, whereas using PRINT restricts you to 22 lines.

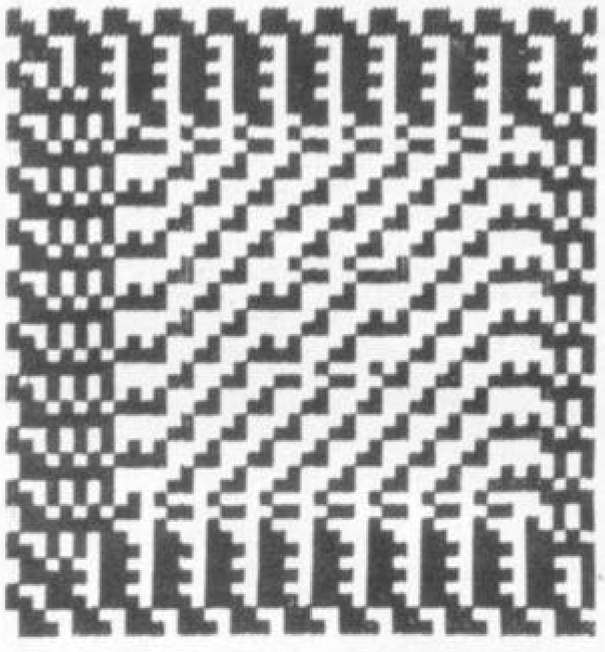
The 'style' of each frame of the pattern is established in lines 40 and 50, plus 70 and 80, which set the probability of the characters chosen being changed within each frame, and hence the 'uniformity' or 'variety' of pattern of the frame, and also the probability of the characters being aligned in or out of phase.

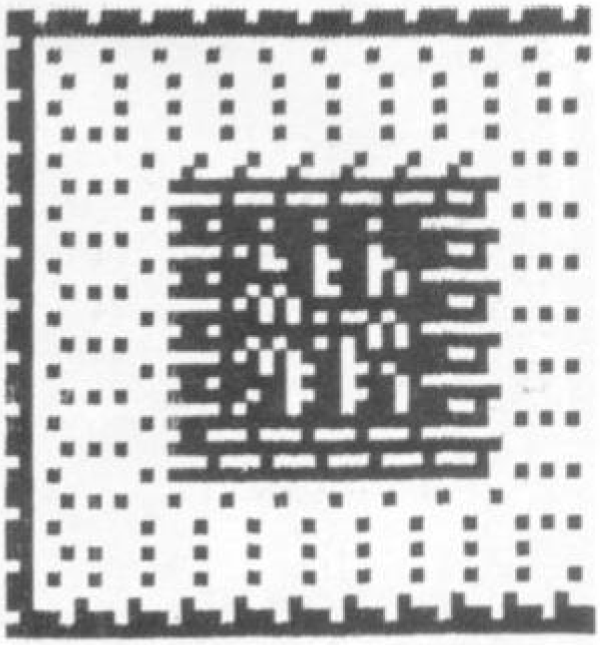




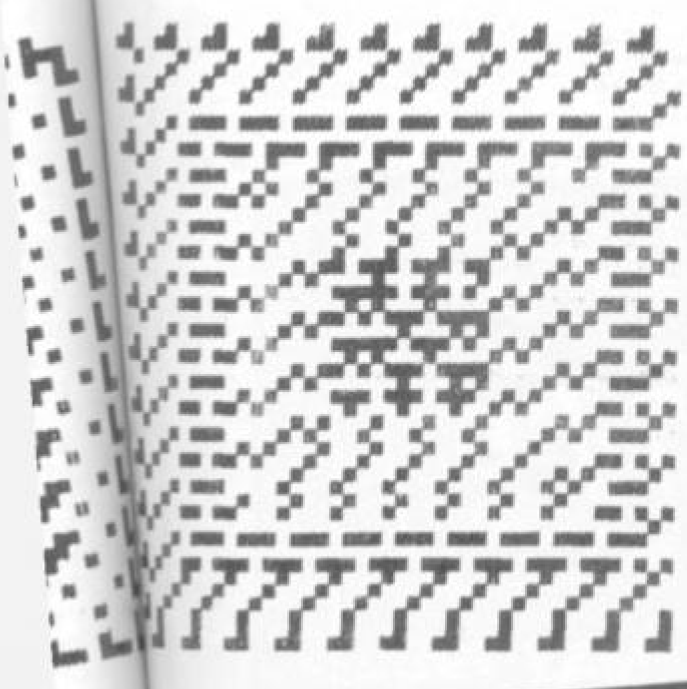


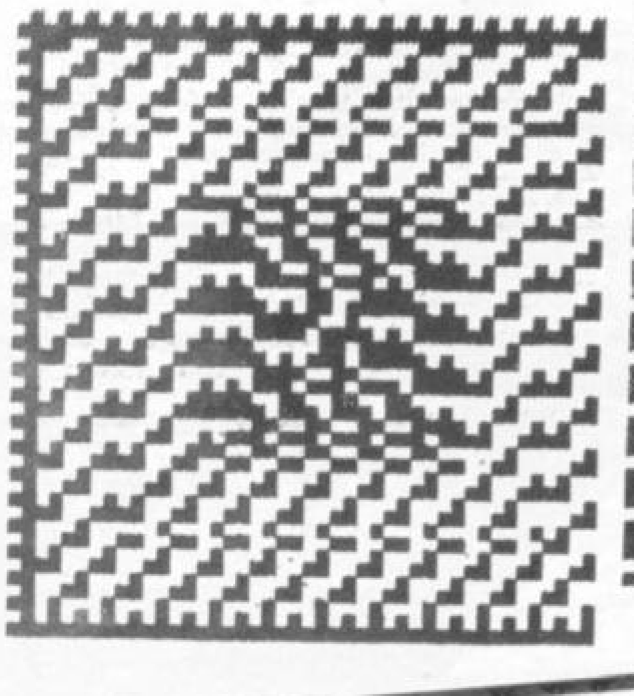


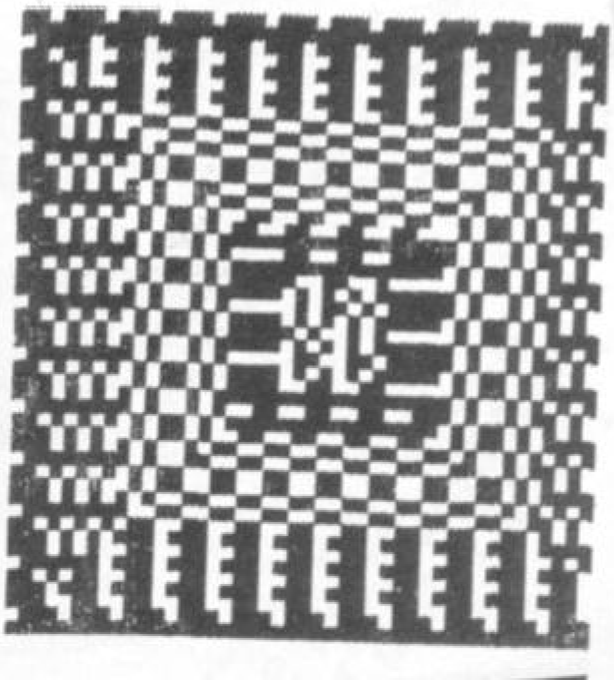




```
  1 REM HYPNO TIM JOHNS 1982
 10 LET X=PEEK 16396+256*PEEK 1
6397+4
 20 RAND
 30 GOSUB 300
 40 LET D=RND
 50 LET E=RND
 60 FOR N=0 TO 11
 70 IF RND<D THEN GOSUB 300
 80 IF RND<E THEN GOSUB 330
100 FOR I=N+1 TO 24-N STEP 2
110 POKE X+I+33*N,A
120 POKE 1+X+I+33*N,B
130 NEXT I
140 FOR I=57+32*N TO 750-34*N S
TEP 66
150 POKE X+I,A
160 POKE X+I+33,B
170 NEXT I
180 FOR I=24-N TO N+1 STEP -2
190 POKE X+I+759-33*N,A
200 POKE X+I+758-33*N,B
210 NEXT I
220 FOR I=727-32*N TO 34+34*N S
TEP -66
230 POKE X+I,A
240 POKE X+I-33,B
250 NEXT I
260 NEXT N
270 FOR I=1 TO 100
280 NEXT I
290 GOTO 30
300 LET A=INT (RND*8)+128*INT (
RND*2)
310 LET B=INT (RND*8)+128*INT (
RND*2)
320 RETURN
330 LET C=A
340 LET A=B
350 LET B=C
360 RETURN
```







NEXT MONTH

PORTABLE POWER?

After years of writing books on computers and awarding his famous 'White Elephants' to the industry, Adam Osborne finally took the ultimate step and produced the Osborne I. Despite the fact it is portable, well . . . almost, and fairly reasonably priced as hardware goes, the main attraction seems to be the fact that you get several hundred pounds' worth of software thrown in for free.

Having humped the system up hill and down dale and generally put it through its paces, our reviewer will hopefully have recovered sufficiently to put pen to paper and report on his findings.

SIMULATING FORTH

The interest sparked off by our recent series on FORTH has been so great that even we were taken by surprise. As a result, next month we'll be publishing a rather novel BASIC simulator for FORTH operations so those of you who would like to try out the language — without actually having to buy a version of it for your system — can have a go. It certainly won't break any speed records, in fact it is extremely slow, but it does allow you to get to grips with Reverse Polish Notation.

PROGRAMMERS' TWO-STEP

OK, so you've learnt to program in BASIC. Now, perhaps, you'd like to have a go at assembly but you're put off by the fact that you can't alter your programs with the same ease as you can in BASIC. Well, what you need is our two-pass assembler which lets you write and modify the assembly code first and then turn it into machine code **without** destroying your original.

.So, if you'd like a professional software tool to help you with your assembly language programming, don't miss out on our next issue!

ARE YOU SECURE?

Or, to be more precise, are your programs? In our next issue we'll be taking a look at a rather clever method of program protection which actually allows your programs to be copied but then won't let them run on any other system . . . frustrating in the extreme! We will also be publishing an extremely ingenious Voting Loader which guarantees perfect loading from cassettes at almost any speed. So, for your own peace of mind and the security of your programs and data, make sure you secure a copy of our June issue.

Articles described here are in an advanced state of preparation but circumstances may dictate changes to the final contents.

GIVE THIS TO YOUR NEWSAGENT

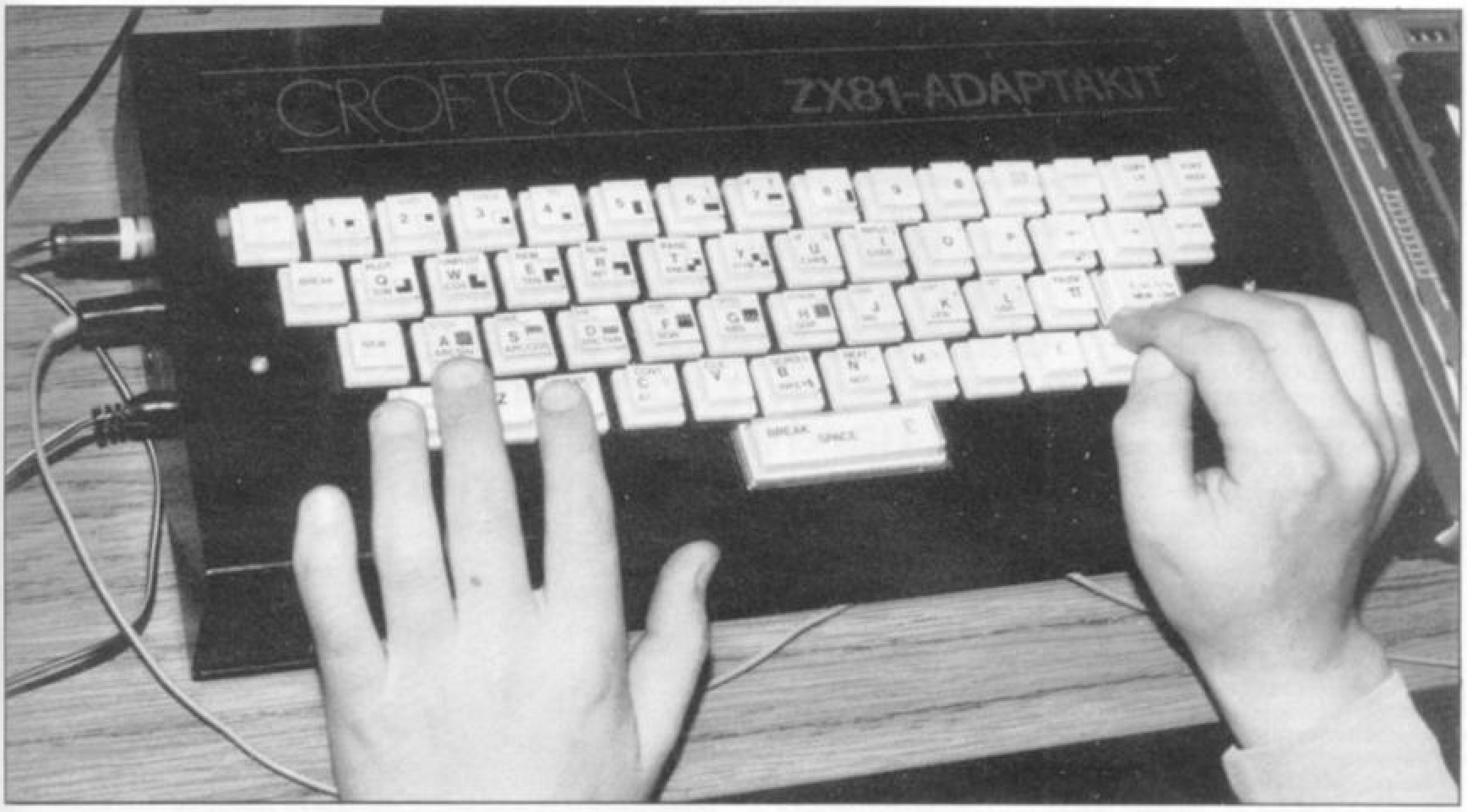
Please reserve me a regular copy of Computing Today

NAME ....

ADDRESS ....

....

JUNE ISSUE ON SALE
FRIDAY 14th MAY



Keyboard Review

The touch-sensitive keyboard is one of the main sources of criticism of the ZX81. Is an external keyboard the answer? Tim Hartnell investigates Crofton's solution to the problem.

Crofton's Adaptakit is housed in a neat, matt-black metal box, around 12 inches by six, which holds the ZX81. There are 54 keys on top which duplicate all the ZX81 keys. Additional keys are provided for many functions such as GRAPHICS, COPY, right and left cursor and EDIT, as well as two shift keys and a BREAK/SPACE bar at the bottom of the keyboard.

The keyboard is not completely satisfactory to use, because the 'feel' of the keys is poor, they tend to rattle and bounce somewhat, and the box-top is not raked, so you are working on a flat surface. Despite these minor criticisms, the keyboard makes working with the ZX81 a much more pleasant occupation than it is when working Sinclair's standard membrane keypad. Once you get used to the new positions of some of the keys, and you become familiar with the double SHIFT keys and the separate function keys, you'll find you'll be working much more quickly than in the past.

The computer is bolted into place within the case, producing a solid package which does not slide around the desk. There is a video amplifier onboard, which provides an output at a level suitable to interface with almost any standard video monitor.

A nice touch is the LED indicator to show power on. It is unfortunate that a separate, and more robust, power supply was not provided onboard, as this would be likely to be of more general value, I would suggest, than the video output. You still have to use the Sinclair power supply and leads, which plug into the ZX81 through holes in the Adaptakit's left-hand side.

The video amplifier circuitry brings the video signal level up to a standard 1 V peak-to-peak composite. This enables the output to be fed directly to a video monitor rather than via the internal modulator to a standard domestic television receiver.

The keyboard is fully assembled when you get it, but the sticky-backed key labels have not been attached, so you can follow the suggested layout, or modify it to your own requirements. The ZX81 PCB must be removed from its case, and the existing keyboard ribbon cables disconnected from their connectors. The PCB is then bolted onto the Crofton keyboard PCB. Short lengths of ribbon cable then connect the two PCBs. Once you've installed your ZX81 into the keyboard, your memory pack and printer may be plugged into the main board via a slot in the back of the keyboard case. A RAM support plate is also supplied (although one was not provided with the review model, but it appears a very useful addition) which is attached to the keyboard case by two self-tapping screws to prevent the ZX81 PCB or edge connector becoming strained. This plate should minimise memory failure.

All in all this is a nice package, and one which turns your ZX81 into a 'real computer' — or at least makes it look like one. The keyboard could be, as I suggested, better, but in its present form it is an enormous improvement over Sinclair's one. The extra keys are very useful and make game responses much quicker than on the standard keyboard. At £42.90 (including VAT and p & p) this is a worthwhile purchase if you intend to make heavy use of your ZX81 and have access to a video monitor. A kit version is available for £37.15. If you don't want the hassle of fitting the whole lot together, you can send them an extra £8.62 plus a complete cased ZX81, and they'll send you an Adaptakit with a ZX81 already wired in.

Crofton Electronics Ltd., 35 Grosvenor Road, Twickenham, Middx. (01-891 1923).

Quicksilva

Tim Hartnell interviews Nick Lambert, who started out building electronic classical organs for a living, and now runs his own company stretching ZX81 hardware beyond Sinclair's specifications.

Quicksilva — The man behind it

In August 1980, Nick Lambert wanted additional money for his ZX80. Sinclair's 3K pack was too expensive so Nick decided to build his own. This was the beginning of Quicksilva, a company which now has four full-time employees, and which has led the way in providing imaginative hardware modifications for the ZX81.

Nick was an electronics enthusiast mainly interested in electronic music, when he bought his ZX80 — the only computer he could afford.

"I realised quickly that the memory as supplied on the ZX80 was very limited, and being unwilling to endure any more Sinclair delivery delays, and because the price was high, I decided to make my own 3K pack", Nick recalled.

"It took a couple of weeks to produce the first prototype, and I then decided that as I was building one for myself — and there didn't seem to be any others on the market at the time — I should make it a real project and produce the memory packs as a business."

Nick had had some experience with PCBs, and with getting production lines moving. His first advertisement for the 3K pack appeared in the users' club magazine Interface, and Quicksilva was underway. Nick swapped a memory pack to pay for that first ad, and sold around 50 of the initial design.

"I was charging around £40 at first — Clive's 3K pack was £64 at that stage — but after Clive released his 16K for £49.95 I gradually dropped the price. It is now £15. Overall I guess I've sold around 400 3K packs."

The electronic organ business was fairly quiet, and Nick began to see the possibilities of making computer peripherals as a business.

"Although my memory pack business was OK, and it supplemented my income nicely, it was not something I could do full time at that stage", said Nick. "Nevertheless, I was enjoying it. I realised then that working for myself in this way was the only way I would be happy.

"I was doing some contract testing work on Winchester discs when the ZX81 came out. I'd always thought of myself as 'testing the water' by selling the 3K packs, and that had gone OK, so I thought I should expand my line and get a few more products together to go with the ZX81."

One of Nick's secret vices is arcade games.

"I thought a good moving graphics game would go well, and decided to do Defender, because it was my favourite. At about the same time, I came up with a sound board for the ZX81. It had been in the back of my mind for quite a while."

Nick soon discovered that between admiring a game in an arcade, and producing a version for a somewhat limited home computer lies a vast gulf, filled with blood, sweat and tears.

"Defender took three months to write. I did most of it myself. By the end I was getting pretty sick of it, and needed considerable moral support to get the thing completed. Three months it took, before I finally got it running properly."

But the trouble was worth it.

"I reckon we've sold four or five thousand", said Nick. "Most of our subsequent software products have been based on arcade games. After Defender we produced the programmable character generator board. It came out just before the South Bank Sinclair Show. It has gone well, just about anybody seems to buy it.

"Schools, in particular, have shown great interest in the character generator", said Nick. "There are many educational applications, such as drawing equations, where the character generator goes down well."

High Resolution

But while 'just about anybody' appeared to find a use for the character generator, Nick noticed a difference with the next product his company developed, a high resolution board.

"The hi-res board seems to go to a noticeably different kind of 'clientele'. These seem to be people who know more about computing, and have been waiting for someone to produce a hi-res board." Even though the high resolution board costs £85, more than the computer, Nick believes it has sold well because it and the computer can perform better than many computers costing much more than the combined price of the board and ZX81. "Compared with other machines which provide the same facility, it is about a third of the price", he said.

Quicksilva's latest project is its most exciting one — a ZX colour board.

"We'll definitely have one soon", said Nick. "We're going for eight colours. It is still in the design stage (early March) but we're well on the way to having a working prototype.

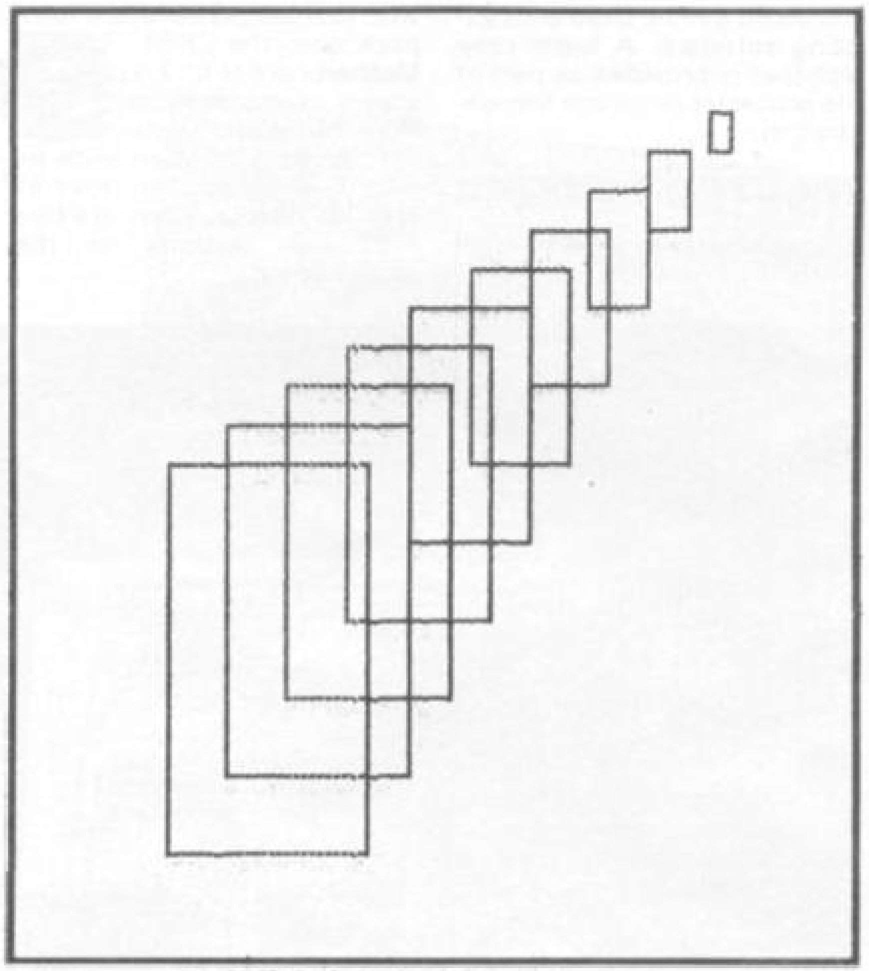
"We hope to have one working at the next Microfair. It will probably have two kinds of control. In one, you'll be able to specify a particular colour for each position on the screen. In the other, we're going to have a character code mode to specify a particular colour map. It will be controlled by simple POKEs. The character code map idea is a good one as it gives you the same facility as you get with the characters board. You can use it with existing programs."

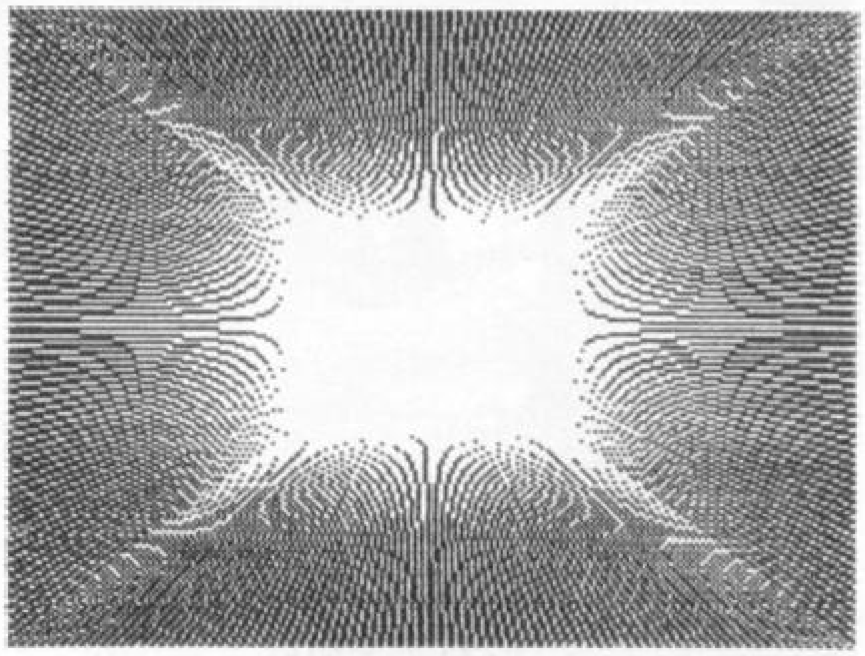
The Quicksilva idea is to allow computer users to load the colour and/or the character generator software, and then load in any game the user already has to take advantage of the programmable characters and the colour.

The company which started off with an ad swapped for its first product now is four people working full time. Nick's sidekick is John Hollis, who has done most of the design work.

"He's done everything except the characters demo software since the two of us started up with Defender and the sound board", said Nick. "As well as that we have Mark Eyles, the production manager, Chris London who is a permanent 'PCB production engineer' — or at least that's what he calls himself — along with Ellen the post-person, and Steinar Lund who does the zappy artwork for the cassette covers."

And now what's around the corner for Quicksilva? The colour board seems certain to be a winner. Quicksilva has also recently developed an interface to allow the Acorn Atom to run a ZX printer, and they're working on one for the VIC 20.





If you want to get an idea of just how good the ZX81 can be, take a good look at the products which have come from Nick Lambert's fertile mind.

Quicksilva — The products

The High Resolution Graphics Board

The hi-res board comes with its own test program in ROM which you trigger with a simple USR command. A test pattern is drawn and the message "QS HI-RES TEST . . . ALL OK" appears. The hi-res board's display is on a 256 by 192 matrix. Any particular point may be white or black, and can be referenced by its X and Y coordinates.

The effects generated by the board are fantastic. We had some samples running in the ZX Computing offices when Nick brought the stuff in to show us, and my office became very crowded as more and more people came in to see the marvellous board do its tricks.

The board is simple to use. You call it with a USR command, and the board follows instructions stored in English in a REM statement. For example, REM BLACK CLEAR MOVE A A DRAW B B, when A is zero and B is 100, will clear to a white area, and a line will be drawn in black from the top left hand corner (0,0) to somewhere near the middle (100,100).

The graphics commands, and what they do, are:

BLACK — sets the pen colour to black.

WHITE — sets the pen colour to white.

CLEAR — clears the whole hi-res display to the opposite of the pen colour.

MOVE X Y — This moves the current co-ordinates (the hi-res cursor) to X Y but does not draw anything.

PLOT X Y — This plots a single point in the current pen colour at the X Y co-ordinate specified.

DRAW X Y — This command draws a line in the current pen colour from the current co-ordinate to the new co-ordinate specified by X Y.

BOX X Y — This command, which is found on few micros, draws a rectangle from the current co-ordinate to the new co-ordinate (X Y) which sets the position of the opposite corner of the box.

UP, DOWN, LEFT, RIGHT — These four commands move the whole hi-res screen by one pixel in the direction specified. Anything moved off the screen will be lost.

SCROLL — Scrolls the lower 24 lines (3 x 8) by eight lines upwards, one at a time, and sets the current print position to the start of the bottom line. This provides a three-character line text window at the bottom of the display for prompts and messages.

COPY — This command, believe it or not, copies the whole hi-res screen to the ZX printer.

To make this simple system even simpler to use, all of the graphics commands can be abbreviated to the first two letters so MOVE X Y DRAW A B SCROLL comes out as MO X Y DR A B SC.

The Software

QS Defender occupies around 3K of machine code, and uses a 34 line display file which extends from the top to the bottom of the television screen. Up to 84 fast-moving characters are on screen at a time. Although this is not the best-looking Defender for the ZX81 on the market, it plays remarkably well, and is greatly enhanced when the sound board is connected.

QS Asteroids, which needs a ZX81 or New ROM ZX80 with the SLOW modification, is a pretty good version of the arcade game, despite the odd-looking asteroids. Quicksilva had 'intelligent' asteroids which homed in on your ship in an early version but discovered that such a game was impossible to defeat, so the program you'll get now is a faithful copy of the arcade game.

QS Invaders is the biggest of Quicksilva's games, with a full 7K of machine code (plus a little BASIC). There are seven rows of 13 invaders. To see this program running with user-programmed invaders on the character board, with sound, is to imagine you're watching a proper arcade machine in action, despite the lack of colour. A number of manufacturers have produced invader programs. This one, without the sound and user-programmed characters, is pretty much par for the course.

The Sound Board

The sound chip used in the board has 16 internal registers, each of which control a different section of the sound generation. Registers 0 and 1 are for controlling the tone and pitch of sound channel A, registers 2 and 3 control channel B and 4 and 5 the channel C tone source. Register 6 tunes the noise generator and register 7 is used to switch the various sound sources, controls the switching of the tone generators, the switching of the two ports to input or output. Registers 8, 9 and 10 control the volume of each channel, 11 and 12 set the length of one envelope cycle and register 13 controls the envelope shape and pattern. A number of sample routines have been provided by Quicksilva showing how flexible and useful the sound board can be. Music, explosions, trains and phasers can all enliven programs.

Character Generator

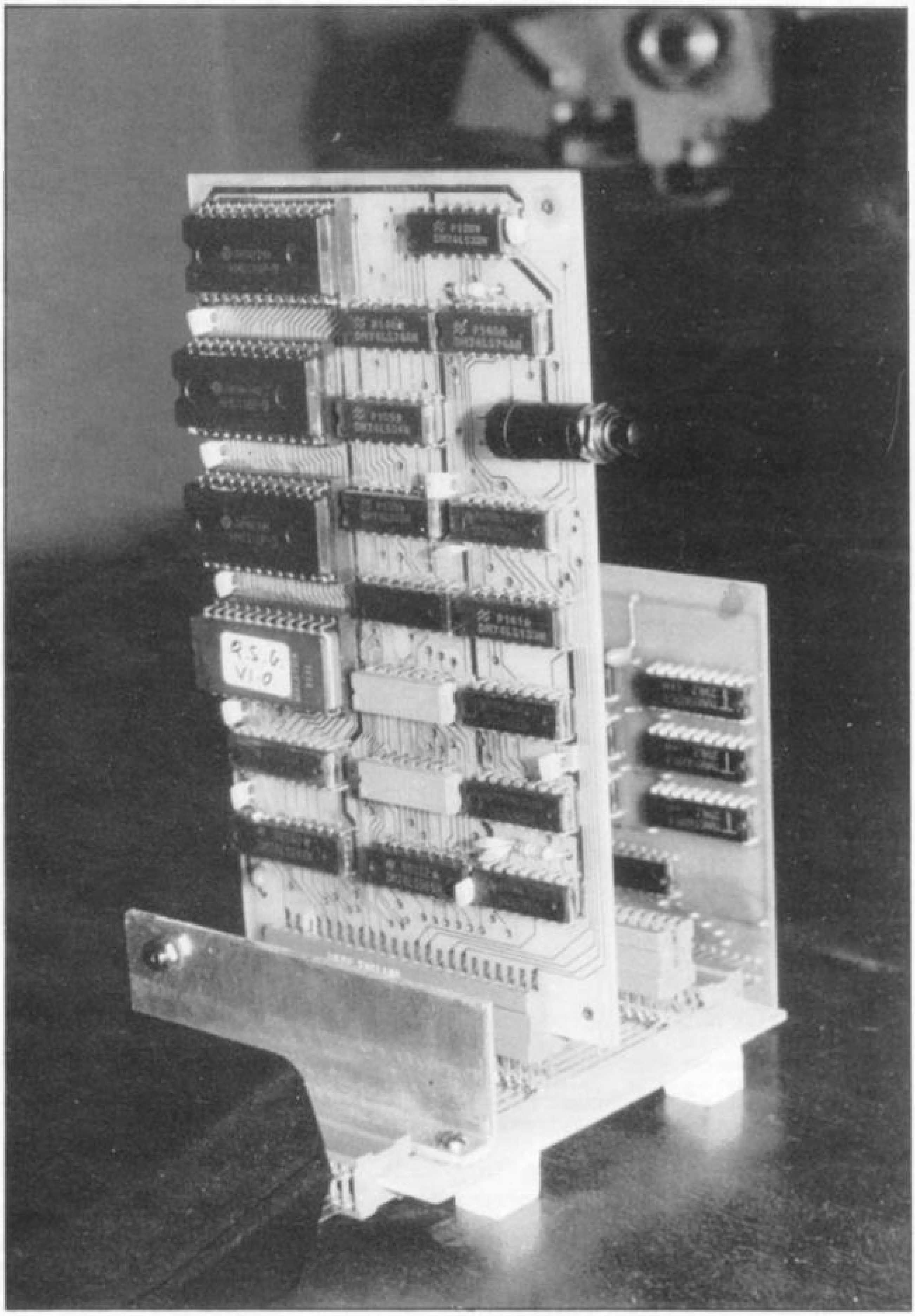
This is a single circuit board which plugs directly into the QS Motherboard, or via the QS connector directly into the computer. There are 128 fully programmable characters, and the board can be used with existing software. A lower-case alphabet is provided as part of the character generator demonstration.

The QS Motherboard

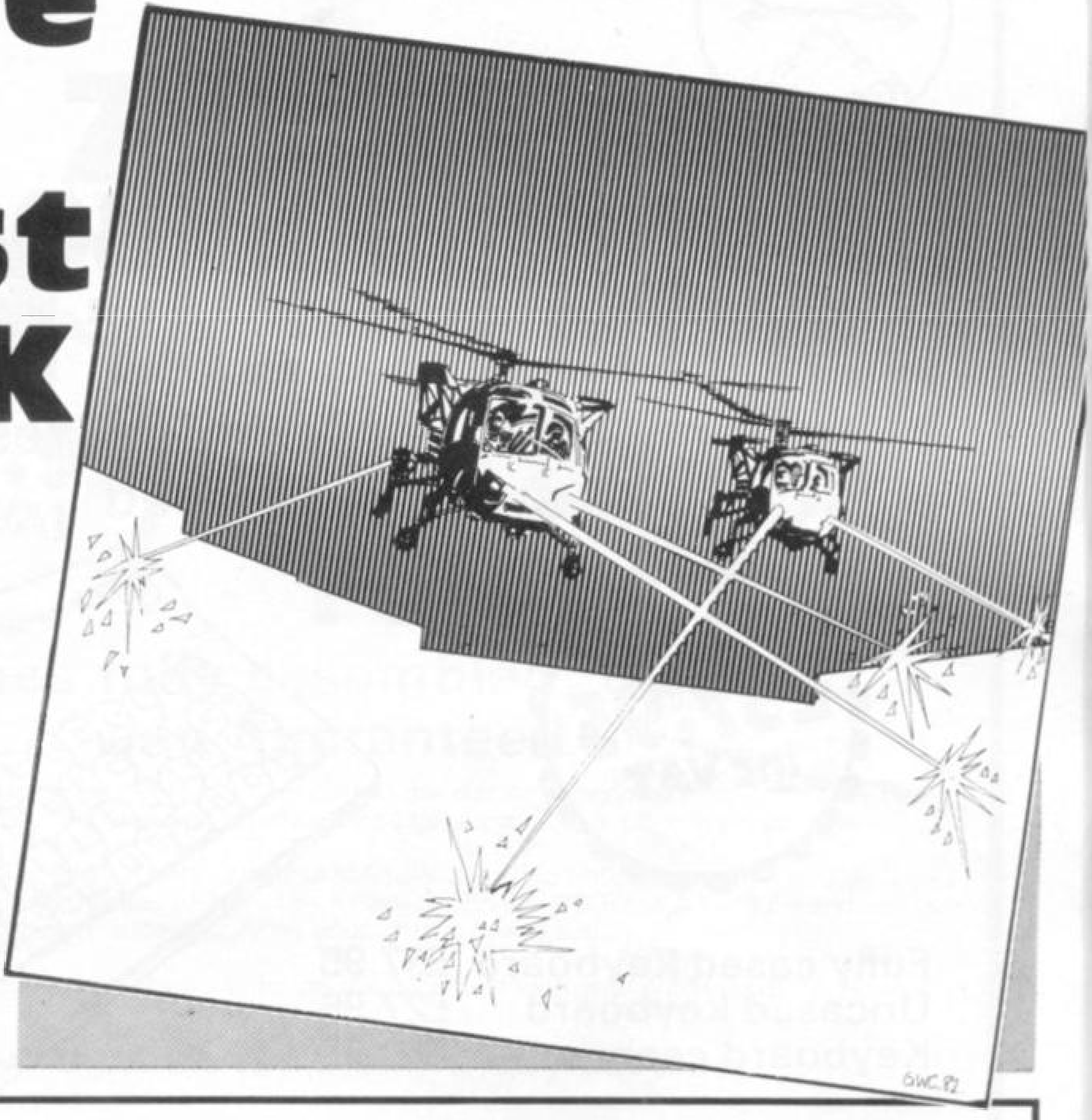
Some kind of motherboard is required if you want to hook more than just the printer and a RAM pack onto the ZX81. The QS Motherboard at £12 represents a very cheap and useful addition to your system. The motherboard is fitted with its own 5 V regulator to drive all external boards. There are two expansion sockets on the motherboard.

Quicksilva are at 95 Upper Brownhill Road, Maybush, Southampton, Hants.

Part of the impressive range of hardware support provided by Nick Lambert's company, Quicksilva.



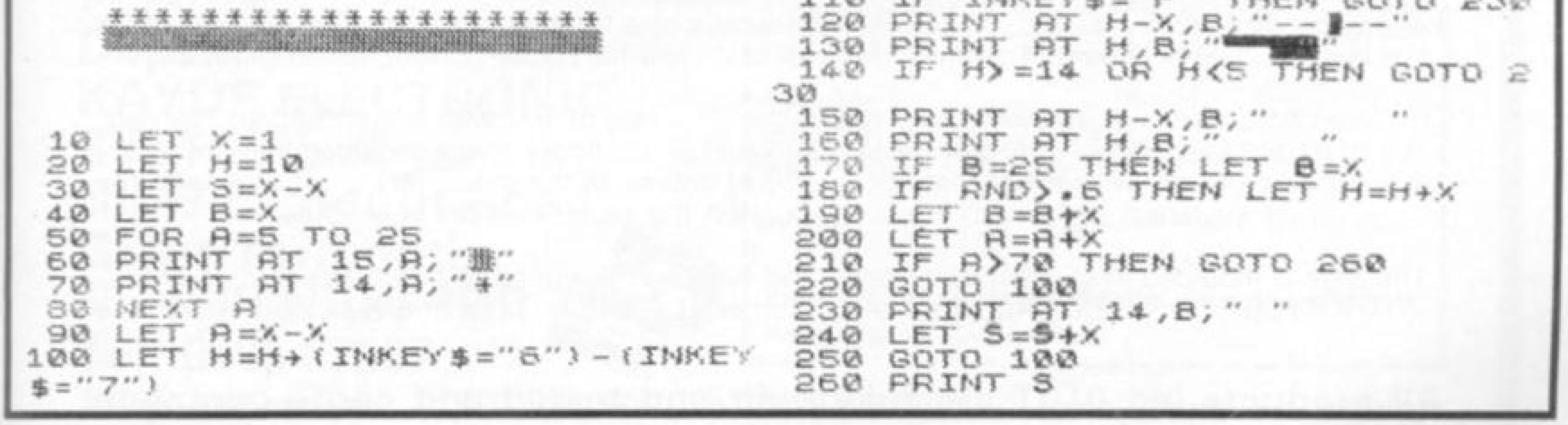
Make the most of 1K



Chopper Squad

R Morland and M Frobisher of Chalfont St Giles sent us this great little program for a 1K ZX81. You are in charge of a cleverly-drawn helicopter. The "6" and "7" keys move your chopper up and down, while the "F" key fires.

Your helicopter flies in from the left of the screen, buffeted by random crosswinds, and air pockets. You use the "6" and "7" to keep your craft in flight, and from time to time, press "F" to obliterate an asterisk or two underneath you. Due to lack of space, the program is not perfect, as the helicopter from time to time leaves an 'image' of itself behind if it lands on the base, then takes off again. You may well want to work on this problem. The variable S is your score.



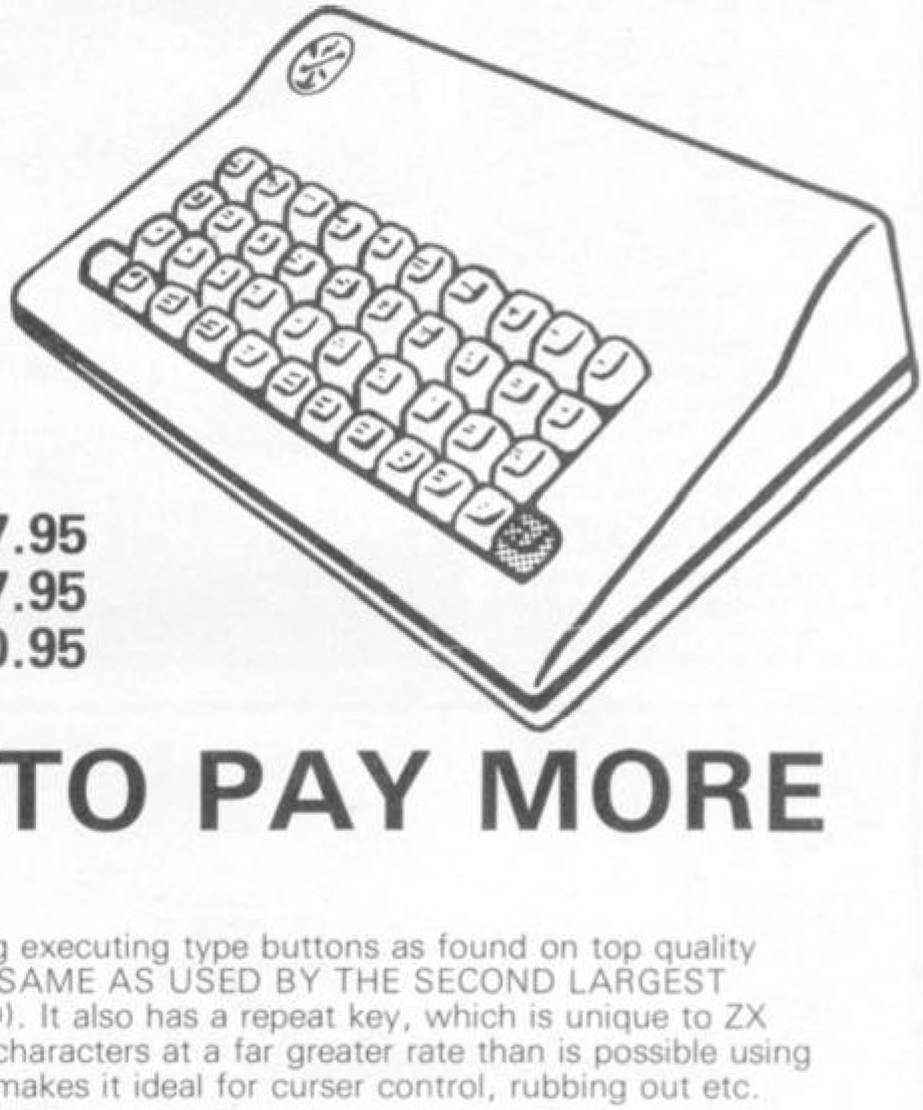
```
     *********************
     ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒


 10 LET X=1
 20 LET H=10
 30 LET S=X-X
 40 LET B=X
 50 FOR A=5 TO 25
 60 PRINT AT 15,A;"▒"
 70 PRINT AT 14,A;"*"
 80 NEXT A
 90 LET A=X-X
100 LET H=H+(INKEY$="6")-(INKEY
$="7")
110 IF INKEY$="F" THEN GOTO 230
120 PRINT AT H-X,B;"--▌--"
130 PRINT AT H,B;"▀▀▀█"
140 IF H>=14 OR H<5 THEN GOTO 2
30
150 PRINT AT H-X,B;"     "
160 PRINT AT H,B;"    "
170 IF B=25 THEN LET B=X
180 IF RND>.6 THEN LET H=H+X
190 LET B=B+X
200 LET A=A+X
210 IF A>70 THEN GOTO 260
220 GOTO 100
230 PRINT AT 14,B;" "
240 LET S=S+X
250 GOTO 100
260 PRINT S
```

KAYDE Elec

ZX 80/I
FULL SIZE KEYBOARDS
WITH REPEAT KEY

comes fully assembled, tested
and guaranteed

£37.95
inc VAT



Fully cased keyboard	£37.95
Uncased keyboard	£27.95
Keyboard case	£10.95

WHY WAIT TO PAY MORE

Fast delivery:

Our keyboard is highly professional using executing type buttons as found on top quality computers (IN FACT THEY ARE THE SAME AS USED BY THE SECOND LARGEST COMPUTER COMPANY IN THE WORLD). It also has a repeat key, which is unique to ZX keyboards. This facility allows you to enter characters at a far greater rate than is possible using a conventional keyboard. Therefore this makes it ideal for curser control, rubbing out etc.

It comes complete with its own luxury case, (but for those of you with a tight budget this can be purchased separately at a later date). The keytops are finger shape moulded and are in a pleasant tone of grey with the repeat key in red, of course all the characters associated with the ZX keyboard are printed in two colours on the keytops as per your ZX80/1.

The case is moulded in sturdy black plastic and has been designed to complement your ZX81. The KAYDE PROFESSIONAL KEYBOARD comes at a realistic price and is second to none.

All products inc. VAT are fully built and tested and come complete with money back guarantee.

tronic Systems

ZX 80/1 16K RAMPACK

£32.95 inc VAT

MASSIVE ADD ON MEMORY FULLY CASED

comes fully assembled, tested and guaranteed

The 16K Rampack plugs straight into the user port at the rear of your computer. And of course as in all our products the Rampack is fully enclosed in a matt black slimline plastic case.

This means you can handle it with complete safety and you do not have to stare at ugly components – as in other makes of 16K. The Rampack NEEDS NO EXTRA POWER and will operate quite happily on your sinclair power supply.

Don't forget you can always order on the telephone with your credit card.

6 days a week for a same day service.

WHY WAIT TO PAY MORE

Fast delivery:

Post to:-
Dept ZXC1
KAYDE ELECTRONIC
SYSTEMS
48/49 Exmouth Road,
Great Yarmouth,
Norfolk NR30 3DP
Tel: 0493 57867

PLEASE SEND ME.....	RAM PACK/S	£32.96 each
PLEASE SEND ME.....	CASED KEYBOARD/S	£37.96 each
PLEASE SEND ME.....	UNCASED KEYBOARD/S	£27.96 each
PLEASE SEND ME.....	KEYBOARD CASE/S	£10.96 each

I enclose £........

NAME ....

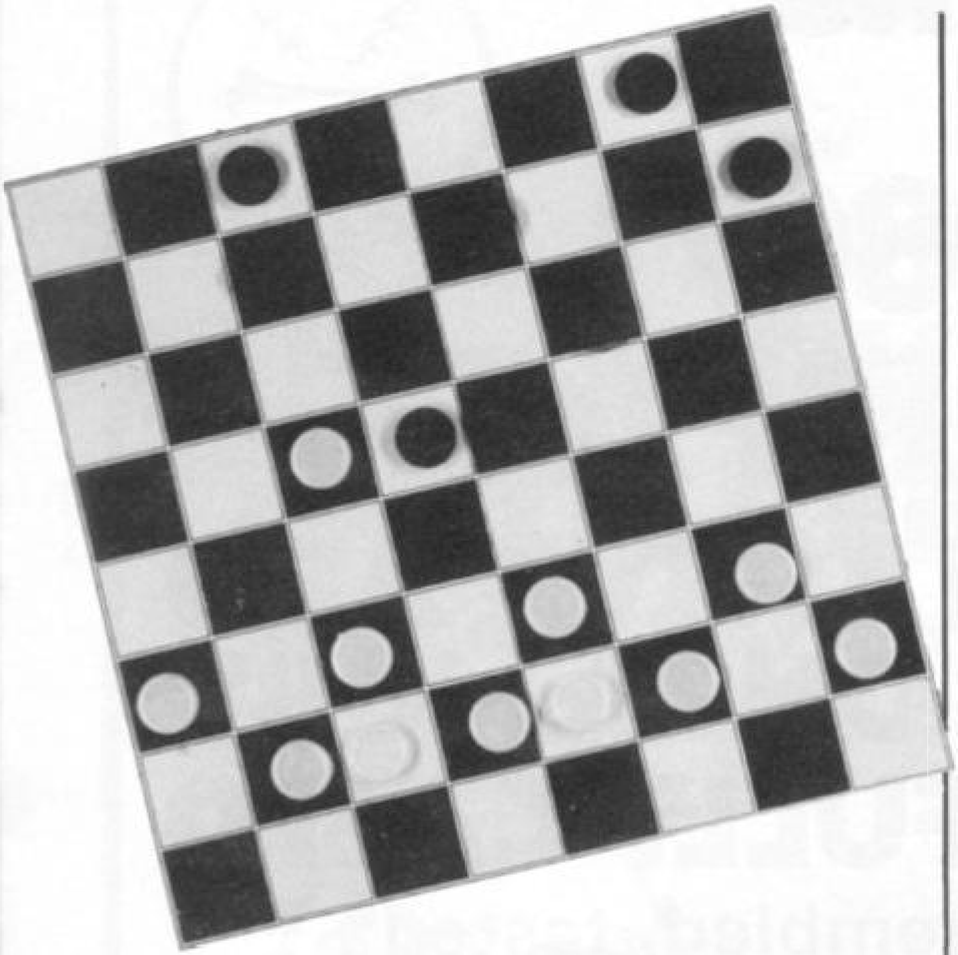
ADDRESS ....

....

....

Please add £1.50 P&P and make cheques payable to Kayde Electronic Systems.

All products inc. VAT are fully built and tested and come complete with money back guarantee.



DRAUGHTS

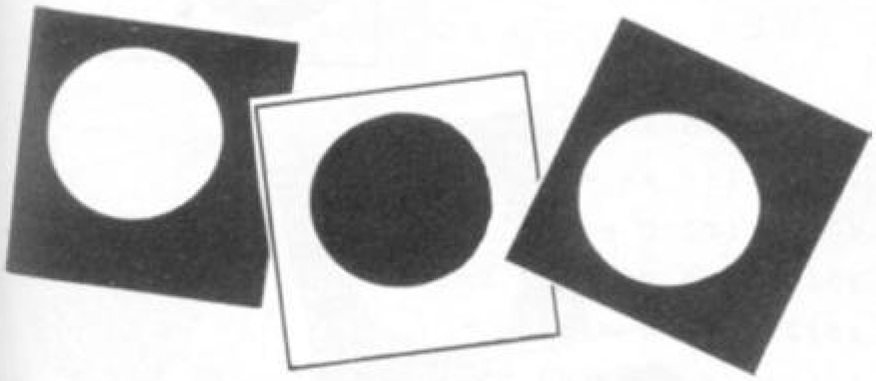
This is a full game of draughts for the 4K ZX80. A complete board is shown. You are at the bottom of the screen playing up, and the ZX80 is playing down. You enter each move as a LETTER NUMBER, then NEWLINE, of the square you're moving from, then LETTER NUMBER, then NEWLINE, of the square you're moving to. You'll understand this easily once you see the display. You'll find the ZX80 plays like a reasonable beginner. Once you've played with it for a while, you might enjoy trying to improve its standard of play.

Although the computer has no strategy as such (other than look for a capture, and if it can't make one, look for a legal move) you'll find it plays as though it had a real method of looking ahead at the game and where it is developing. Kings are made automatically.

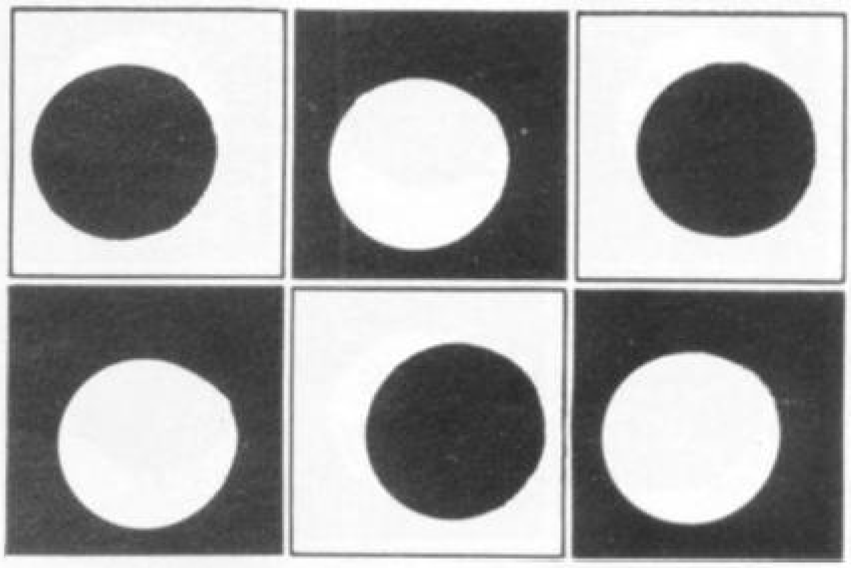
```
1     GOTO 9000
2     FOR Z = 6 TO 40
3     IF Z > 37 AND A(Z) = 1 THEN LET
      A(Z) = 2
4     IF Z < 10 AND A(Z) = - 1 THEN
      LET A(Z) = - 2
5     IF P >  Ø THEN LET SI = SI + 1
6     LET P = Ø
21    IF A(Z) = - 2 THEN LET A(Z) = 188
22    IF A(Z) = -1 THEN LET A(Z) = 180
23    IF A(Z) = Ø THEN LET A(Z) = 128
24    IF A(Z) = 1 THEN LET A(Z) = 189
25    IF A(Z) = 2 THEN LET A(Z) = 186
26    NEXT Z
39    PRINT ,"        ABCDEFGH"
40    PRINT "ZX80 ";SI;"eight shift A"
(in the next section, @ equals shift A)
41    PRINT ," 1@ ";CHR$(A(40));" ";CHR$
      (A(39));" ";CHR$(A(38));" ";CHR$
      (A(37));"@"
42    PRINT ," 2@";CHR$(A(36));" ";CHR$
      (A(35));" ";CHR$(A(34));" ";CHR$
      (A(33))" @"
43    PRINT ," 3@ ";CHR$(A(31));" ";CHR$
      (A(30));" ";CHR$(A(29));
              " ";CHR$(A(28));"@"


44    PRINT ," 4@";CHR$(A(27));" ";CHR$
      (A(26));" ";CHR$(A(25));" ";CHR$
      (A(24));" @"
45    PRINT ," 5@ ";CHR$(A(22));" "CHR$
      (A(21));" ";CHR$(A(20));" ";CHR$
      (A(19));"@"
46    PRINT ," 6@";CHR$(A(18));" ";CHR$
      (A(17));" ";CHR$(A(16));" ";CHR$
      (A(15));" @"
47    PRINT ," 7@ ";CHR$(A(13));" ";CHR$
      (A(12));" ";CHR$(A(11));" ";CHR$
      (A(10));"@"
48    PRINT ,"  8@";CHR$(A(9));" ";CHR$
      (A(8));" ";CHR$(A(7));" ";CHR$(A(6));
      " @"
49    PRINT "HUMAN  ";SI;"  @@@@@@@@"
50    FOR Z = 6 TO 40
```

```
51    IF A(Z) = 188 THEN LET A(Z) = - 2
52    IF A(Z) = 180 THEN LET A(Z) = - 1
53    IF A(Z) = 128 THEN LET A(Z) = Ø
54    IF A(Z) = 189 THEN LET A(Z) = 1
55    IF A(Z) = 186 THEN LET A(Z) = 2
56    NEXT Z
```

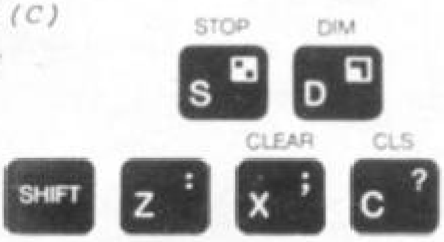
```
58    IF SI = 12 THEN PRINT "I WIN"
59    IF SM = 12 THEN PRINT "YOU WIN"
60    IF SM = 12 OR SI = 12 THEN STOP
```

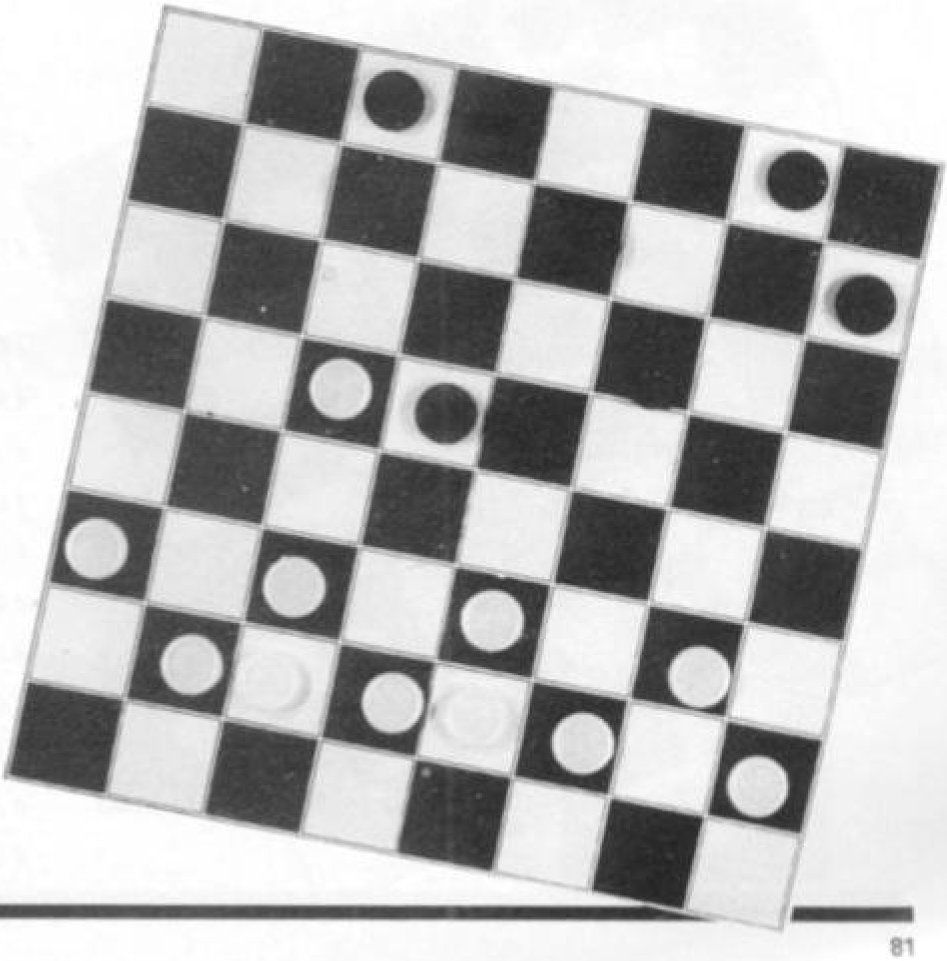


```
68    PRINT "LAST TO ";F$;" FROM? (LET.
       NUM.  ";
69    INPUT C$
70    PRINT C$;" TO?  ";
71    INPUT B$
74    LET F$ = B$
79    FOR W = 1 TO 2
77    IF W = 1 THEN LET D$ = C$
78    IF W = 2 THEN LET D$ = B$

80    LET D = - 40*(D$ = "B1") - 39* (D$
      = "D1") - 38*(D$ = "F1") - 37*(D$
      = "H1") - 36*(D$ = "A2") - 35*(D$
      = "C2") - 34*(D$ = "E2") - 33*(D$
      = "G2") - 31*(D$ = "B3") - 30*(D$
      = "D3") - 29*(D$ = "F3") - 28*(D$
      = "H3") - 27*(D$ = "A4") - 26*(D$
      = "C4") - 25*(D$ = "E4") - 24*(D$
      = "G4") - 22*(D$ = "B5") - 21*(D$
      = "D5") - 20*(D$ = "F5") - 19*(D$
      = "H5") - 18*(D$ = "A6") - 17*(D$
      = "C6") - 16*(D$ = "E6") - 15*(D$
      = "G6") - 13*(D$ = "B7") - 12*(D$
      = "D7") - 11*(D$ = "F7") - 10*(D$
      = "H7") - 9*(D$ = "A8")  -  8*(D$
      = "C8") - 7*(D$ = "E8")  -  6*(D$
      = "G8")
```

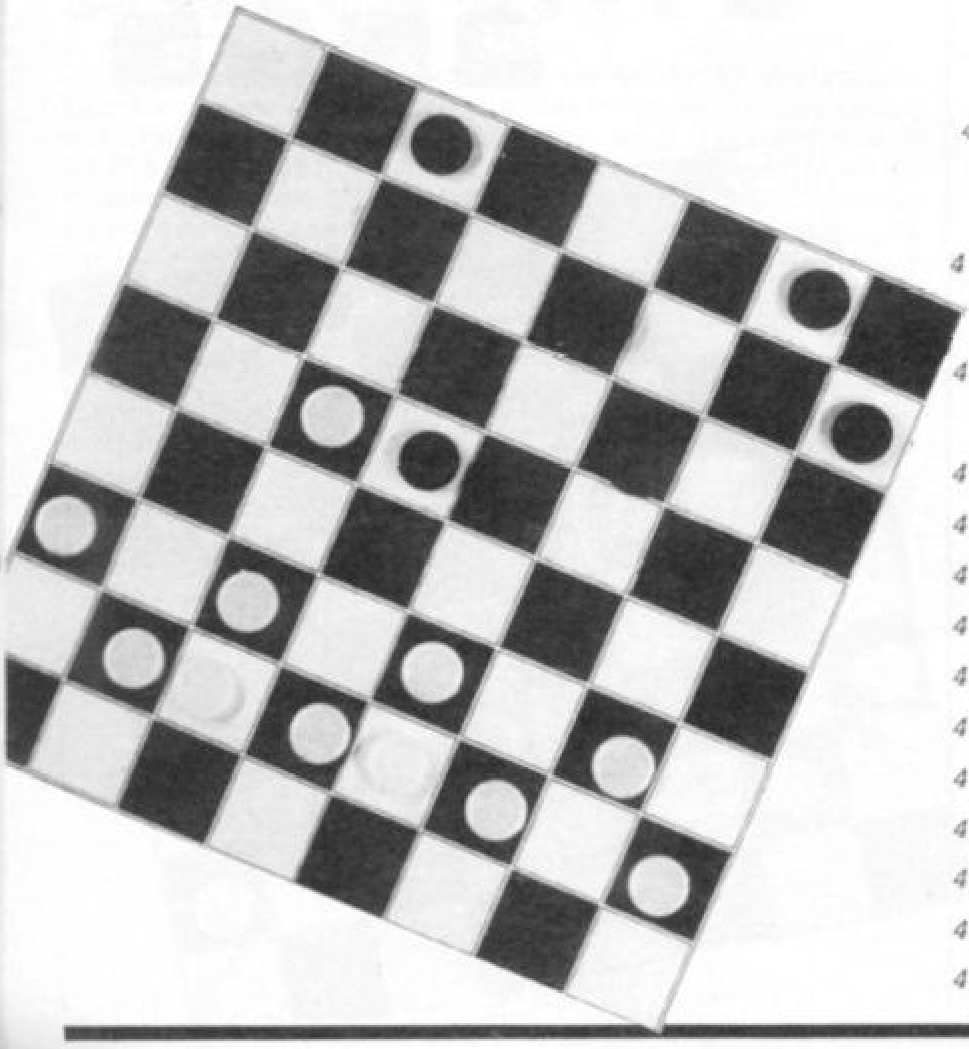


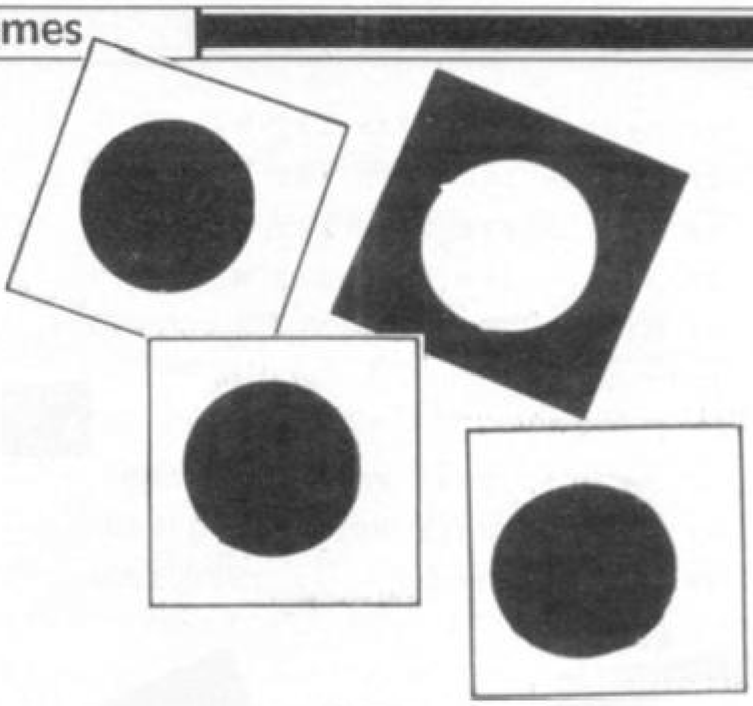
```
90    IF W = 1 THEN LET C = D
95    IF W = 2 THEN LET B = D
100   NEXT W

103   IF ABS(C - B) = 10 OR ABS(C - B) =
      8 THEN LET SM = SM + 1
142   IF B - C = 10 THEN LET A(B - 5) = Ø
146   IF B - C = 8 THEN LET A(B - 4) = Ø
147   IF C - B = 10 THEN LET A(C - 5) = Ø
148   IF C - B = 8 THEN LET A(C - 4) = Ø
150   LET A(B) = A(C)
160   LET A(C) = Ø
165   LET U$ = ""
```





```
170  IF ABS(C - B) = 10 OR ABS(C -B)
     = 8 THEN PRINT "MULTI-JUMP?"
180  IF ABS(C - B) = 10 OR ABS(C - B)=
     8 THEN INPUT U$
1990 CLS
1995 IF U$ > "" THEN GOTO 2
2000 FOR Z = 6 TO 40
2005 IF Z < 9 THEN GOTO 2015
2010 IF A(Z) < 0 AND (A(Z - 4) = 1
     OR A(Z - 4) = 2) AND A(Z - 8)
     = 0 THEN GOTO 4090
2012 IF Z < 11 THEN GOTO 2015

2014 IF A(Z) < 0 AND (A(Z - 5) = 1
     OR A(Z - 5) = 2) AND A(Z - 10)
     = 0 THEN GOTO 4200
2015 IF Z > 25 THEN GOTO 2020

2016 IF A(Z) = - 2 OR (A(Z + 4) = 1
     OR A(Z + 4) = 2) AND A(Z + 8) =
     0 THEN GOTO 4300
2017 IF A(Z) = - 2 AND (A(Z + 5) = 1
     OR A(Z + 5) = 2) AND A(Z + 10)
     = 0 THEN GOTO 4400
2020 NEXT Z
```

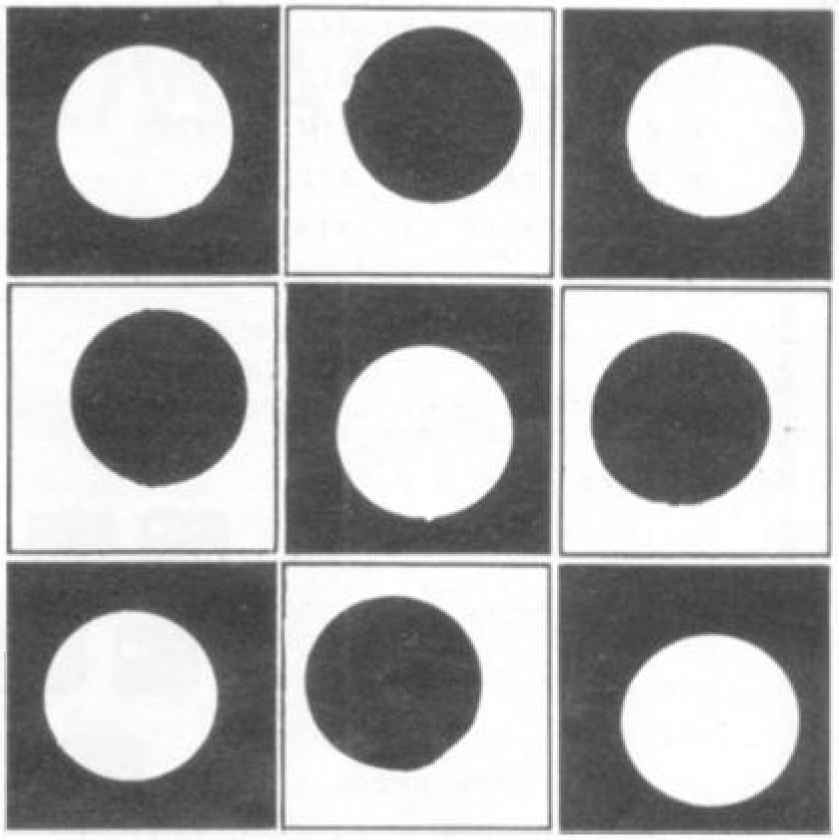


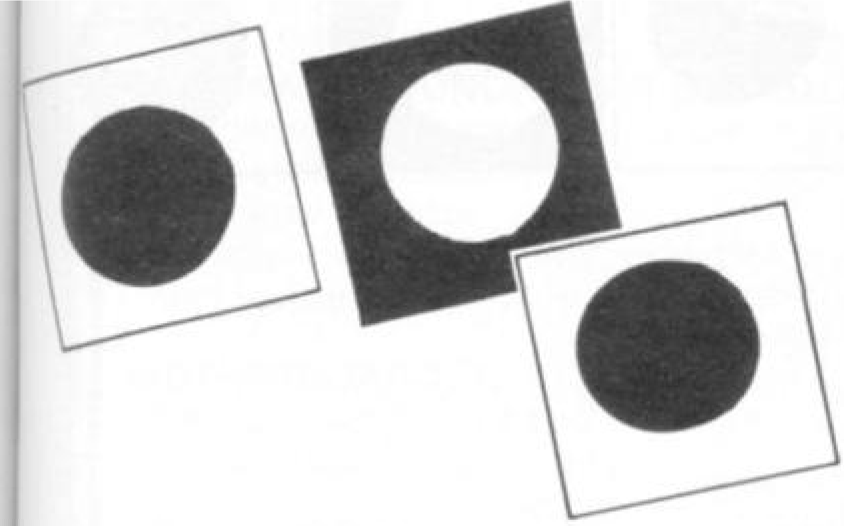


RANDOM MOVE INTELLIGENT

```
4005 LET KK = 0
4007 LET U = 0
4010 LET Z = 6 + RND(35)
4012 LET K = 0
4015 IF U > 199 THEN LET KK = KK + 1
4016 LET U = U + 1
4020 IF A(Z) < 0 AND A(Z - 4) = 0
     THEN LET K = 1
4026 IF A(Z) < 0 AND A(Z - 5) = 0
     AND K = 0 THEN LET K = 2
4029 IF K = 0 AND Z < 26 AND A(Z) =
     - 2 AND A(Z + 4) = 0 THEN
     LET K = - 7
4035 IF Z < 10 THEN GOTO 4040


4038 IF K = 1 AND U < 200 AND A(Z
     - 10) = 1 THEN GOTO 4010
4039 IF K = 2 AND U < 200 AND A(Z - 10)
     = 1 THEN GOTO 4010
4045 IF K = 0 AND KK < 350 THEN GOTO 4010
4050 IF K = 0 THEN LET SM = 12
4051 IF K = 0 THEN GOTO 2
4072 LET A(Z - (3 + K))= A(Z)
4073 LET A(Z) = 0
4077 GOTO 2
4092 LET A(Z - 8) = A(Z)
4095 LET A(Z) = 0
4100 LET A(Z - 4) = 0
4105 LET SI = SI + 1
4107 IF Z < 24 THEN GOTO 2
```

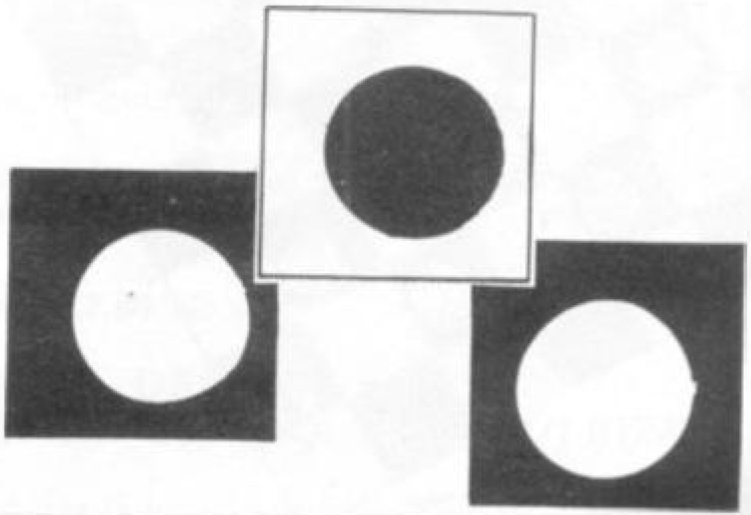
```
4110   IF (A(Z - 13) = 1 OR A(Z - 13)=
       2 ) AND A(Z - 18) = 0 THEN LET
       P = 1
4115   IF (A(Z - 12) = 1 OR A(Z - 12) =
       2) AND A(Z - 18) = 0 THEN LET P
       = 2
4120   IF P = 1 THEN LET A(Z - 18) =
       A(Z - 8)
4130   IF P = 1 THEN LET A(Z - 13) = 0
4140   IF P = 2 THEN LET A(Z - 16) =
       A(Z - 8)
4150   IF P = 2 THEN LET A(Z - 12) = 0
```



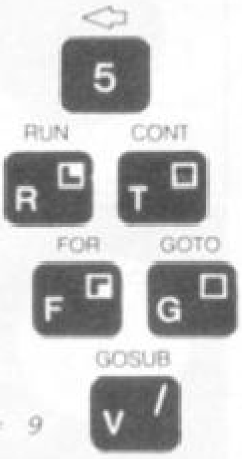


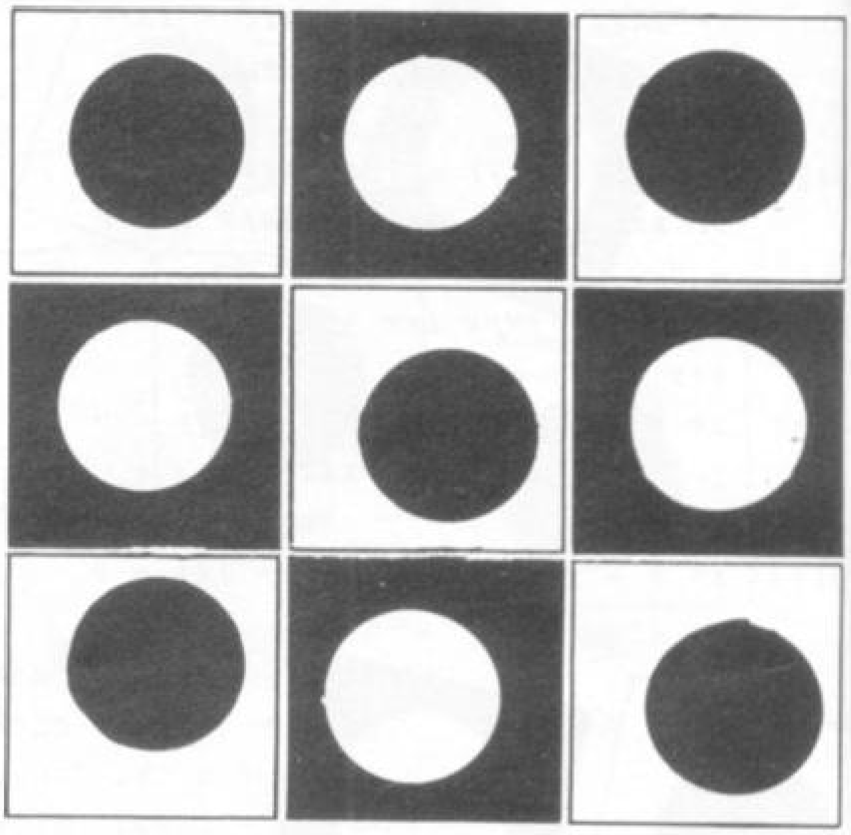
```
4160   IF P > 0 THEN LET A(Z - 8) = 0
4170   GOTO 2
4200   LET A(Z - 10) = A(Z)
4205   LET A(Z) = 0
4210   LET A(Z - 5) = 0
4212   LET SI = SI + 1
4215   IF Z < 25 THEN GOTO 2
4220   IF (A(Z - 15) = 1 OR A(Z - 15)
       = 2) AND A(Z - 20) = 0 THEN LET
       P = 1
4230   IF(A(Z - 14) = 1 OR A(Z - 14)
       = 2) AND A(Z - 18) = 0 THEN
       LET P = 2
4240   IF P = 1 THEN LET A(Z - 15) = 0

4250   IF P = 1 THEN LET A(Z - 20)
       = A(Z - 10)
4260   IF P = 2 THEN LET A(Z - 14) = 0
4270   IF P = 2 THEN LET A(Z - 18)
       = A(Z - 10)
4280   IF P > 0 THEN LET A(Z - 10) = 0
```

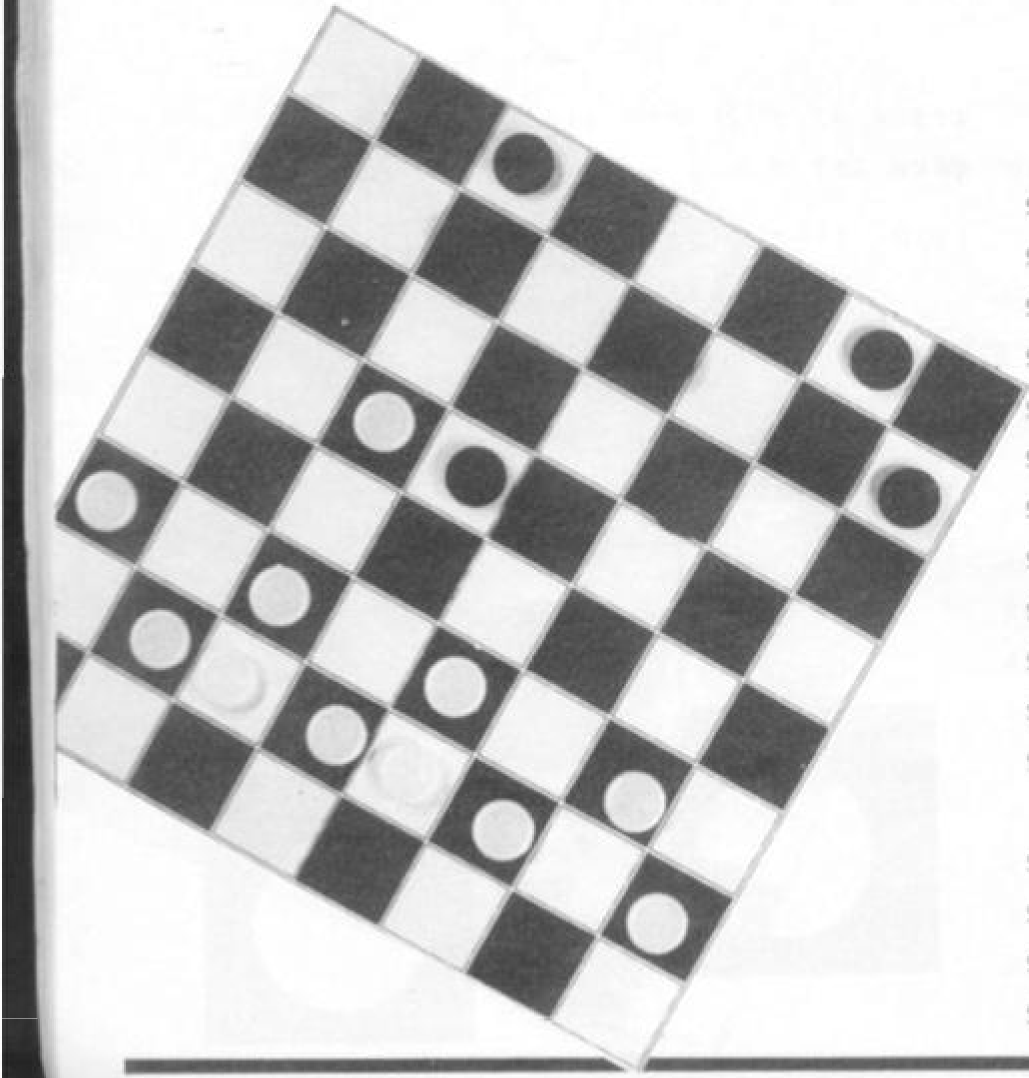
```
4290   GOTO 2
4300   LET A(Z + 8) = - 2
4310   LET A(Z + 4) = 0
4320   LET A(Z) = 0
4330   LET SI =  SI + 1
4340   IF Z < 32 AND (A(Z + 3) = 1 OR


A(Z + 3) = 2) AND A(Z - 2) = 0
THEN LET P = 1

4350   IF Z < 23 AND (A(Z + 14) = 1 OR
       A(Z + 14) = 2) AND A(Z + 16) =
       0 THEN LET P = 2
4360   IF Z < 23 AND (A(Z + 13) = 1 OR
       A(Z + 13) = 2) AND A(Z + 18) =
       0 THEN LET P = 3
```



```
4365  IF P = 1 THEN LET A(Z + 3) = 0
4367  IF P = 1 THEN LET A(Z - 2) = - 2
4370  IF P = 2 THEN LET A(Z + 14) = 0
4375  IF P = 2 THEN LET A(Z + 16) = - 2
4380  IF P = 3 THEN LET A(Z + 13) = 0
4385  IF P = 3 THEN LET A(Z + 18) = - 2
4390  IF P > 0 THEN LET A(Z + 8) = 0
4395  GOTO 2
4405  LET A(Z + 10) = - 2
4410  LET A(Z + 5) = 0
4420  LET A(Z) = 0
4425  LET SI = SI + 1
4430  GOTO 2
9000  DIM A(45)
9005  FOR Z = 1 TO 45
9010  IF Z < 6 THEN LET A(Z) = 9
9020  IF Z > 5 AND Z < 19 THEN LET
      A(Z) = 1

9030  IF Z > 18 AND Z < 28 THEN
      LET A(Z) = 0
9040  IF Z > 27 AND Z < 41 THEN LET
      A(Z) = - 1
```







```
9050  IF Z > 40 THEN LET A(Z) = 9
9060  NEXT Z
9070  LET A(14) = 9
9080  LET A(23) = 9
9090  LET A(32) = 9
9095  LET U$ = ""
9100  LET P = 0
9110  LET F$ = "- - "
9120  RANDOMISE
9130  LET SI = 0
9140  LET SM = 0
9150  PRINT "DO YOU WANT FIRST MOVE?
      (Y OR N)"
9160  INPUT J$
9170  CLS
9180  IF NOT J$ = "Y" THEN GOTO 2000
9190  GOTO 2
```

ZX81 HARDWARE

40 KEY KEYBOARD. Kit £20.50 Built £25.75 (RE77)
- Proper typewriter style keys.
- All legends and graphics in two colours.
- No soldering required to ZX81. Plugs in. (RAM/Printer not affected)
- Complete with all parts, connectors, feet and comprehensive instructions.

24 LINE IN/0UT PORT. Kit £16.95 Built £18.95 (RE98)
- Each line either in or out.
- Controlled by BASIC.
- Allows printer/RAM to be used without a motherboard. (Motherboard version kit £13.50 built £14.50)

3 CHANNEL SOUND/TIMER BOARD. Kit £16.95 Built £18.95 (RE161)
- 3 independent channels.
- Controlled by BASIC.
- Full range of notes.
- Complete instructions with examples provided.
- Can be used as a sophisticated timer/counter.

MOTHERBOARD. Kit £15.75 Built £18.50 (RE82)
- Two connectors on board.
- Six connector board. TBA.

CONNECTORS & PLUGS
- 23 Way female connector for ZX80/1. (RE80) £2.95
- 23 Way male connector. (RE87) £1.30
- 23 Way male connector to fit two female connectors together. (RE90) £1.60
- 30 Way ribbon cable. £1.40 metre.
- RAM pack connector. Allows RAM pack to be remote from ZX80/1. RE170. £6.95 built.
- In/out connector & sound board connector. (RE78B) £2.95.

BOOKS & TAPES

Getting acquainted with ZX81. £4.95
Mastering machine code. £5.95
Programming for real applications. £6.95
Tape for real applications books. £11.44

Send SAE 5" x 7" for free illustrated catalogue.

All produces available ex stock (allow 7 days extra for built products.

Payment: Cash with order. Or ACCESS/BARCLAYCARD. Official order welcome. Dealers write for rates.

All prices include P+P and VAT. Overseas add £1.80

REDDITCH ELECTRONCS. DEPT ZX.
21 FERNEY HILL AVE, REDDITCH, WORCS. B97 4RU. Tel: (0527) 61240.

CUSTOMER FILE

I Wrigley of Ravenshead draws on his experience to produce a very useful program, intended for the small business with around 100 customers.

The program, written for a 16K ZX81, would be fairly easy to convert to run on other systems. It is meant for small businesses with up to 100 customers. It will print out their names and address, tell you who hasn't paid their bills, and other things you'll discover.

It's pretty easy to use the program, because all the prompts are self-explanatory. There's one point to watch, and this concerns the entering of a customer's name and address. The format for this is: "Customer's name*1st line of the address*2nd line of the address*...and so on...last line of the address NEWLINE". Entering * (shift B) after each line of the name and address ensures that it is correctly formatted when dumping on the printer.

The second point to be careful of in entering the program is that in certain lines a double asterisk (**) appears. The *must* be entered as SHIFT-H, not as SHIFT-B. The lines in question are:- 170, 2050, 2120, 4060 and 9750.

In line 5020, 80 x Z means type in the letter Z 80 times.

Lines 100 to 220 are only ever used once, to input all the customers' names and addresses.

Lines 230 to 400 are the menu selection lines. For other BASICs, line 390 will have be changed to ON B GOSUB 1000, 2000...9000 Line 380 is present because the manual state that a program cannot be SAVE'd from within a GOSUB loop.

Lines 1000 to 1040 list all the customers.

2000 to 2130 amend the name or address of a customer.

3000 to 3040 alter the amount a customer owes.

4000 to 4090 enter a new customer.

5000 to 5050 delete an old customer.

6000 to 6040 list all customers owing money.

7000 to 7040 print an address label.

8000 to 8040 save the program and its variables on tape. Some BASICs may not allow this. Saving the program from the program means that when it is LOADed, it will jump straight to the menu.

9000 to 9040 checks that the program has been SAVE'd on tape, and if it has, the machine is NEWed, clearing its memory for other uses.

9100 to 9220 is a bubblesort, to place the names in alphabetical order. It can be replaced by any other sort if required. However, if you do this, don't forget to change the amount owing (held in array A), along with the names.

9500 to 9670 asks the user for a customer's name or his customer number (the customer number may change during any run of the program) and returns the customer number in variable G. If the name is not found, the flag F is set to 1, and the subroutine prints a message.

9700 to 9840 prints out one customer's name on the printer.

If your computer supports file handling but does not save the variables on tape when the rest of the program is SAVE'd, then the variables which *must* be stored on tape are:

A$(1) to A$(100), the array which holds the customers' names and addresses,

A(1) to A(100), which holds all the amounts owing, and C, which stores the number of customers.

None of the variables need to be stored.

Using The Program

Using the program is quite easy since all the prompts within the program are self-explanatory. There is only one point to watch, and this concerns the entering of a customer's name and address.

The format for this is: "Customer's name*1st line of the address*2nd line of the address*...last line of the address newline".

Entering the * (SHIFT-B) after each line of the name and address ensures that the name and address is correctly formatted when it is output to the printer.

```
 100 DIM A$(100,80)
 110 DIM A(100)
 120 PRINT "HOW MANY CUSTOMERS?"
 130 INPUT C
 140 FOR F=1 TO C
 150 PRINT "NAME AND ADDRESS?"
 160 INPUT B$
 170 LET A$(F)= B$+"**"
 180 PRINT "AMOUNT OWING?"
 190 INPUT A(F)
 200 CLS
 210 NEXT F
 220 GOSUB 9100
 230 REM MENU
 240 CLS
 250 PRINT "ENTER:"
 260 PRINT "1 TO LIST ALL CUSTOMERS"
 270 PRINT "2 TO AMEND A CUSTOMER'S NAME OR
     ADDRESS"
 280 PRINT "3 TO ALTER THE AMOUNT OWING"
 290 PRINT "4 TO ENTER A NEW CUSTOMER"
 300 PRINT "5 TO DELETE A CUSTOMER"
 310 PRINT "6 TO LIST CUSTOMERS OWING MONEY"
 320 PRINT "7 TO PRINT AN ADDRESS LABEL"
 330 PRINT "8 TO SAVE THE PROGRAM ON TAPE"
 340 PRINT "9 TO STOP"
 350 PRINT,, "ENTER ONE OF THESE NUMBERS"
 360 INPUT B
 370 CLS
 380 IF B=8 THEN GOTO 8000
 390 GOSUB B*1000
 400 GOTO 240
1000 LET H=1
1010 LET D=1 TO C
1020 GOSUB 9700
1030 NEXT D
1040 RETURN
2000 GOSUB 9500
2010 IF F=1 THEN RETURN
2020 LET X=0
2030 LEX X=X+1
2040 IF A$(G,X)="*" THEN GOTO 2080
2050 IF A$(G,X)="**" THEN GOTO 2100
2060 PRINT A$(G,X);
2070 GOTO 2030
2080 PRINT
2090 GOTO 2030
2100 PRINT,, "NEW NAME AND ADDRESS?"
2110 INPUT B$
2120 LET A$(G)+B$+"**"
2130 GOTO 9100
3000 GOSUB 9500
3010 PRINT "OLD AMOUNT OWING £";A(G)
3020 PRINT "NEW AMOUNT?"
3030 INPUT A(G)
3040 RETURN
4000 IF C<100 THEN GOTO 4030
4010 PRINT "TOO MANY CUSTOMERS"
4020 RETURN
4030 LET C=C+1
4040 PRINT "NAME AND ADDRESS?"
4050 INPUT B$
4060 LET A$(C)=B$+ "**"
4070 PRINT "AMOUNT OWING?"
4080 INPUT A(C)
```

```
4090 GOTO 9100
5000 GOSUB 9500
5010 IF F=1 THEN RETURN
5020 LET A$(G)= " [80 x Z ]"
5030 GOSUB 9100
5040 LET C=C-1
5050 RETURN
6000 LET H=0
6010 FOR D=1 TO C
6020 IF A(D)>0 THEN GOSUB 9700
6030 IF A(D)>0 THEN LPRINT, A(D),,,,
6040 NEXT D
6050 RETURN
7000 GOSUB 9500
7010 IF F=1 THEN RETURN
7020 LET H=0
7030 LET D=G
7040 GOTO 9700
8000 PRINT "SET TAPE TO RECORD AND PLAY"
8010 PRINT "THEN PRESS NEWLINE"
8020 INPUT B$
8030 SAVE "CUSTOMER"
8040 GOTO 240
9000 PRINT "HAVE YOU SAVED THE PROGRAM ON"
9010 PRINT "TAPE?"
9020 INPUT B$
9030 IF B$(1)="N" THEN RETURN
9040 NEW
9099 REM BUBBLESORT
9100 LET J=0
9110 FOR D=1 TO C-1
9120 IF A$(D)<A$(D+1) THEN GOTO 9200
9130 LET K$=A$(D)
9140 LET K=A(D)
9150 LET A$(D)=A$(D+1)
9160 LET A$(D)=A(D+1)
9170 LET A$(D+1)=K$
9180 LET A(D+1)=K
9190 LET J=1
9200 NEXT D
9210 IF J=0 THEN RETURN
9220 GOTO 9100
9499 REM NAME OR NUMBER
9500 LET F=0
9510 PRINT "NAME OR NUMBER (1 or 2)?"
9520 INPUT J
9530 IF J=2 THEN GOTO 9630
9540 PRINT "WHAT NAME?"
9550 INPUT B$
9560 FOR K=1 TO C
9570 IF A$(K) (TO LEN(B$))=B$ THEN GOTO 9660
9580 NEXT K
9590 PRINT "NAME NOT FOUND"
9600 LET F=1
9610 PAUSE 100
9620 RETURN
9630 PRINT "WHAT NUMBER?"
9640 INPUT G
9650 RETURN
9660 LET G=K
9670 RETURN
9699 REM PRINT A CUSTOMER
9700 LET X=0
9710 IF H=1 THEN LPRINT D;
9720 LPRINT TAB(5);
9730 LET X=X+1
9740 IF A$(D,X)="*" THEN GOTO 9780
9750 IF A$(D,X)="**" THEN GOTO 9810
9760 LPRINT A$(D,X);
9770 GOTO 9730
9780 LPRINT
9790 LPRINT TAB(5);
9800 GOTO 9730
9810 LPRINT
9820 LPRINT
9830 LPRINT
9840 RETURN
```

Software review

We take a look at one of the leaders in the ZX81 business software field — Hilderbay Ltd. Hilderbay produce a number of splendid software packs for the ZX81, including STOCK CONTROL and PAYROLL, and their follow-up service and documentation is an object lesson in how professional software should be presented.

Hilderbay produce a number of splendid software packs for the ZX81, including STOCK CONTROL and PAYROLL, and their follow-up service and documentation is an object lesson in how professional software should be presented.

All programs come complete with an extensive instruction booklet. Hilderbay also offer a backup service, and will replace programs made obsolete by legislation changes. We'll look at their range program by program:

Financial Pack 1: This pack has three programs, LOAN, which — for a loan repayable in equal instalments — computes one of the following in terms of the other three, principal, number of payments, instalment and interest; VAT, which works out, and displays neat tables and running totals, as you enter information from a mixture of bills with and without VAT at different rate; and MORTGAGE, which can be used to get answers to such questions as "How much longer will your mortgage take to repay if you pay £25 less per month?" and "How much of your latest payment went towards interest?"

Budget: This program keeps track of expenses for up to 50 headings over a year. For each heading, a budget can be entered at any time. A second version of the program is available, which handles the information in a different manner. Both versions have a facility to trigger the printer for a permanent record. Both versions can be used for private or business purposes. VAT is not included explicitly, but can be handled if required as one or more separate headings.

Critical Path Analysis: This program allows the user to apply the critical path analysis technique to any projects which consist of a well-defined set of activities which may be started and stopped independently of one another and must in some cases be performed in a certain order. Michael Levy, head of the American firm Mindware, which is concentrating on providing business applications software for the ZX81 in the US, said that the CPA program from Hilderbay was one of the best he had seen for any system — and he added that he'd seen a large number running on a wide variety of machines, from IBM mainframes down to pocket computers. This program for the ZX81 will deal with problems including up to 500 activities in 16K. If you buy one of the larger memory units available, you can easily deal with many thousands of activities.

Payroll: This program which, like all the others, is supported by clear and useful documentation and comes with a follow-up service, allows employers to keep and update pay records for up to 30 employees. The basis can be weekly, monthly, or any other time interval you choose. Mike Salem, head of Hilderbay, has rightly concluded that many people who use the ZX81 in their businesses may have no knowledge of, and little interest in, computers as such, but still want to be able to use the programs without trouble. This program was designed to cater for these people. You can, if you like, write the results by hand on the usual cards, rather than using the printer. As well, you can always revert to doing it all by hand if you like You won't become dependent on the computer.

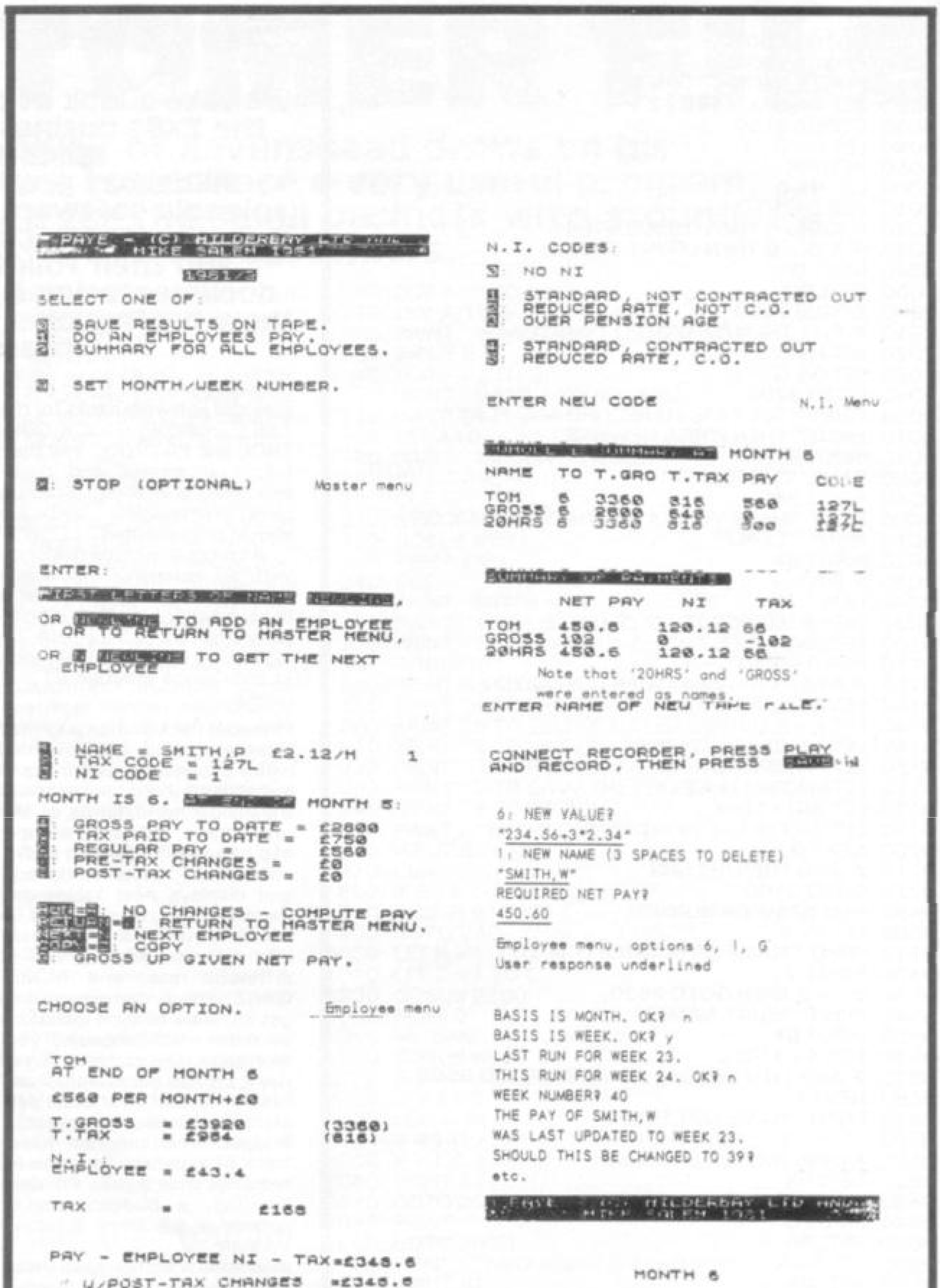
The program is designed to be saved after each use, thus saving the current variables on tape, then run next time with the command GOTO 1, rather than RUN. The current program being sold by Hilderbay is, so to speak, obsolete, because of tax and other changes, but those having earlier versions will be supplied with up to date versions. This fact shows the attention Mike Salem has paid to ensuring that he provides a proper software service, rather than just running a "post the cassette and run" operation. All payroll programs, and not just ones sold by Hilderbay, will become obsolete in 1983/84 due to legislative changes which were due to come into effect in 1982/83 but were postponed. The rules themselves will then change, not just the numbers. Until then, however, you can work out your payroll requirements without problems, knowing that if — in the meantime — the numbers change again, Hilderbay will change your program for you.

The Payroll program caters for all pay levels and all tax codes consisting of a number followed by a letter (such as 214H). National insurance contributions may be nil, standard rate, reduced rate or whatever, and there is even provision for contracted-out employees. The program will 'watch' what you're doing, and query some input if it believes you are making an error. It is impossible to pay one employee twice in one day period, for example.

Although the program is easy, and self-explanatory to some extent, Hilderbay will provide training in its use if you want it.

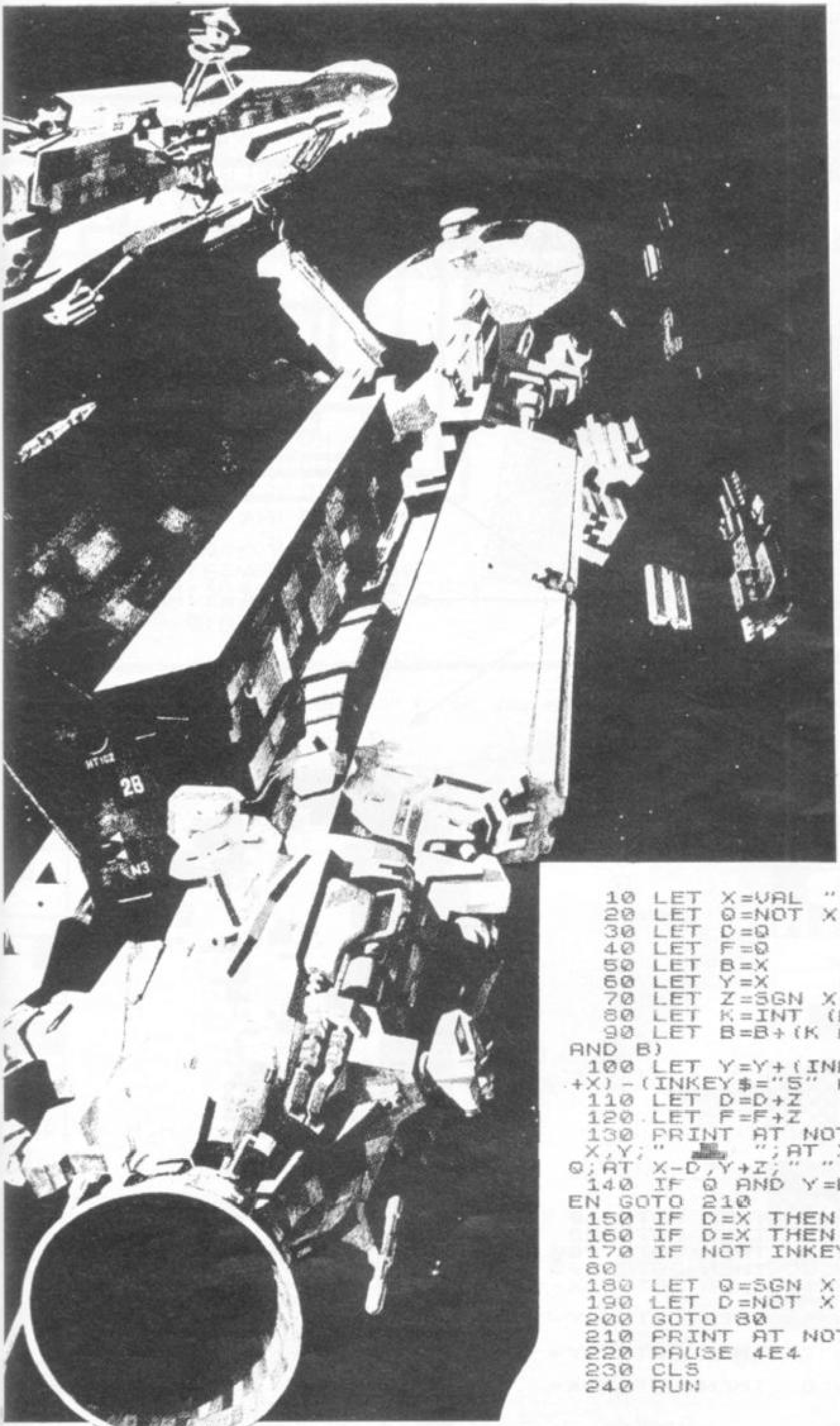
Stock Control: There are two programs on this cassette, one designed to hold a relatively large amount of information about a great many lines. "Stock I" will produce lists by supplier, by type, of understocked lines, or of all lines. Lists may be total or may cover only a certain range of the alphabetical lists. Lists are always displayed on the screen, and are printed if the ZX printer is connected. The program occupies 9K, leaving a fair amount of the 16K for data. "Stock II" identifies stock lines by a numeric code in the range zero to 65,000, storing the stock level for each line. You can handle more than 2000 lines with 16K. Both of Hilderbay's stock control programs are fairly easy to use, and allow lines to be entered or deleted at any time.

As can be seen from the above, Hilderbay have an extensive range of business software. Clive should be pleased that Mike Salem, and others like him, are doing their bit to convince the world that the ZX81 is more than a toy. Other distributors of ZX business software should be grateful to Mike Salem for showing how well it can be done.



```
PAYE - (C) HILDERBAY LTD [illegible]
       MIKE SALEM 1981
            1981/2
SELECT ONE OF:

[illegible]: SAVE RESULTS ON TAPE.
[illegible]: DO AN EMPLOYEES PAY.
[illegible]: SUMMARY FOR ALL EMPLOYEES.

[illegible]: SET MONTH/WEEK NUMBER.




[illegible]: STOP (OPTIONAL)        Master menu



ENTER:

[illegible] [illegible],

OR [illegible] TO ADD AN EMPLOYEE
  OR TO RETURN TO MASTER MENU,

OR [illegible] TO GET THE NEXT
  EMPLOYEE



[illegible]: NAME = SMITH,P   £2.12/H     1
[illegible]: TAX CODE = 127L
[illegible]: NI CODE  = 1
MONTH IS 6. [illegible] MONTH 5:
[illegible]: GROSS PAY TO DATE = £2800
[illegible]: TAX PAID TO DATE =  £750
[illegible]: REGULAR PAY =       £560
[illegible]: PRE-TAX CHANGES =   £0
[illegible]: POST-TAX CHANGES =  £0

[illegible]: NO CHANGES - COMPUTE PAY
[illegible]:  RETURN TO MASTER MENU.
[illegible]:  NEXT EMPLOYEE
[illegible]:  COPY
[illegible]: GROSS UP GIVEN NET PAY.

CHOOSE AN OPTION.          Employee menu


 TOM
 AT END OF MONTH 6

 £560 PER MONTH+£0

 T.GROSS   = £3920        (3360)
 T.TAX     = £984         (816)

 N.I.:
 EMPLOYEE  = £43.4

 TAX       =       £166


 PAY - EMPLOYEE NI - TAX=£348.6
  " W/POST-TAX CHANGES   =£348.6


DETAILS OF DEDUCTIONS, ETC, CAN
BE PRINTED BY STOPPING THE
PROGRAM (OPTION X), THEN DOING
 LPRINT "INFO",
THEN "GOTO 1 NEWLINE"
                               Payslip
```

```
N.I. CODES:

[illegible]: NO NI

[illegible]: STANDARD, NOT CONTRACTED OUT
[illegible]: REDUCED RATE, NOT C.O.
[illegible]: OVER PENSION AGE

[illegible]: STANDARD, CONTRACTED OUT
[illegible]: REDUCED RATE, C.O.

ENTER NEW CODE                  N.I. Menu


[illegible] MONTH 6
NAME  TO T.GRO T.TAX PAY   CODE
TOM   6  3360  816   560   127L
GROSS 6  2800  648   0     127L
20HRS 6  3360  816   300   127L


[illegible]
        NET PAY    NI     TAX

TOM    450.6     120.12 66
GROSS  102       0      -102
20HRS  450.6     120.12 66
     Note that '20HRS' and 'GROSS'
     were entered as names.
ENTER NAME OF NEW TAPE FILE.


CONNECT RECORDER, PRESS PLAY
AND RECORD, THEN PRESS [illegible]


 6: NEW VALUE?
 "234.56+3*2.34"
 1: NEW NAME (3 SPACES TO DELETE)
 "SMITH,W"
 REQUIRED NET PAY?
 450.60

 Employee menu, options 6, 1, G
 User response underlined


 BASIS IS MONTH. OK?  n
 BASIS IS WEEK. OK? y
 LAST RUN FOR WEEK 23.
 THIS RUN FOR WEEK 24. OK? n
 WEEK NUMBER? 40
 THE PAY OF SMITH,W
 WAS LAST UPDATED TO WEEK 23.
 SHOULD THIS BE CHANGED TO 39?
 etc.
PAYE - (C) HILDERBAY LTD [illegible]
       MIKE SALEM 1981


          MONTH 6

        [illegible]
    RESTART WITH "GOTO 1"
  [illegible]
```



UFO

This game UFO, written by S Hassen of Worthing, was written to show just how much can be squeezed into 1K without resorting to machine code. Mr Hassen tells us he has used every trick he knows to get the program into 1K. The object of the game is to shoot down the saucer hovering near the top of the screen. As you probably expect, the "5" will move you left while the "8" moves you right. Press "0" to fire. As you'll see when you run the program, the saucer does not sit still, but moves back and forth, calling for all your aiming skill. When struck (by a little full stop which zaps up to the saucer) it blows apart, and your score is shown. The lower the score, the better. Any score over 72 is lousy. Any score less than 10, I don't believe you. The game ends once you've hit the saucer, and waits until you press any key to start again. The PAUSE 4E4 (developed by Trevor Toms, author of THE ZX81 POCKET BOOK) is remembered by thinking of it as 'pause forever'. It obligingly waits, more or less, forever for you to press a key to get things underway again. If you have trouble seeing the full stop as it slowly zaps towards the target, replace it with a graphic A.

```
  10 LET X=VAL "10"
  20 LET Q=NOT X
  30 LET D=Q
  40 LET F=Q
  50 LET B=X
  60 LET Y=X
  70 LET Z=SGN X+SGN X
  80 LET K=INT (RND+RND)
  90 LET B=B+(K AND X+X)-(NOT K
AND B)
 100 LET Y=Y+(INKEY$="8" AND Y<X
+X)-(INKEY$="5" AND Y)
 110 LET D=D+Z
 120 LET F=F+Z
 130 PRINT AT NOT X,B;" ▀▀ ";AT
 X,Y;" ▄█▄ ";AT X-D,Y+Z;"." AND
Q;AT X-D,Y+Z;" " AND Q
 140 IF Q AND Y=B AND NOT D-X TH
EN GOTO 210
 150 IF D=X THEN LET Q=NOT X
 160 IF D=X THEN LET D=Q
 170 IF NOT INKEY$="0" THEN GOTO
 80
 180 LET Q=SGN X
 190 LET D=NOT X
 200 GOTO 80
 210 PRINT AT NOT X,B;")>*<( ";F
 220 PAUSE 4E4
 230 CLS
 240 RUN
```

ZX Drawing Board

This simple little program, quite apart from enabling one to draw pretty pictures on the screen of one's ZX80, shows how you can control on-screen movement. You can change the symbol in use and the direction in which you are moving at any time during the prógram execution.

Instructions and Variables

When the program is RUN an asterisk (star) is displayed on the screen. To alter the symbol in use press '5' then 'NEWLINE' followed by the code of the required symbol and 'NEWLINE'. Changes in the direction of movement can be made by keying the desired direction code (see Fig. 1.) and the inevitable 'NEWLINE'.

The program uses the following variables:

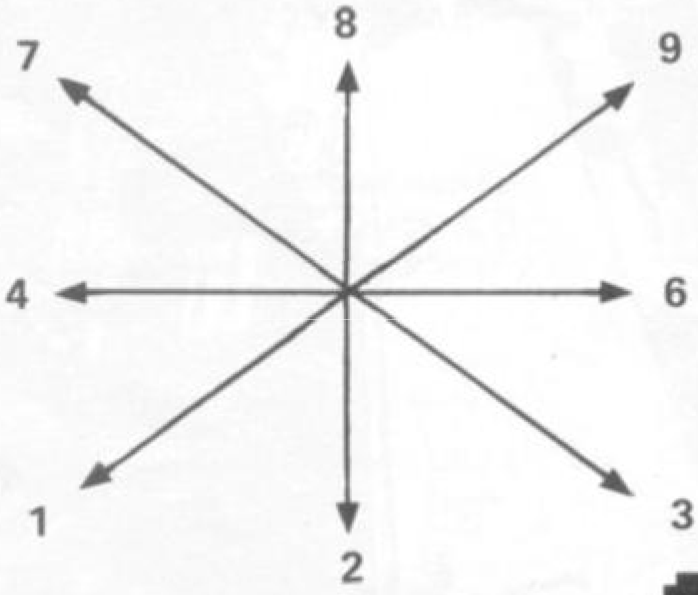
D - Position on-screen of current symbol.
C - Code of displayed symbol.
A - Direction code.
I - Dummy variable.

Sample Program

Try the following data string, the '/' represents the NEWLINE.

5/128/8/4/2/2/4/4/2/2/2/2/6/6/2/6/8/6/6/6/6/6/6/2/6/8/6/6/8/4/8/8/8/8/6/8/4/4/4/2/2/2/4/4/4/4/4/4/8/8/8/5/8/9/9/9/9.

```
 10  LET D=310
 20  LET C=20
 30  FOR I=1 T.O 640
 40  PRINT" ";
 50  NEXT I
 60  POKE PEEK(16396)+
     256*PEEK(16397)+D,C
 70  INPUT A
 80  IF A=1 THEN LET D=D+32
 90  IF A=2 THEN LET D=D+33
100  IF A=3 THEN LET D=D+34
110  IF A=4 THEN LET D=D-1
120  IF A=6 THEN LET D=D+1
130  IF A=7 THEN LET D=D-34
140  IF A=8 THEN LET D=D-33
150  IF A=9 THEN LET D=D-32
160  IF A=5 THEN INPUT C
170  GOTO 60
```



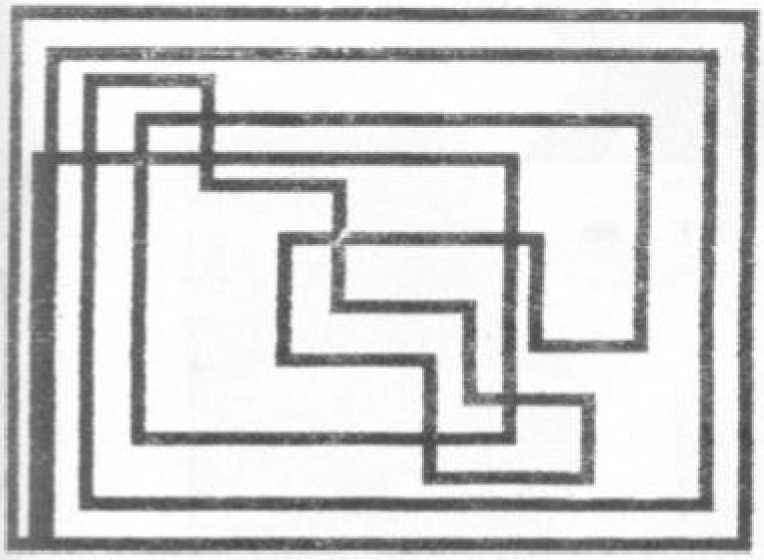
From Kowloon, Hong Kong, H. Hugh McAllum shows us a few tricks with the ZX printer.

```
  1 REM "SKETCHPAD"
  5 FAST
  6 FOR N=0 TO 63
  7 PLOT N,43
  8 PLOT N,0
  9 NEXT N
 10 FOR P=0 TO 43
 11 PLOT 0,P
 12 PLOT 63,P
 13 NEXT P
 15 SLOW
 20 LET X=32
 30 LET Y=22
 40 GOSUB 500
 50 UNPLOT X,Y
 60 PLOT X,Y
 70 GOTO 40
 80 GOSUB 500
 90 GOTO 80
100 GOSUB 500
110 PLOT X,Y
120 UNPLOT X,Y
130 GOTO 100
500 IF INKEY$="1" THEN GOTO 40
510 IF INKEY$="2" THEN GOTO 80
520 IF INKEY$="3" THEN GOTO 600
530 IF INKEY$="0" THEN GOTO 100
540 IF INKEY$="5" THEN LET X=X-
1
550 IF INKEY$="6" THEN LET Y=Y-
1
560 IF INKEY$="7" THEN LET Y=Y+
1
570 IF INKEY$="8" THEN LET X=X+
1
590 RETURN
```

```
 50 FOR N=0 TO 1000
 60 LET X=INT (RND*22)+10
 70 LET Y=INT (RND*22)
 80 IF X+Y<24 THEN GOTO 55
 90 PLOT X,Y
100 PLOT X,43-Y
110 PLOT 63-X,43-Y
120 PLOT 63-X,Y
130 NEXT N
```

Etchasketch

This very short program allows you to use the keys "5", "6", "7" and "8" to move the PLOT blob around the screen, drawing pictures of your choice. Once you've got it running, try and modify it to (a) give you a choice of starting position; and/or (b) 'turn off' the blob from time to time tomove it to a new position on the screen without leaving a trail.

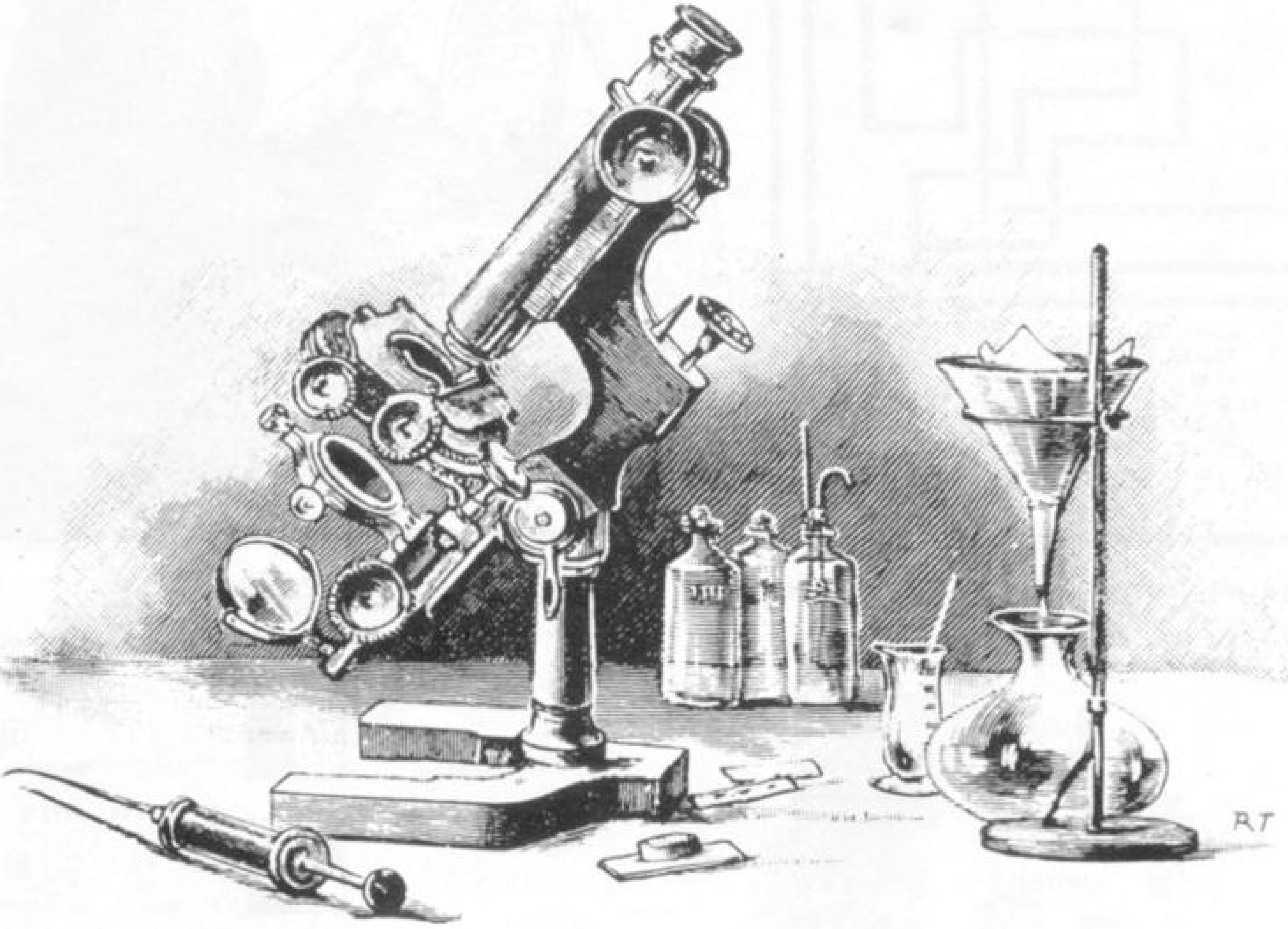


```
 10 LET A=VAL "1"
 20 LET B=A
 30 LET A$=INKEY$
 40 IF A$="" THEN GOTO 30
 60 LET A=A+(A$="7")-(A$="6")
 70 LET B=B+(A$="8")-(A$="5")
 80 PLOT B,A
100 GOTO 30
```

Keep this handy reference guide to aid you when programming your ZX81

Symbol	Code	How obtained				Symbol	Code	How obtained
	0	K or L SPACE	▒	8	G shifted A	▐	133	G shifted 8
▘	1	G shifted 1	🮏	9	G shifted D	▚	134	G shifted Y
▝	2	G shifted 2	🮎	10	G shifted S	▗	135	G shifted 3
▀	3	G shifted 7	█	128	G SPACE	▒	136	G shifted H
▖	4	G shifted 4	▟	129	G shifted Q	▀▒	137	G shifted G
▌	5	G shifted 5	▙	130	G shifted W	▒▄	138	G shifted F
▞	6	G shifted T	▄	131	G shifted 6			
▛	7	G shifted E	▜	132	G shifted R			

A CLOSER LOOK



How good is the ZX81? How does it compare with the ZX80? Does it represent a good buy for £69.95? We attempt to answer these questions for you.

It can be pretty hard trying to decide which computer you'll buy. For the benefit of those unlucky souls who do not yet own a ZX81, Peter Freebrey takes a close look at the ZX81, and compares it with its predecessor, the ZX80.

The ZX81 gives the impression of being more robust than the ZX80, and although the keyboard is still the now familiar 'touch type' as on the ZX80, I find the slightly textured matt finish gives me a greater feeling of control than its glossy forerunner. The ZX81 comes complete with separate mains power supply and leads to connect it to your TV set and tape recorder. The 212-page instruction manual is comprehensive and is Sinclair's best yet. The ZX81 uses the new 8K ROM which was announced in 1980.

There are some 20 commands and/or statements that were not available to the ZX80. From the list of these in Table 1 you can see that they include the option of a FAST or SLOW mode — this is certainly a useful improvement. When running in the FAST mode the ZX81 operates in a similar manner to the ZX80 — fast operation, with a very noticeable flicker on the display whenever a key is operated or command actioned together with a blank grey screen while computations are taking place. In SLOW mode operation you get a flicker-free picture.

Although the ZX81 does not have a memory mapped display, it refreshes and updates the screen information while still proceeding with the program it is running. This mode is much slower in operation: a FOR . . . NEXT loop of 10000 takes 176 S in SLOW and 44 S in FAST mode. Even though SLOW really does mean slower operation it gives the user the option of using moving graphics-apart from making the display less of a strain to the eyes!

Whereas the ZX80 was limited to integer calculations the ZX81 has a full floating point notation and this has

meant the addition to its repertoire of such functions as logs and trig. These, together with its ability to PLOT pixels (quarter square graphics) means you can draw, among other things, sinewaves across your TV screen to your heart's content!

Another advantage of the new 8K ROM is the option of adding a printer to your system. The ZX printer offers full alphanumerics across 32 columns and the ZX81 has the commands necessary to LPRINT and LLIST to the printer. It also has the facility to COPY, which will print out exactly what is displayed on the TV screen without further commands.

Programming Techniques

For those not familiar with the ZX80, the ZX81 uses a 'one touch' keyword entry technique — when you have entered a program line number the next requirement must be a command of some description. The ZX81 recognises this necessity and the next single key pressed will result in the related command being entered and appearing on the screen in front of you. If you have entered a line number and then press key 'P' you will actually enter PRINT, press 'H' and get GOSUB, and key 'N' gives NEXT. Likewise, after THEN a command is expected: key 'P' again gives PRINT, 'Y' gives RETURN, and so on. Many keys are multifunction: some are capable of generating up to three keywords, a letter, that letter reversed and also a graphics character!

Initially this can be a little daunting, not to say confusing, but you will soon get used to the operation of the keyboard. It has a fairly logical sequence and like any new keyboard can pose an unfamiliar operator with a few initial problems.

The cursor takes the form of a reversed character: if 'K' is displayed this indicates that a keyword will be generated, should the next key pressed have such a function. If the cursor displays a 'G' then either a graphics symbol or a reversed character will appear. A reversed 'F' indicates that a further set of keywords such a SIN, COS, RND will be called. Should a reversed 'S' be displayed then you have a syntax error and the ZX81 will not allow the entry of that line to the program until you have made the necessary correction.

Reading Matters

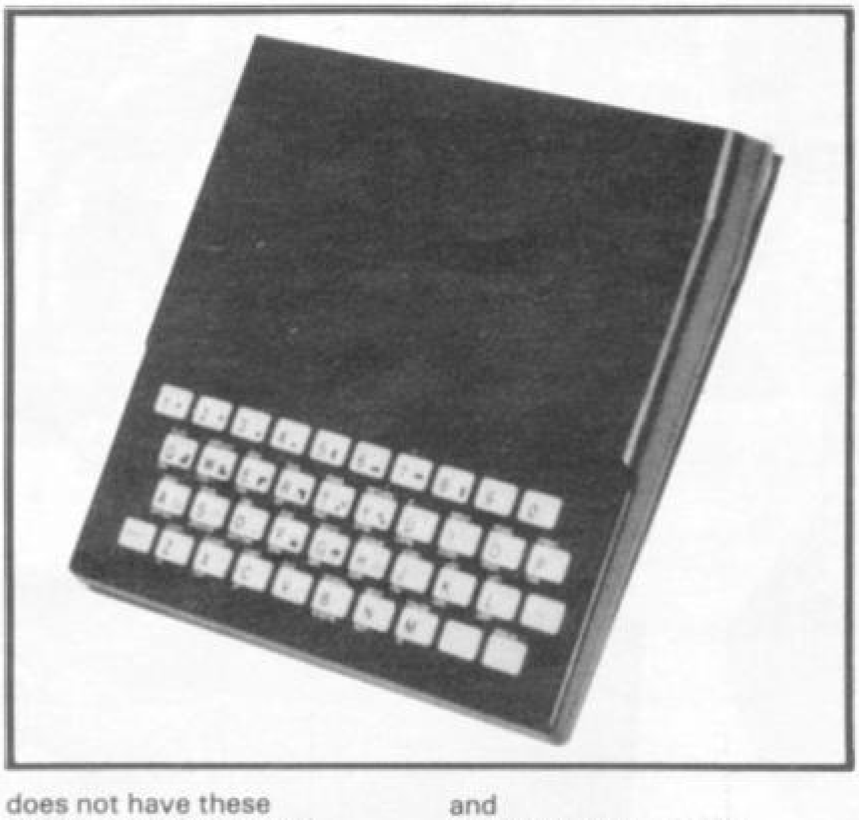
The manual has 28 chapters and three appendices. Chapter one gives instructions on how to connect and set up the ZX81 for operation initially. It also recommends that those already knowing BASIC should read Appendix C to familiarise themselves with ZX81 BASIC and use the remainder of the instruction manual as reference as and when needed. The main bulk of the book is for the novice, explaining clearly and concisely all the statements and commands. Most of the chapters are essentially a 'hands-on' teaching programme in ZX81 BASIC, with a number of useful exercises to extend the knowledge gained in the preceding text. Chapters 26, 27 and 28 introduce the idea of machine code, the organisation of the memory, and explain what the system variable are and where they are stored.

Appendix A lists the character set, Appendix B gives the Report (Error) codes and Appendix C give a short résumé of the ZX81 and its individual characteristics under the heading: "The ZX81 for those that understand BASIC". The Index is useful and comprehensive, giving not only page references but also the key sequences required for all keywords and shifted characters.

Those who already have some knowledge of BASIC will note from the command/statement list that there are a few common BASIC commands that are noticeable by their absence, principally DATA, READ, RESTORE, ON-GOTO, LEFT$, MID$, RIGHT$ and TL$. These are listed in the index but not in the bold type, indicating that they are ZX81 keywords. Although the ZX81 does not have these commands, the manual explains the use of simple routines to obtain access to the first four effectively. The latter four deal with string manipulation, and whereas the ZX81 does not have these specific commands the way in which it deals with strings is comprehensive, albeit non-standard. It uses a notation called 'slicing' for describing substrings. This can be applied to arbitrary string expressions and takes the general form . . .'string expression (start TO finish)' so that

"ABCDEF" (2 TO 5) = "BCDE"

and
"ABCDEF"(3) = "C"



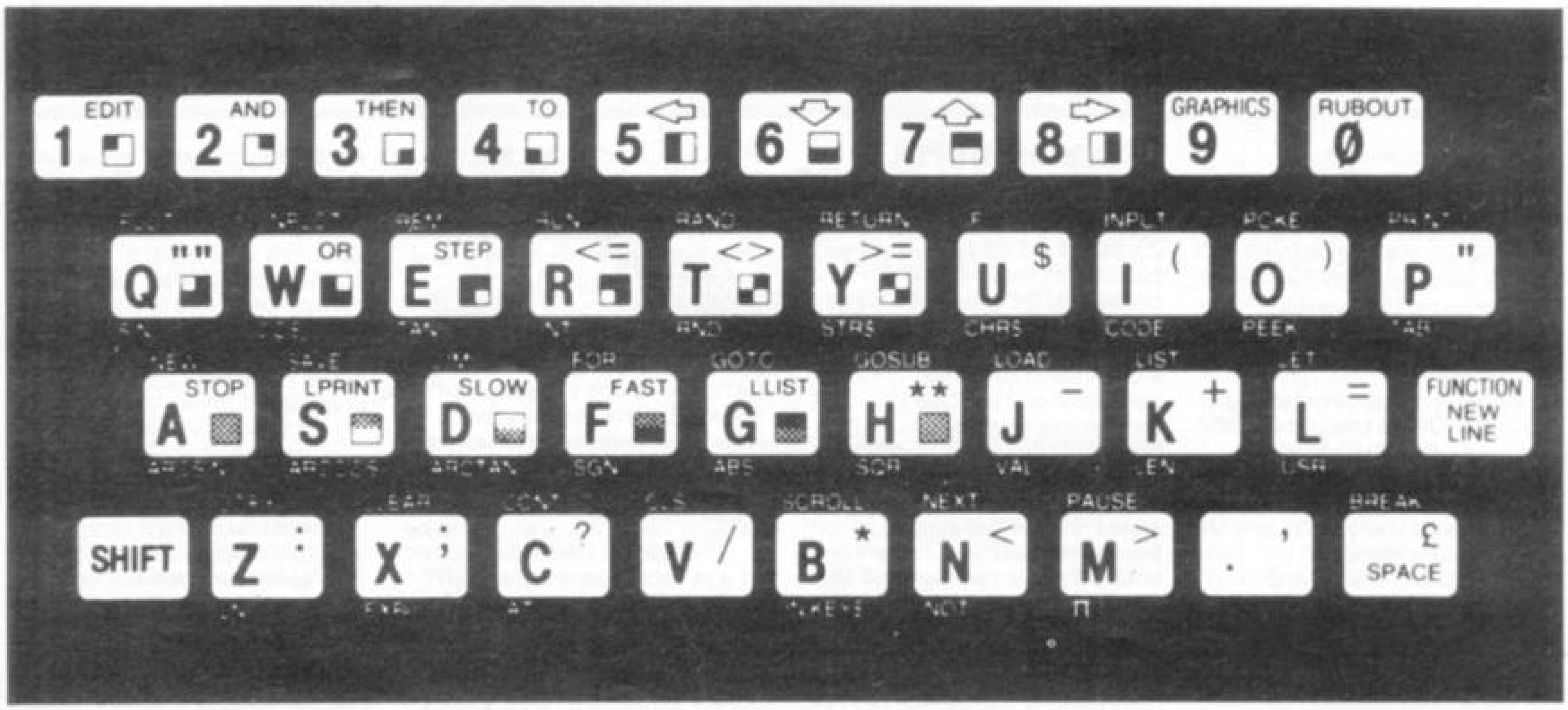
Different as this may be from other common micros it is easily understood and will readily yield the results you would expect from the more usual string commands.

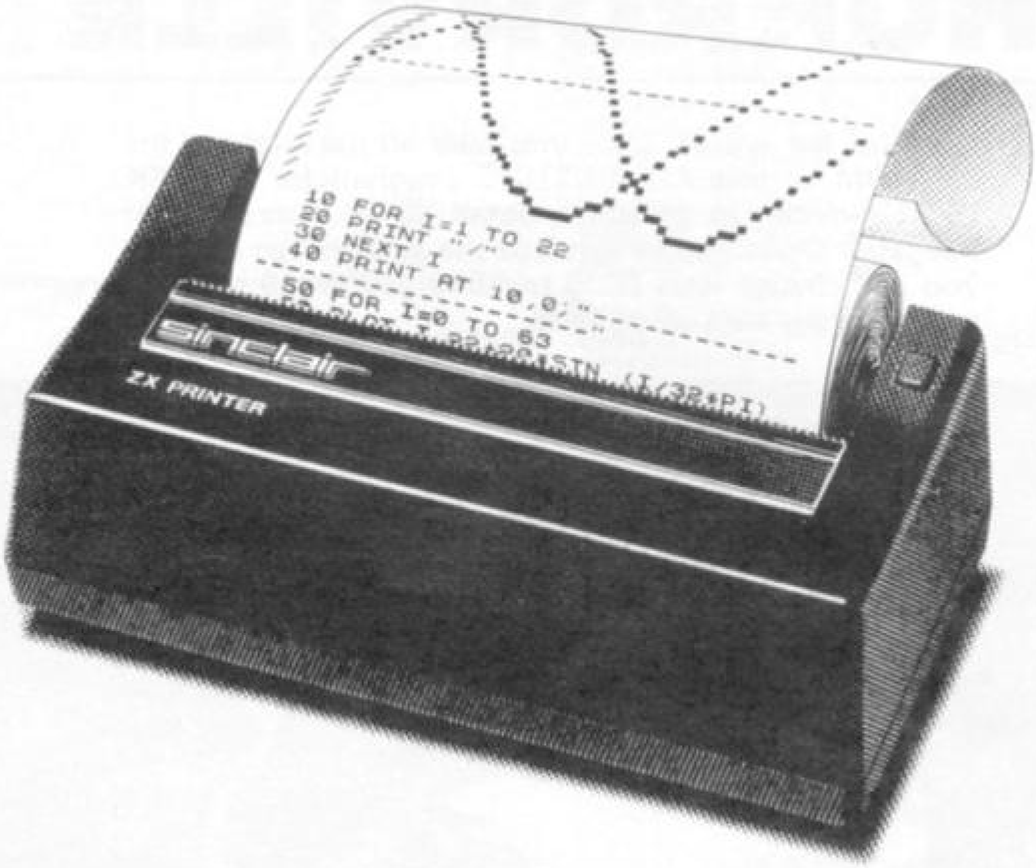
Using It

In use, the ZX81 presents no real lasting problem to operate. If you are new to using a computer then you have probably used a calculator and so the keyboard spacing will not seem all that different. If, on the other hand, you regularly use a typewriter keyboard, you will take a few hours to adapt to the diminutive size of the ZX81: but then those few hours will enable you to become accustomed to the fact that some keys may give up to six different results! One very small but important improvement on the ZX81 is its rubber feet. The ZX80 and ZX81 are both very light in weight, and with its new feet at least the ZX81 does not skate all over the table as you use it!

All in all, I can recommend the ZX81 to any intending purchaser — it is very good value for money. It does *not* have the facilities one would require for a business machine but anyone wanting to learn *personally* what a computer can do, without initially spending a fair amount of money, should seriously consider one. After all, the ZX81 does not cost much more than the cassette recorder that you will need to buy for some makes of micro.

The only personal drawback to the whole Sinclair ZX episode is that my seven-year-old son tends to look upon the ZX80 and the ZX81 as more his size! He does not have quite the reverence for the ZXs as he does for other micros of the desktop variety. Demand for its use seems to be unending and we often hear "Press NEWLINE, silly!" as he instructs his five-year-old sister in the uses of twentieth-century technology.





```
10 FOR I=1 TO 22
20 PRINT "/"
30 NEXT I
40 PRINT AT 10,0;"----------------"
50 FOR I=0 TO 63
60 PLOT I,22+20*SIN (I/32*PI)
```

SYSTEM/MULTIFUNCTION COMMANDS	
BREAK	Halts run and returns to command mode.
CLEAR	Deletes all variables, freeing the space they occupied.
CLS	Clears screen — clears display file.
CONT	Continues STOPped program.
COPY	Sends copy of display to printer.
EDIT	Returns current line (indicated in program list) to bottom of screen for editing.
FAST	Starts fast mode. Display file is displayed only at end of program, while INPUT data is being typed in or during PAUSE.
FUNCTION	Returns alternative keyword set.
GRAPHICS	Returns alternative character set.
LIST	Lists specified line(s) of program on screen.
LLIST	Like LIST but using printer instead of screen.
LOAD	Looks for specified program on tape — loads it and its variables.
LPRINT	Like PRINT but using printer instead of screen.
NEW	Clears memory and awaits new program.
NEWLINE	Enters command as statement.
PAUSE	Stops computing and outputs the display file to the screen for specified time, or until another key is pressed.
RUBOUT	Deletes character to left of cursor.
RUN	Executives current program.
SAVE	Writes specified program to cassette interface. Records program and variables.
SLOW	Puts computer into compute and display mode, in which the display file is displayed continuously.
STOP	Stops a program that is RUNning.
STATEMENTS	
ABS	Returns absolute value of specified variable.
ACS	Returns arcosine (in radians).
AND,OR,NOT	Comparative tests.
ASN	Returns arcsine (in radians).
AT	Defines position of next PRINT statement (in screen lines/columns).
ATN	Returns arctangent (in radians).
CHR$	Gives character whose code is specified.
CODE	Gives code of the first character in specified string.
COS	Returns cosine of angle (in radians).
DIM	Dimensions array size.
EXP	Returns exponential of specified number.
FOR	Used in conjunction with TO and NEXT to execute a defined loop.
GOSUB	Jumps to defined subroutine.
GOTO	Jumps to specified program line number.
IF	Conditional test, used in conjunction with THEN followed by specified statement.
INKEY$	Reads keyboard, result is character of next key pressed.
INPUT	Assigns value of keyboard entry to specified variable.
INT	Returns integer part of number (rounded down).
LEN	Returns length of specified string.
LET	Assigns specified value to specified variable.
LN	Returns natural logarithm of a number.
PEEK	Returns value of specified memory location.
PI	TI(3.14159265 . .).
PLOT	Blacks in pixel at specified co-ordinates.
RAND	Sets the seed used to generate the next value of RND.
REM	No effect on program, allows inclusion of text for comments.
RETURN	Returns to main program from subroutine.
SCROLL	Scrolls the display file up one line.
SGN	Returns sign of number.
SIN	Returns sine of angle (in radians).
SQR	Gives square root of number.
STEP	Used with FOR . . . NEXT loops, defining increment between loops.
STR$	Returns string representation of number.
TAB	Defines column in which PRINT statement will begin.
TAN	Returns tangent of angle (in radians).
UNPLOT	Like PLOT, but blanks out a pixel instead of blacking in.
USR	Calls machine code subroutine.
VAL	Evaluates a string as a numerical expression.

TABLE 1

HORRORVILLE

Only for the brave! Dare you take on the might of the 8K ROM in this ADVENTURE program for the 16K ZX81, written in gruesome detail by N. Alexander of Margate. Once you've survived the Alexandran horrors, you can change lines 6000 to 6500, and 8000 to 8500 to enter your own adventures.

```
  1 REM ADVENTURE
  2 RAND
  5 DIM H(3)
 10 DIM Z(15)
 15 LET A=135
 20 GOSUB 9000
 30 GOSUB 1500
 40 CLS
 50 LET A$="SNAKE"
100 GOSUB 1000
110 PRINT "N,E,S OR W ?"
115 PAUSE 9999
120 LET Y$=INKEY$
130 IF Y$="N" THEN LET B=-33
```

```
140 IF Y$="S" THEN LET B=33
150 IF Y$="E" THEN LET B=1
160 IF Y$="W" THEN LET B=-1
170 LET A=A+B
175 IF PEEK (P+A)=0 THEN GOTO 9
800
190 IF A=175 THEN GOTO Z(1)
200 IF A=446 THEN GOTO Z(2)
210 IF A=180 THEN GOTO Z(3)
220 IF A=531 THEN GOTO Z(4)
230 IF A=286 THEN GOTO Z(5)
240 IF A=186 THEN GOTO Z(6)
250 IF A=290 THEN GOTO Z(7)
260 IF A=618 THEN GOTO Z(8)
```

```
 270 IF A=275 THEN GOTO Z(9)
 280 IF A=540 THEN GOTO Z(10)
 290 IF A=371 THEN GOTO Z(11)
 300 IF A=239 THEN GOTO Z(12)
 310 IF A=452 THEN GOTO Z(13)
 320 IF A=161 THEN GOTO 7501
 330 GOTO 100
1000 PRINT AT 0,0;"       A D V
E N T U R E"
1010 PRINT
1012 PRINT " START       █ █
    END"
1014 PRINT "             █ █"
1016 PRINT "████████████████████
███████████"
1020 PRINT "  █    █████████████
████    █  "
1030 PRINT "  █    █         █
   █    █  "
1040 PRINT "  █    █         █
   █    █  "
1050 PRINT "  ██████████████████
█████████  "
1060 PRINT "       █    █   █
█  █       "
1070 PRINT "       █    █   █
█  █       "
1060 PRINT "       █    █   █
█  █       "
1090 PRINT "       █    █   █
█  █       "
1100 PRINT "       █    █   █
████       "
1110 PRINT "  ██████   ██   █
   █       "
1120 PRINT "  █        █    █
   █████   "
1130 PRINT "  █        █    █
       █   "
1140 PRINT "  █        █    █
       █   "
1150 PRINT "  ██████████    ████
████████   "
1155 LET P=PEEK (16396)+256*PEEK
 (16397)
1160 POKE P+A,180
1165 RETURN
1500 GOSUB 9050
1505 PRINT " YOU ARE CAPTIVE IN
A GIANTS"
1510 GOSUB 9050
1515 PRINT "CASTLE AND THE ONLY
WAY YOU CAN"
1520 GOSUB 9050
1525 PRINT "ESCAPE IS TO TRAVEL
THROUGH "
1530 GOSUB 9050
1535 PRINT "A NETWORK OF UNDERGR
OUND CAVES"
1540 GOSUB 9050
1545 PRINT "BUT BEWARE"
1550 GOSUB 9050
1555 GOSUB 9050
1560 PRINT "YOU WILL COME UP AGA
INST "
1565 GOSUB 9050
1570 PRINT "PERILS AND DECISIONS
"
1575 GOSUB 9050
1580 PRINT "YOU MAY BE OFFERED V
ARIOUS"
1585 GOSUB 9050
1590 PRINT "OBJECTS BUT YOU MAY
ONLY CARRY"
1595 GOSUB 9050
1600 PRINT "ONE AT A TIME AND MU
ST DECIDE"
1605 GOSUB 9050
1610 PRINT "WHICH WILL BE MOST U
SEFUL"
1615 GOSUB 9050
1620 PRINT "READY?    PRESS ANY K
EY TO START"
1625 PAUSE 9999
```

```
1630 RETURN
2000 CLS
2005 PRINT "ON YOUR LEFT YOU CAN
 SEE TWO ","KEYS DO YOU WANT TO
PICK ONE UP?"
2010 INPUT Y$
2012 IF Y$="N" THEN CLS
2015 IF Y$="N" THEN GOTO 100
2020 PRINT "WHICH KEY DO YOU WAN
T,1 OR 2?"
2025 INPUT Y
2030 LET R$="KEY ONE"
2035 IF Y=2 THEN LET R$="KEY TWO
"
2040 CLS
2045 PRINT "REMEMBER YOU ARE NOW
 CARRYING",R$
2050 FOR I=1 TO 50
2055 NEXT I
2060 CLS
2065 GOTO 100
2500 CLS
2505 PRINT "ON YOUR RIGHT IS A P
LANK OF WOOD",,"DO YOU WANT TO C
ARRY IT?"
2510 INPUT Y$
2515 IF Y$="N" THEN CLS
2520 IF Y$="N" THEN GOTO 100
2525 LET R$="WOOD"
2530 CLS
2535 PRINT "REMEMBER YOU ARE CAR
RYING",R$
2540 FOR I=1 TO 50
2545 NEXT I
2550 CLS
2555 GOTO 100
3000 CLS
3005 PRINT "AT YOUR FEET IS A DI
AMOND","NECKLACE DO YOU WANT TO
PICK","IT UP"
3010 INPUT Y$
3015 IF Y$="N" THEN CLS
3020 IF Y$="N" THEN GOTO 100
3025 PRINT "DONT FORGET:- YOU HA
VE PICKED","UP A NECKLACE"
3030 LET R$="NECKLACE"
3035 FOR I=1 TO 50
3040 NEXT I
3045 CLS
3050 GOTO 100
3500 CLS
3505 PRINT "DO YOU WANT TO PICK
UP A PIECE ","OF ROPE?"
3510 INPUT Y$
3515 IF Y$="N" THEN CLS
3520 IF Y$="N" THEN GOTO 100
3525 PRINT "THE ROPE IS YOURS."
3530 LET R$="ROPE"
3535 FOR I=1 TO 50
3540 NEXT I
3545 CLS
3550 GOTO 100
4000 CLS
4010 PRINT "YOU HAVE TO CLIMB UP
 A SMALL","CLIFF .HAVE YOU GOT A
 ROPE?"
4015 INPUT Y$
4020 IF Y$="Y" AND R$="ROPE" THE
N GOTO 4200
4021 IF Y$="N" THEN GOTO 4023
4022 PRINT "OH NO YOU HAVENT"
4030 PRINT "THEN YOU MUST BORROW
 ONE OF ","OURS ONE IS SAFE AND
ONE IS NOT","CHOOSE YOUR ROPE:-1
 OR 2"
4032 LET H$="2"
4035 IF RND<.5 THEN LET H$="1"
4040 INPUT I$
4045 IF I$=H$ THEN GOTO 4200
4050 PRINT "YOU CHOSE WRONG SPLA
T"
4055 GOTO 9800
4200 PRINT "OK.YOU MAY CONTINUE"
4205 FOR I=1 TO 50
4210 NEXT I
```

```
4215 CLS
4220 GOTO 100
4500 CLS
4505 PRINT "YOU MUST MEET THE OG
RE","HAVE YOU A GIFT FOR HIM?"
4510 INPUT Y$
4515 IF Y$="N" THEN GOTO 4700
4520 IF RND<.8 THEN GOTO 4800
4530 PRINT "SORRY THE OGRE DOES
NOT LIKE",R$
4535 FOR I=1 TO 90
4540 PRINT "RIP  ";
4545 NEXT I
4550 GOTO 9800
4700 PRINT "THEN BE ON YOUR WAY
AND DO NOT","TRY TO COME THIS WA
Y AGAIN"
4705 LET A=A-B
4710 CLS
4715 GOTO 100
4800 IF R$="O" THEN GOTO 4530
4805 PRINT "OK HE LIKED THE ";R$
4806 FOR I=1 TO 50
4807 NEXT I
4810 CLS
4815 GOTO 100
5000 CLS
5005 PAUSE (RND*10)
5010 PRINT "YOU HAVE COME TO THR
EE DOORS ","CHOOSE YOUR DOOR,1,2
 OR 3"
5020 INPUT Y
5030 LET X=RND*4
5040 FOR I=1 TO 3
5050 LET X=X+1
5060 IF X>3 THEN LET X=1
5070 GOSUB 5080+(I*10)
5075 NEXT I
5076 LET A=H(Y)
5079 CLS
5080 GOTO 100
5090 LET H(X)=185
5095 RETURN
5100 LET H(X)=291
5105 RETURN
5110 LET H(X)=519
5115 RETURN
5500 CLS
5510 PRINT "AHEAD IS A DOOR IN F
RONT OF ","WHICH IS A LARGE SLEE
PING GIANT","HAVE YOU GOT A KEY?
"
5520 INPUT Y$
5530 IF Y$="N" THEN GOTO 5900
5540 IF R$="KEY ONE" OR R$="KEY
TWO" THEN GOTO 5600
5550 PRINT "OH NO YOU HAVENT"
5560 GOTO 5900
5600 LET T=RND
5610 IF T<.2 THEN PRINT "THE GIA
NT ATE YOU "
5620 IF T<.2 THEN GOTO 9950
5630 IF T>.7 THEN PRINT "SORRY Y
OUR KEY DIDNT FIT NOW ","THROW I
T AWAY"
5640 IF T>.7 THEN LET R$="O"
5650 IF T>.7 THEN LET A=A-B
5660 IF T>.7 THEN GOTO 5900
5670 PRINT "OK YOUR KEY FITS YOU
 MAY ","CONTINUE"
5680 GOTO 5910
5900 PRINT "SO BE ON YOUR WAY"
5905 LET A=A-B
5910 FOR I=1 TO 50
5920 NEXT I
5930 CLS
5940 GOTO 100
6500 CLS
6505 LET T$="TOILET ROLL"
6510 IF RND<.3 THEN LET T$="SHOE
"
6520 IF RND>.6 THEN LET T$="DEAD
 FISH"
6530 PRINT "AHA YOU SAY I CAN ",
"SEE A ";T$
6540 PRINT "BUT WILL YOU PICK IT
 UP?"
6550 INPUT Y$
6560 IF Y$="N" THEN CLS
6570 IF Y$="N" THEN GOTO 100
6580 LET R$=T$
6590 PRINT "NOW YOU HAVE YOUR VE
RY OWN ";T$
6600 FOR I=1 TO 50
6610 CLS
6620 GOTO 100
7000 CLS
7010 PRINT "YOU ARE CARRYING ",R
$;" SO DO YOU","WANT TO PICK UP
SOME VACCINE","INSTEAD?"
7020 INPUT Y$
7030 IF Y$="Y" THEN LET R$="VACC
INE"
7040 CLS
7050 GOTO 100
7490 GOTO 8500
7500 GOTO 8500
7501 CLS
7510 FOR I=1 TO 42
7520 PRINT "CONGRATULATIONS";
7530 NEXT I
7540 GOTO 9950
8500 CLS
8505 PRINT "YOU HAVE COME ACROSS
 A PLAGUE ","OF RATS"
8510 IF R$="VACCINE" THEN PRINT
"BUT LUCKILY YOU CAN USE YOUR","
VACCINE .YOU MAY CONTINUE"
8520 IF R$="VACCINE" THEN GOTO 8
900
8530 IF RND<.2 THEN GOTO 8600
8540 PRINT "YOU HAVE NO VACCINE
RUN QUICKLY"
8545 LET A=A-B
8550 GOTO 8900
8600 PRINT "THE RATS BIT YOU AND
 YOU HAVE","NO VACCINE SORRY"
8610 GOTO 9950
8900 FOR I=1 TO 50
8910 NEXT I
8915 IF R$="VACCINE" THEN LET R$
="O"
8920 CLS
8930 GOTO 100
9000 LET X=RND*14
9002 FOR N=2000 TO 8500 STEP 500
9005 LET X=X+1
9010 IF X>13 THEN LET X=1
9015 LET Z(X)=N
9020 NEXT N
9025 RETURN
9050 SCROLL
9055 FOR N=1 TO 6
9060 NEXT N
9065 RETURN
9950 PRINT "AGAIN?"
9960 INPUT Y$
9970 IF Y$="Y" THEN RUN
```

Subscriptions

Make sure you get every issue of ZX Computing.

ZX
COMPUTING
Summer 1982
Britain's Biggest Magazine For The Sinclair User
Over 100 pages of information and programs for the ZX81 and ZX80 user including a 1K chess routine!
PLUS
• Software Reviews — which programs are the best buy?
• Business Routines — put your ZX to work!
• Expansion Systems — how good and how much?
• Machine Code for ZX-81 — secrets revealed at last!
• DIY Memory Upgrade — cheap way of adding bytes

Just £7.00 will ensure the next four issues will be lovingly wrapped and posted to you. Just fill in the form below, cut it out and send it with your cheque or postal order (made payable to ASP Ltd) to:-

ZX Computing Subscriptions
513 London Road,
Thornton Heath,
Surrey CR4 6AR

Alternatively you can pay by Access or Barclaycard in which case simply fill in your card number, sign the form and send it off. Do NOT send your card!

Make the most of your ZX computer with ZX Computing.

Subscription Order Form

Cut out and SEND TO:

ZX COMPUTING Subscriptions
513, London Road,
Thornton Heath,
Surrey, England.

Please commence my subscription to ZX Computing with the autumn issue.

SUBSCRIPTION RATES

(tick — as appropriate)

£7 for 4 issues
UK
£15.15 for 12 issues
Overseas and air-mail rates upon application.

I am enclosing my (delete as necessary)
Cheque/Postal Order/International Money
Order for £..........
(made payable to A.S.P. Ltd)
OR
*Debit my Access/Barclaycard**
*(*delete as necessary)*

□□□□□□□□□□□□□□□□□□

Please use BLOCK CAPITALS and include post codes.

Name(Mr/Mrs/Miss) ....
delete accordingly

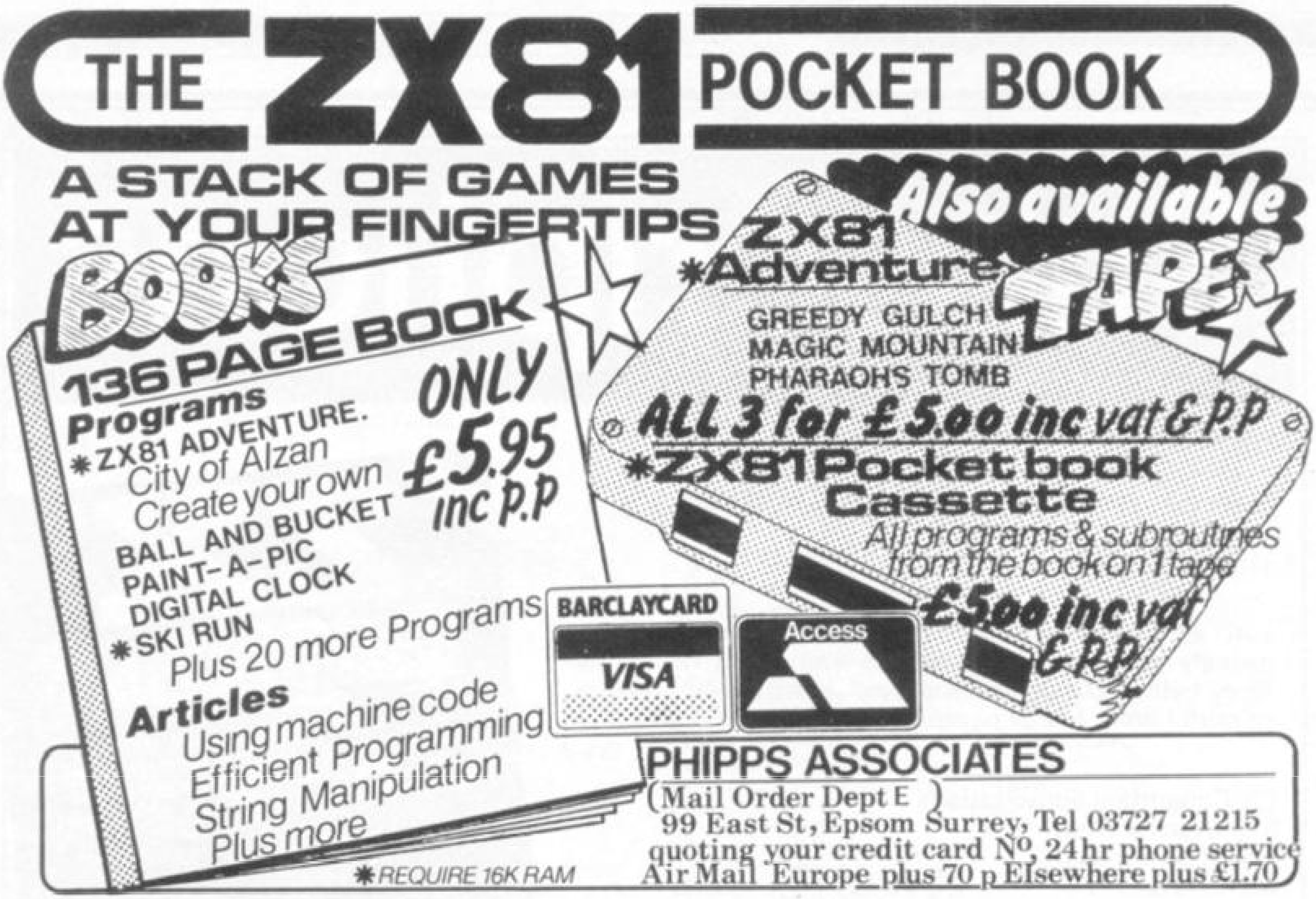
Address ....

....

....

Signature ....

Date..................................



THE ZX81 POCKET BOOK

A STACK OF GAMES AT YOUR FINGERTIPS

BOOKS

136 PAGE BOOK

ONLY £5.95 inc P.P

Programs
- ✱ZX81 ADVENTURE. City of Alzan
- Create your own BALL AND BUCKET
- PAINT-A-PIC
- DIGITAL CLOCK
- ✱SKI RUN
- Plus 20 more Programs

Articles
- Using machine code
- Efficient Programming
- String Manipulation
- Plus more

✱REQUIRE 16K RAM

Also available TAPES

ZX81 ✱Adventure

GREEDY GULCH
MAGIC MOUNTAIN
PHARAOHS TOMB

ALL 3 for £5.00 inc vat & P.P

✱ZX81 Pocket book Cassette

All programs & subroutines from the book on 1 tape

£5.00 inc vat & P.P

BARCLAYCARD VISA

Access

PHIPPS ASSOCIATES
(Mail Order Dept E)
99 East St, Epsom Surrey, Tel 03727 21215
quoting your credit card Nº, 24hr phone service
Air Mail Europe plus 70 p Elsewhere plus £1.70

HEWSON CONSULTANTS

HINTS AND TIPS for the ZX81 by Andrew Hewson **£4.25**

"Good value and quite fascinating . . . a very inexpensive way of acquiring months of programming experience" YOUR COMPUTER Nov 81.

"Excellent . . . very good value for money" SYNC

Saving Space . . . Understanding the Display File . . . Converting ZX80 Programs . . . Chaining Programs (pass data between programs, call subroutines from cassette, establish data files) . . . Machine Code Programs (write, load, edit, save and debug machine code).

Routines and programs are scattered liberally throughout the text and the final chapter consists of twelve useful, interesting and entertaining programs.

16K RAM PACK £37.50

A top quality add-on 16K dynamic memory specially designed for the ZX81. Simply plugs into the port at the back of your Sinclair. Can be used in conjunction with the ZX printer. Neatly packaged in a black plastic shell to match your ZX81. Incredible value, why pay more?

SPECIAL OFFER a **free** copy of our super Space Intruders cassette will be sent with every 16K RAM order before 1.5.82.

Z80 OP CODES £1.45

A must for the beginner and the experienced programmer alike. This handy ready reckoner lists all 600 plus Z80 machine code instructions in decimal and hexadecimal with their mnemonics. Each Op code is succinctly explained and cross-referenced. Supplied in a protective transparent wallet for easy reference and durability.

MEMOPACK 64K RAM £79.00

Developed by Memotech this amazing 64K RAM needs **no** extra power supply. Simply plugs neatly and firmly into the rear port of your ZX81. Can be used with the ZX printer and other add-on hardware.

Just look at these features
- 8-16K can be switched in and out in 4K blocks to leave space for memory mapping.
- 12-16K can be protected against overwriting during loading from cassette.
- Directly addressable and user transparent.
- Up to 15K of basic program area.
- 32K or more of variable area.

PROGRAMMERS TOOLKIT £6.50

Are you writing your own programs for the ZX81? Then use our TOOLKIT to do the donkey work. Copy into RAM before you start work and then you will have it at your fingertips. Comprehensive LINE RENUMBER including GOTOs and GOSUBs; LOAD, EDIT and RUN machine code programs; INSPECT the ZX81 system routines; COPY them into RAM and PATCH and/or EXTEND them; FIND a given piece of Basic code and REPLACE all occurrences of it; move blocks of Basic lines with EDIT.

SPACE INTRUDERS £5.95

40 alien ships in each squadron. All action display. Automatic option — the machine plays itself — can you do better? Written in machine code for super fast fun. Squadron after squadron attack your position. Three laser bases. Full score display.

STATISTICS £3.75

Three programs on the one cassette.

1) i) Statistics — prints the current mean and standard deviation after each value is entered.
ii) Regression — prints the current mean and standard deviation of the y and x values and the intercept and slope of the regression line.
iii) Trend — prints the current mean and standard deviation of the y and x values and the intercept and slope of the trend lines.

2) CHI SQUARED TEST — prints the current value of the Chi squared statistic and the current sample size.

3) GRAPH PLOT — plots a graph of data entered from the keyboard.

Cheque with order or quote Access or Barclaycard number to: HEWSON CONSULTANTS Dept L, 7 Grahame Close, Blewbury, Didcot, Oxon OX11 9QE. TEL: (0235) 850075.

ZX80 fights back

Music

Kev Molloy decides the days of silent movies are over, and produces a routine to turn your ZX80 into a music machine.

Kev's program plays a passable version of Greensleeves in this program, which is stored as a string. The length of each note is stored in another string. The higher the number (from one to eight) the longer the length of the note. The key for notes is as follows:
D — V; D sharp — R; E — P; F — N; F sharp — L; G — J; G sharp — M; A — G; B flat — E; B — C; C — A; C sharp — 8; D — sharp; D sharp — 6; E — 5; F — 4; F sharp — 3; G — 1; G sharp — 0. Although this is only a rough guide, it works on most occasions and whenever it sounds a bit off, all you need to do is add or subtract one or two.

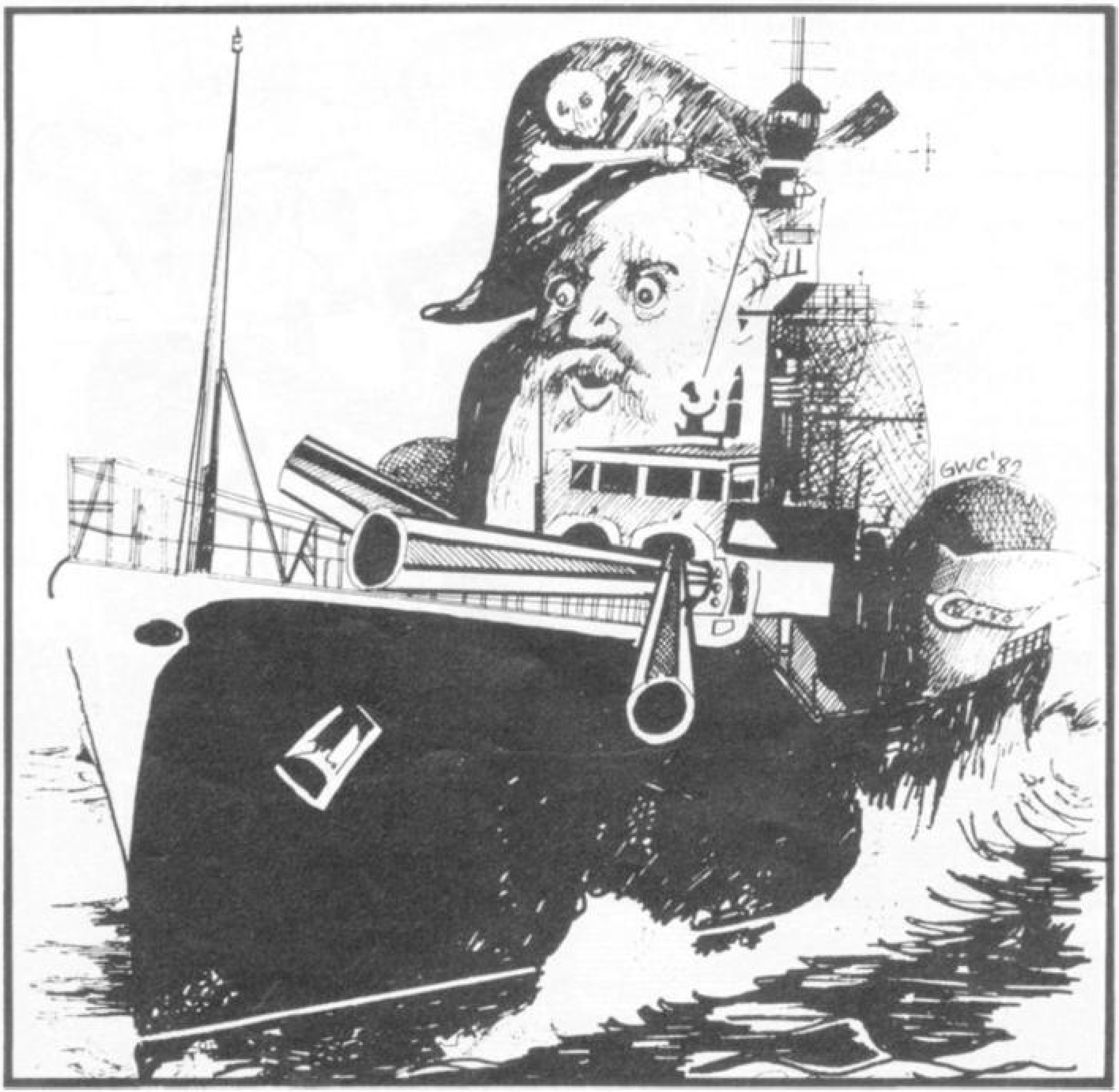
```
 10 LET B$ = "24231242312423124242423124231
    231231242669224231242312426692242312 4
    2312C"
 20 LET T$ = "GA75457CJGCAGGHGCHPGA75457CJGC
    ACGHLHGGG11357CJGCAGGHGCHP11357CJGCAG
    HLHG
 30 IF B$ = " " THEN STOP
 40 LET N = CODE(T$) - 28
 50 LET P = (CODE(B$) - 28)*10
 60 FOR I = 1 TO P
 70 LET D = 1**N
 80 NEXT I
 90 LET B$ = TL$(B$)
100 LET T$ = TL$(T$)
110 GOTO 30
```

Battleships

A boom, boom here, and a bang, flash there, and devastation covers the seas, in J Calderwood's version of the old pencil and paper game.

Here's your chance to play battleships against the ZX80. The programs asks you to enter a number up to 10, and a letter to J. These are to be entered one at a time. The clever computer will then show where your bomb landed. If you hit a ship, it will be shown in inverse otherwise it shows as an asterisk. You have to try and land three bombs on each ship.

```
  5 RANDOMISE
 10 PRINT "BATTLESHIPS"
 20 PRINT "************"
 21 PRINT "THERE ARE 8 SHIPS EACH 3 SQUARES LONG"
 22 PRINT "ENTER A NUMBER UP TO 10"
 24 PRINT "AND A LETTER UP TO J"
 30 DIM Z(100)
 40 FOR X=1 TO 8
 50 LET Y=RND(99)
 60 IF 10*(Y/10)=Y OR 10*((Y-1)/10)=Y-1 THEN
    GOTO 50
 65 IF Z(Y)=10 OR Z(Y-1)=10 OR Z(Y+1)=10 THEN
    GOTO 50
 70 LET Z(Y)=10
 80 LET Z(Y-1)=10
 90 LET Z(Y+1)=10
100 NEXT X
110 INPUT A
120 INPUT B$
130 LET B=CODE (B$)
140 LET B=(10*(B-38))+A
145 IF Z(B)=-1 OR Z(B)=9 THEN GOTO 110
150 LET Z(B)=Z(B)-1
155 CLS
160 PRINT "12345678910"
200 FOR T=1 TO 100
210 IF Z(T)=0 OR Z(T)=10 THEN PRINT " ";
220 IF Z(T)=-1 THEN PRINT "*";
230 IF Z(T)=9 THEN PRINT CHR$(148);
235 IF (T/10)*10=T THEN PRINT CHR$(37+T/10)
240 NEXT T
250 GOTO 110
```



Catch

Avoid being trapped by your sneaky, little ZX80.

The object of this game by I Soutar is to avoid getting nabbed by the computer. You move by entering "5" to move left, "6" to move down, "7" to move up or "8" to move right. When you've moved, the ZX80 will place a black square on one of the four sides of your position.

If you move onto the black square, all hell will break loose. Well, actually it won't, but the game will end, and the ZX80 will tell you how many moves you survived. Anything over 95 is very good, and you'll be awarded a gold star.

```
  1 RANDOMISE
  2 LET B=16
  4 LET A=6
  6 LET M=0
  8 LET P=0
 10 LET W=0
 12 LET D=0
 14 LET C=0
 20 LET Z=-1
 25 GOSUB 700
 30 FOR I=1 TO 9
 40 PRINT ,,,,
 50 NEXT I
 52 GOSUB 700
 60 GOSUB 500
 70 POKE W + 181,20
 80 LET Z=Z+1
 90 INPUT C
100 GOSUB 500
110 GOSUB 600
120 POKE M,0
130 IF C=6 AND A<10 OR C=7 AND A>1 THEN LET
    A=A-1*C+13
140 IF C=5 THEN LET B=B-1
150 IF C=8 THEN LET B=B+1
160 GOSUB 600
170 IF PEEK(M)=128 THEN GOTO 400
180 POKE M,20
190 LET D=RND(4)
195 IF A=10 AND D=4 OR A=1 AND D=3 THEN GOTO
    190
200 LET D=(D=1)-1*(D=2)+33*(D=3)+33*(D=4)
210 GOSUB 500
220 GOSUB 600
230 POKE M+D,128
240 GOTO 80
400 CLS
410 PRINT "YOU LASTED FOR";Z;"MOVES"
499 STOP
500 LET P=PEEK(16397)
510 IF P>127 THEN LET P=P-256
520 LET W=PEEK(16396)+P*256
530 RETURN
600 LET M=W+(A-1)*33+B
610 RETURN
700 FOR I=1 TO 32
710 PRINT CHR$(128);
720 NEXT I
730 RETURN
```

Shift

Winging across the seas from New Brunswick, Canada, is Joseph Ho's ZX80 game based on the slide games in which tiles with the numbers one to 15 have to be arranged in order.

When you press RUN, you'll see a 4 x 4 grid. On it are the numbers one to 15, and an inverse space. The numbers are scrambled, and you have to try and put them in order with the black square in the lower right-hand corner. The black square represents an empty space.

The numbers can be moved around by entering a number that is directly above, below, directly to the left or to the right of the space. A rather quaint error message — DO OVER — appears if you try to cheat. The number you've entered, if you haven't cheated, and all the numbers between it and the space will shift one place towards the space and the space will appear in the place of the number entered. Once you've solved the problem, the ZX80 will tell you how long it took you to do it. You'll find it fairly easy to convert this program to run on a ZX81, although it will not fit within 1K as does the ZX80 version.

```
  10 DIM A(15)
  20 FOR A=0 TO 15
  30 LET A(A)=A+1
  40 NEXT A
  50 FOR A=2 TO 15
  60 LET B=RND(A)-1
  70 LET C=A(A)
  80 LET A(A)=A(B)
  90 LET A(B)=C
 100 NEXT A
 110 FOR A=0 TO 1000
 140 CLS
 150 PRINT "   SHIFT"
 160 PRINT "   _____"
 170 FOR B=0 TO 3
 180 PRINT
 190 FOR C=0 TO 3
 200 LET D=B*4+C
 210 IF A(D)<10 THEN PRINT " ";
 220 IF NOT A(D)=16 THEN PRINT " ";A(D);
 230 IF A(D)=16 THEN PRINT "   ";CHR$(128);
 240 IF A(D)=16 THEN LET G=D
 250 NEXT C
 260 PRINT
 270 PRINT
 280 NEXT B
 290 LET E=G/4+1
 300 LET F=G-(G/4)*4+1
 310 FOR B=0 TO 15
 320 IF A(B)=B+1 THEN NEXT B
 325 IF B=16 THEN GOTO 2000
 330 PRINT
 340 PRINT "NUMBER?"
 350 INPUT B
 360 IF B<1 OR B>15 THEN GOTO 1000
 370 FOR C=0 TO 15
 380 IF NOT A(C)=B THEN NEXT C
 390 LET B=C/4+1
 400 LET D=C-(C/4)*4+1
 410 IF NOT B=E AND NOT D=F THEN GOTO 1000
 420 LET I=ABS(C-G)
 430 IF D=F THEN LET I=I/4
 440 FOR H=0 TO I-1
 450 IF D>F THEN LET A(G+H)=A(G+H+1)
 460 IF D<F THEN LET A(G-H)=A(G-H-1)
 470 IF D>E THEN LET A(G+H*4)=A((G+H*4)+4)
 480 IF D<E THEN LET A(G-H*4)=A((G-H*4)-4)
 490 NEXT H
 500 LET A(C)=16
 510 NEXT A
1000 PRINT "DO OVER"
1010 GOTO 350
2000 PRINT "YOU DID IT IN ";A;" MOVES"
2010 CLEAR
2020 PRINT "TYPE Y TO PLAY AGAIN"
2030 INPUT A$
2040 IF A$="Y" THEN RUN
2050 PRINT " HOPE YOU HAD FUN"
```

Down in the Depths

Phil Garratt takes the plunge with J K Greye's "3-D Monster Maze".

If I had to choose just one program to impress an audience with the capabilities of the ZX81, then J K Greye's "3-D Monster Maze" would be the one without a doubt. Written mainly in machine code, this 9½K game includes graphics which had me checking that the program was really being run on a ZX81, rather than one of the micros that measure their price, as well as their memory, in Ks.

The impressive features of this program start even before the game itself begins. A cleverly-drawn clown announces the instructions which scroll up the right-hand side of the screen. The clown even doffs his hat (although I must admit that at first I thought his head had fallen off, even these graphics have some limitations). Any program which can liven up the presentation of the instructions has to have a lot going for it.

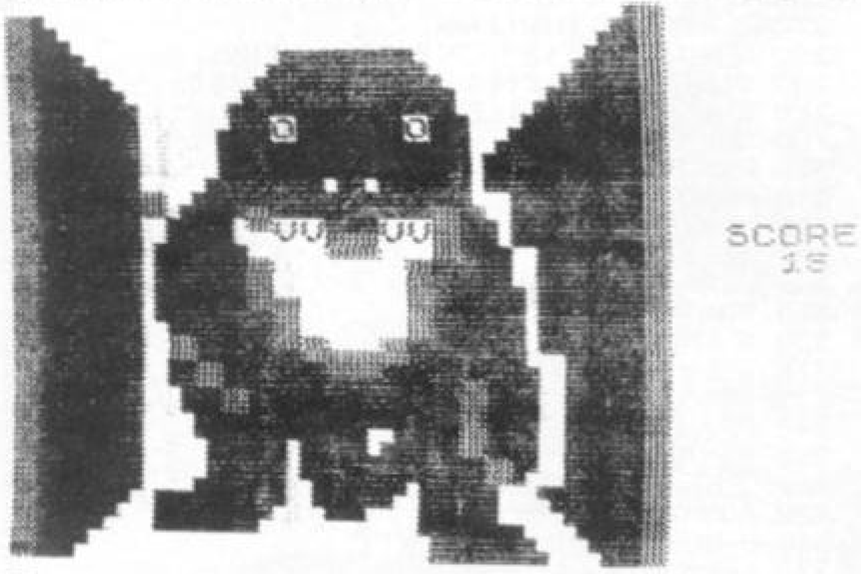
The ZX81 goes into FAST mode to set up the maze, and when it is complete, you are given an excellent 3-D representation. All walls are shaded and passages to the left and right can be seen clearly quite a way ahead. Keys 5, 7 and 8 are used to turn left, go forward and turn right, and the response is instantaneous. There is a real sense of motion through the maze if you keep your finger on 'forward'. Having played a couple of very slow 3-D mazes written in BASIC, this program would have been streets ahead . . . if it wasn't for Rex.

Rex is the unfriendly neighbourhood dinosaur who inhabits the maze. You only score points when he is chasing you . . . or when you find the exit . . . so this is no game for the faint-hearted. You are told if he is getting close, although if you're lucky (?) you'll see him lumbering towards you, getting bigger and bigger, the jaws opening wider and wider. Even after playing the game many times, the sight of him still makes me jump. Truly the stuff of nightmares. The graphics are incredible and I found I had to copy the screen to a printer to prove to myself that these amazing pictures were made up from the standard Sinclair character set.

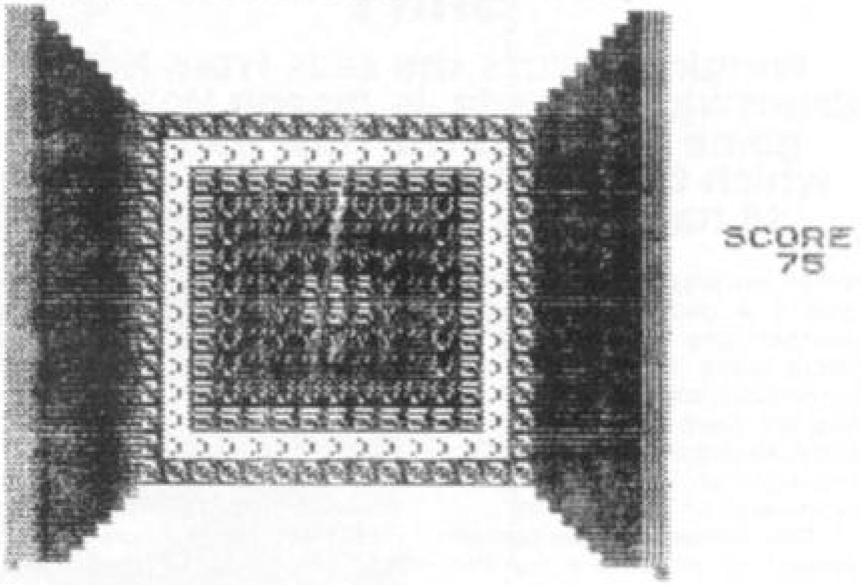
So the graphics are great, but how good is it as a game? I'm pleased to be able to say that it is very good indeed. There is on-screen scoring as you move around, plus bonus points if you find the exit, which involves yet more graphics effects. The exit is a moving kaleidoscope of letters and symbols which grows as you approach it. But beware of being hypnotised by it, as Rex is always close by to wake you up.

If you find the exit, a new maze is created and your score is carried forward. If you do happen to end up in Tyranno's tummy, you get the opportunity to start again with the same maze, so you can build up your chances of escape.

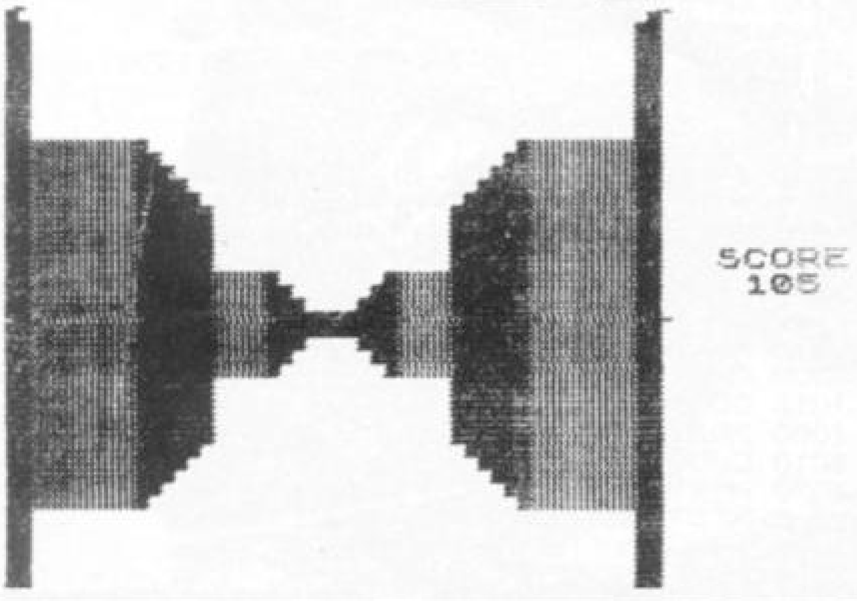
With such a professional standard of presentation maintained throughout the program, it deserves to do very well indeed. I certainly haven't seen anything like it before, but I hope it is just the start of 'realistic' graphics games for the ZX81. "3-D Monster Maze" is £5.95.



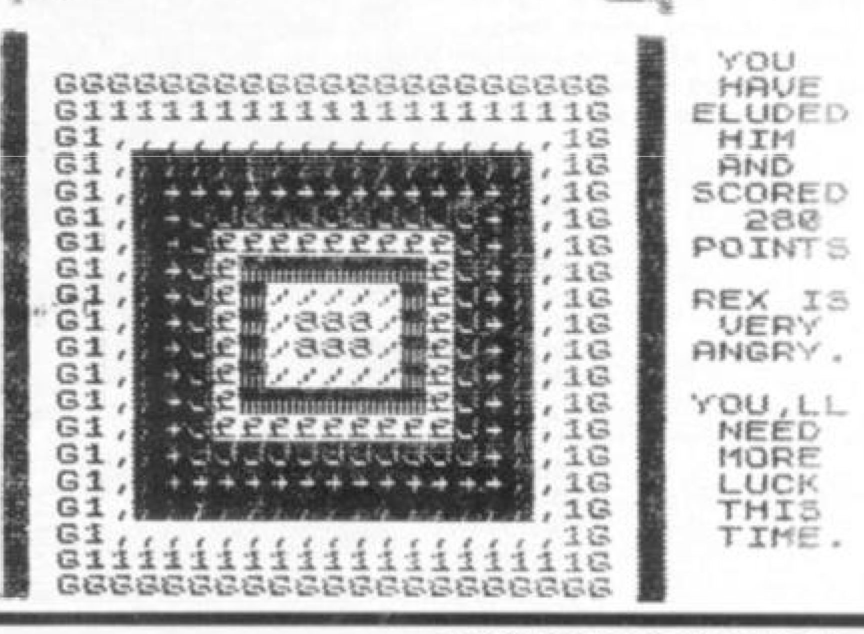
```
SCORE
 15
```



```
SCORE
 75
```



```
SCORE
 105
```



```
  YOU
  HAVE
 ELUDED
  HIM
  AND
 SCORED
   280
 POINTS

 REX IS
  VERY
 ANGRY.

 YOU,LL
  NEED
  MORE
  LUCK
  THIS
  TIME.
```

Orwin strikes again

Graham Charlton spends an afternoon with Michael Orwin's Cassette Two and finds that although the graphics for all the programs are good, the games themselves are a mixed bag.

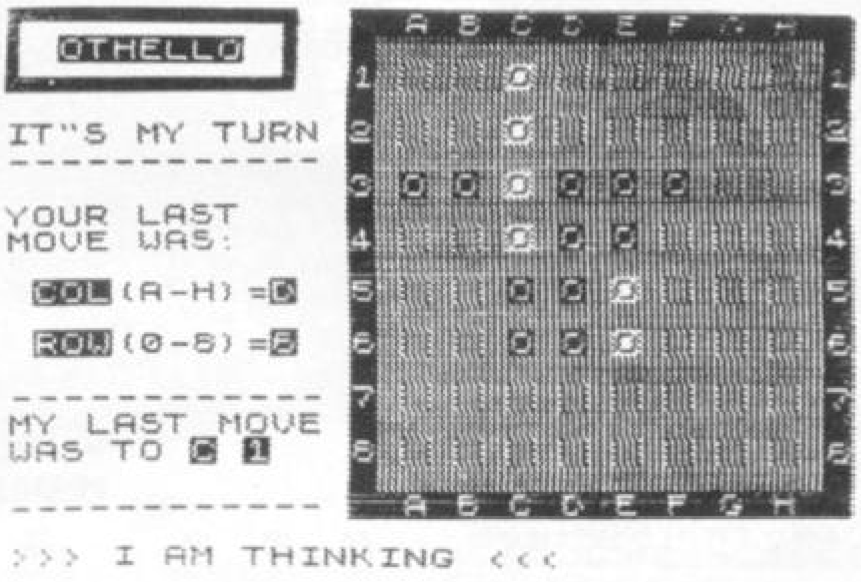
Cassette Two contains 10 games in BASIC, dubbed on each side of the cassette in a different order. My first impression of Cassette Two was one of slight disappointment after reading highly enthusiastic reviews of Cassette One. However, after going through the programs for a second time, and taking a closer look at them, I feel there is much to be said for Cassette Two. There is a good selection of games, and the clear graphics show what can be done when you have 16K to play with. All programs need more than 1K.

The games on the cassette are by P.Canter and C.Panayi, except for Laser Bases and Rectangles by Mr Orwin himself, and P.Rushton's Roulette. I found no trouble loading the programs, which all start automatically. The instructions are included in the programs for all but two of them, and rules for these are supplied in an accompanying leaflet.

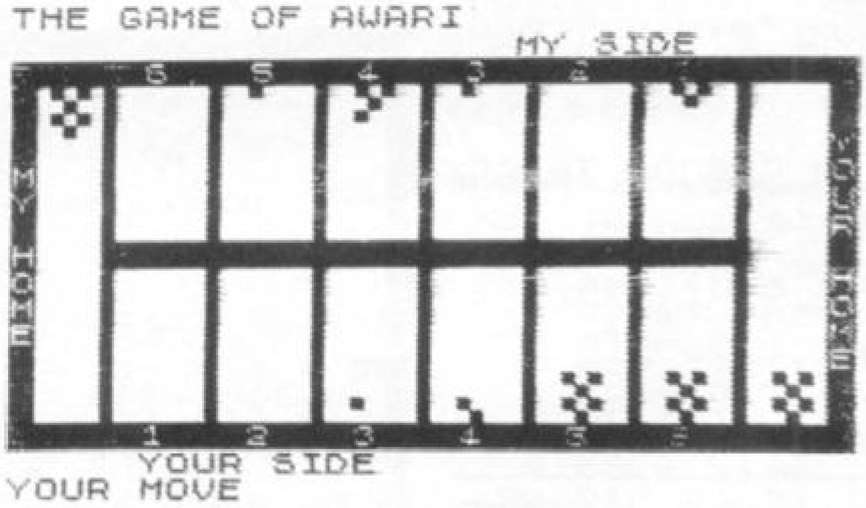
I'll go through the games one by one.

OTHELLO: The programming in this case is unwise (as in most of the other games on the cassette) with the rules at the beginning of the listing, rather than the end, thus slowing down every GOTO and GOSUB access. You get four options at the beginning of the game: "Do I play my best?", "Fast or slow?", "White or black?" and "Do you want to go first?". The square chosen by the ZX81 is flashed off and on a few times before it makes its move, to show clearly where it is playing. The program points out how many pieces the ZX81 is capturing. There are polite comments on attempted illegal moves and a comment on the game when it is finished. The program display is great, but the program does not play particularly well. There are much stronger (and more expensive) Othello programs on the market.

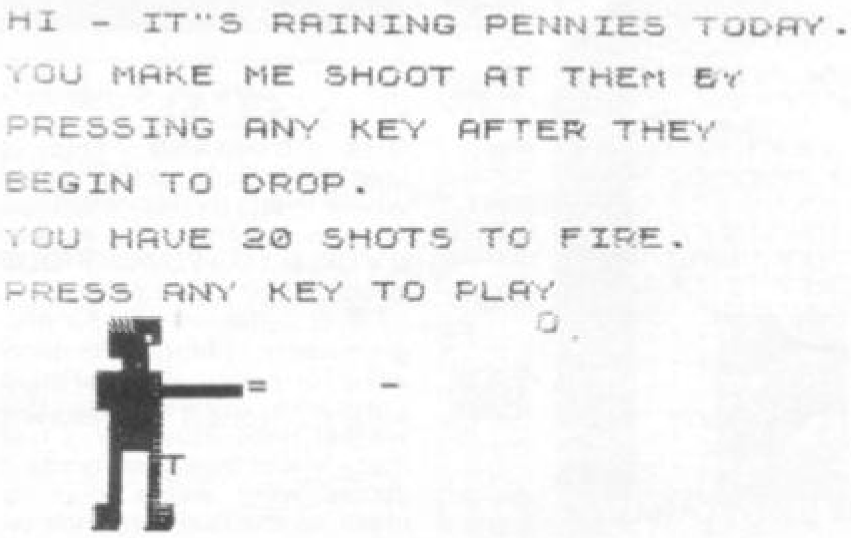
AWARI: This is a game of strategy, again with a clear, full-screen display. The ZX81 plays quite well, but there is no option on the level of play, and the whole game is over fairly quickly. This program is fairly easy to beat. It provides a good game if you don't think too hard.



```
OTHELLO

IT"S MY TURN
------------

YOUR LAST
MOVE WAS:

 COL (A-H) =D

 ROW (0-8) =5

------------
MY LAST MOVE
WAS TO C 1

------------

>>> I AM THINKING <<<
```



```
THE GAME OF AWARI
                    MY SIDE
     6    5    4    3    2    1

MY HOME                       YOUR HOME

     1    2    3    4    5    6
   YOUR SIDE
YOUR MOVE
```



```
HI - IT"S RAINING PENNIES TODAY.
YOU MAKE ME SHOOT AT THEM BY
PRESSING ANY KEY AFTER THEY
BEGIN TO DROP.
YOU HAVE 20 SHOTS TO FIRE.
PRESS ANY KEY TO PLAY
```

LASER BASES: You and the ZX81 each have 10 laser bases. The object — needless to say — is to destroy your opponent's bases. On each turn you have the option to fire, shield or do nothing at all: You are supposed to make your decisions at the same time as the ZX81, when it displays its decision and then asks for yours. After this it displays what has happened to each base and the new positions. I found it hard to resist the temptation of cheating by changing my decisions after seeing what the ZX81 had done. Perhaps the game could be written so that this cheating was not possible.

WORD MASTERMIND: This is a nice variation on the numerical Mastermind games. It has a vocabulary of over 100 words which can be changed fairly easily. The vocabulary is not in the visible program, but you won't find it very hard to get the ZX81 to reveal what words it is holding. This is a very good program, and it really got me thinking.

RECTANGLES: Horrors! The instructions for this game come on a separate sheet. They read like a script for the Monty Python Show. I haven't a clue how to play the game, despite repeated attempts, and I can't work it out from running the program. Perhaps an example in the rules would have helped a little. In its favour, this program has a full-screen display, and a little machine code is included to speed it up.

CRASH: You use the cursor keys to steer your chequered worm to try and trap a black worm, controlled by the zany ZX81, to ensure that it cannot move without hitting you or itself. It is trying to do the same thing to you. Very addictive, although a touch on the slow side. However, if it was any faster, I'm sure I'd lose every time.

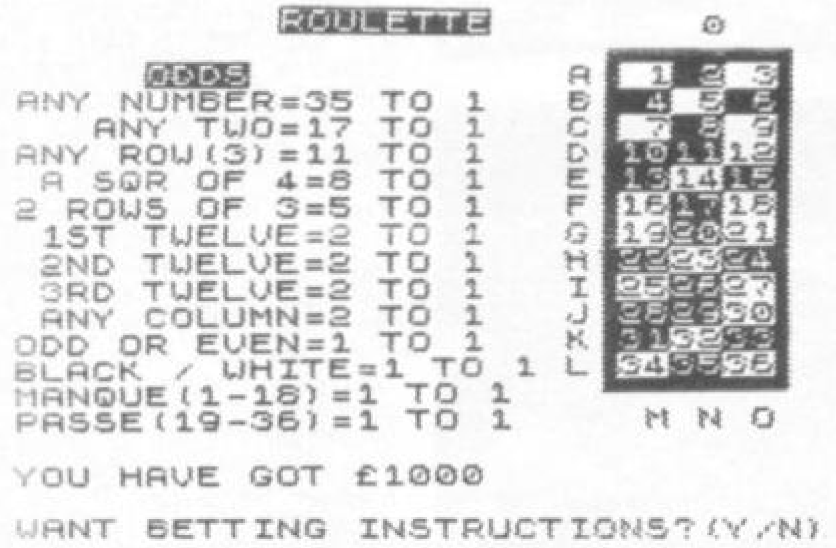
ROULETTE: This is a great way to try out your betting system before going to Monte Carlo. There are 10 different ways of betting, and the system to enter your bets is easy to understand. Although the game seems to simulate the rules of 'proper' roulette properly, I found that a system I invented enabled me to amass a vast fortune. I'm tempted now to try the system in real life. When the 'wheel' is spun in this program, I think it would be good for the squares on the board to flash or something, rather than just overprint a series of random numbers below the board.

PONTOON: Ho hum, you might think. But you'd be wrong. This program — which features an excellent display of the cards — shows just how much can be done using good graphics when you have 16K to play with.

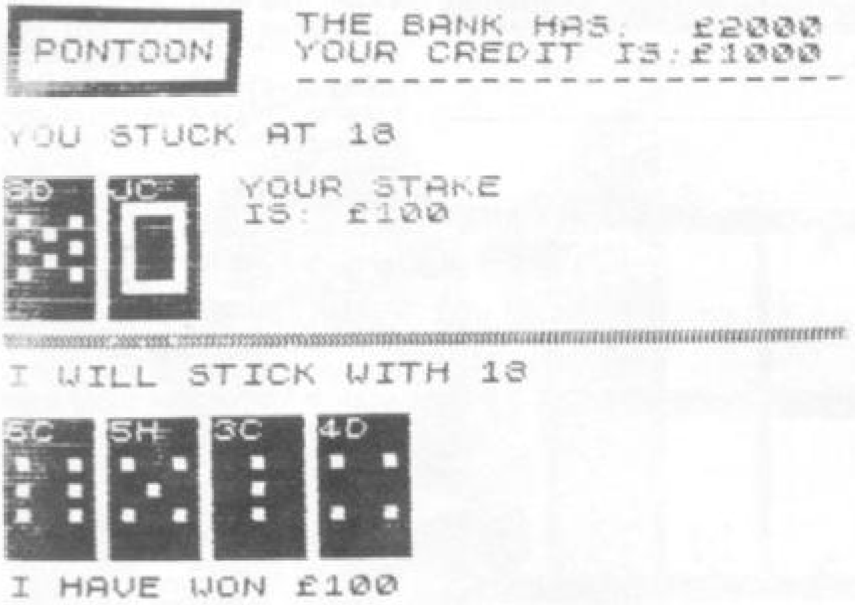
PENNY SHOOT: In this program, you make a robot shoot at pennies which are raining down from the sky. The introduction to the program was a nice touch, more fascinating than the game itself. A little figure appears to read the instructions as they appear, then turns and shoots a sample penny. This program is not particularly challenging.

GUN COMMAND: This is another moving action game where you try to intercept missiles traversing the screen. It's OK, but like PENNY SHOOT could be a bit more challenging.

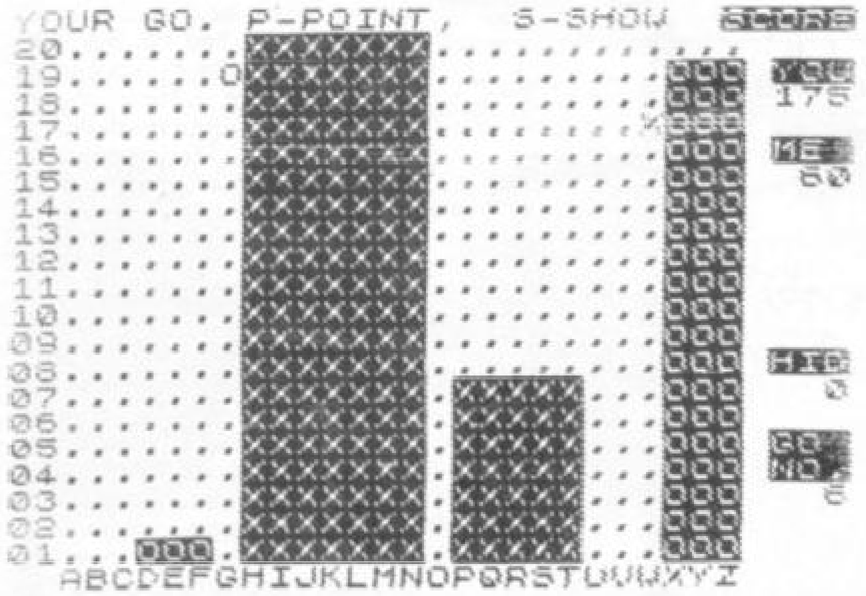
Overall, for the £5 this program costs, it represents good value for money, with the good programs outweighing the weaker ones. However, I feel that a few of the lower standard games were added more to make up the numbers than be really good additions to the package. The programs all show how well the somewhat limited graphics of the ZX81 can be made to perform, although they also indicate that there is a temptation to 'dress up' fairly shallow programs in fancy clothes to disguise them. Despite all these comments, it represents a good selection of games. You're sure to find one or two favourites which you'll play over and over again. Cassette Two is £5 from Michael Orwin.



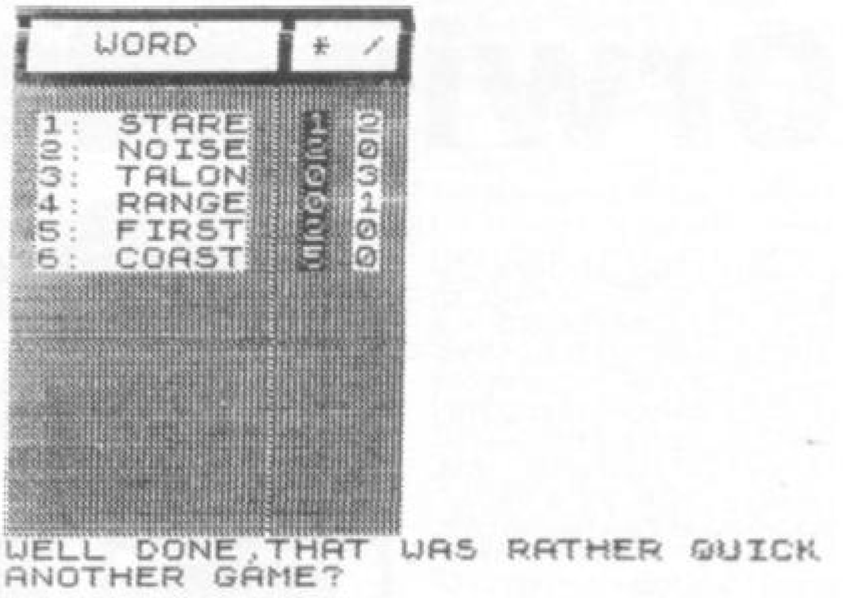
```
                ROULETTE              0

       ODDS                        A  1  2  3
ANY NUMBER=35 TO 1                 B  4  5  6
   ANY TWO=17 TO 1                 C  7  8  9
ANY ROW(3)=11 TO 1                 D 10 11 12
 A SQR OF 4=8 TO 1                 E 13 14 15
2 ROWS OF 3=5 TO 1                 F 16 17 18
 1ST TWELVE=2 TO 1                 G 19 20 21
 2ND TWELVE=2 TO 1                 H 22 23 24
 3RD TWELVE=2 TO 1                 I 25 26 27
 ANY COLUMN=2 TO 1                 J 28 29 30
ODD OR EVEN=1 TO 1                 K 31 32 33
BLACK / WHITE=1 TO 1               L 34 35 36
MANQUE(1-18)=1 TO 1
PASSE(19-36)=1 TO 1                  M  N  O

YOU HAVE GOT £1000

WANT BETTING INSTRUCTIONS?(Y/N)
```



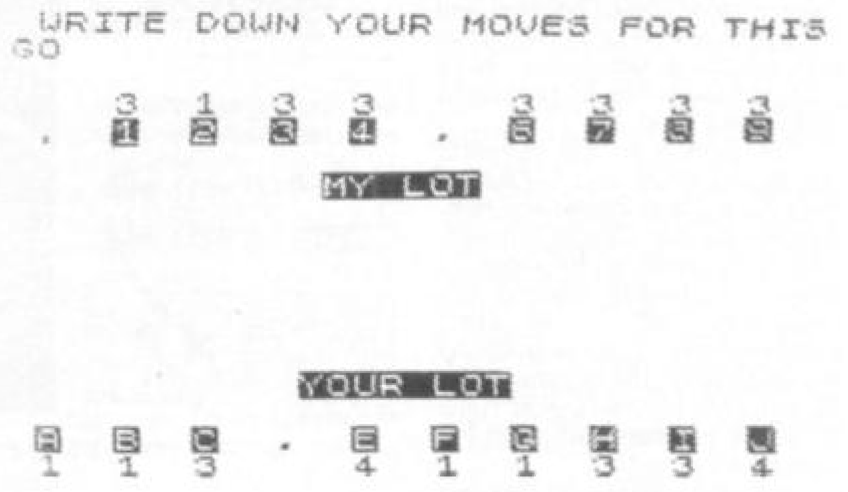
```
PONTOON     THE BANK HAS:    £2000
            YOUR CREDIT IS:£1000
            ----------------------
YOU STUCK AT 18

            YOUR STAKE
            IS: £100

I WILL STICK WITH 18

I HAVE WON £100
```



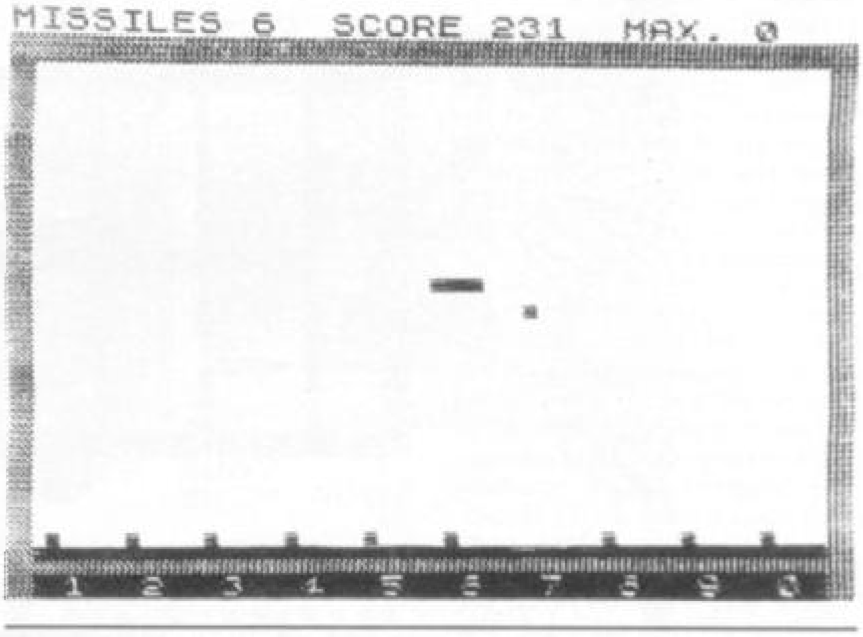
```
YOUR GO. P-POINT, S-SHOW
```



```
WORD          * /

1: STARE    1 2
2: NOISE    2 0
3: TALON    0 3
4: RANGE    0 1
5: FIRST    2 0
6: COAST    5 0

WELL DONE,THAT WAS RATHER QUICK
ANOTHER GAME?
```



```
 WRITE DOWN YOUR MOVES FOR THIS
GO

   3  1  3  3     3  3  3  3
.  1  2  3  4  .  6  7  8  9

            MY LOT




           YOUR LOT

A  B  C  .  E  F  G  H  I  J
1  1  3     4  1  1  3  3  4
```



```
MISSILES 6  SCORE 231  MAX. 0
```

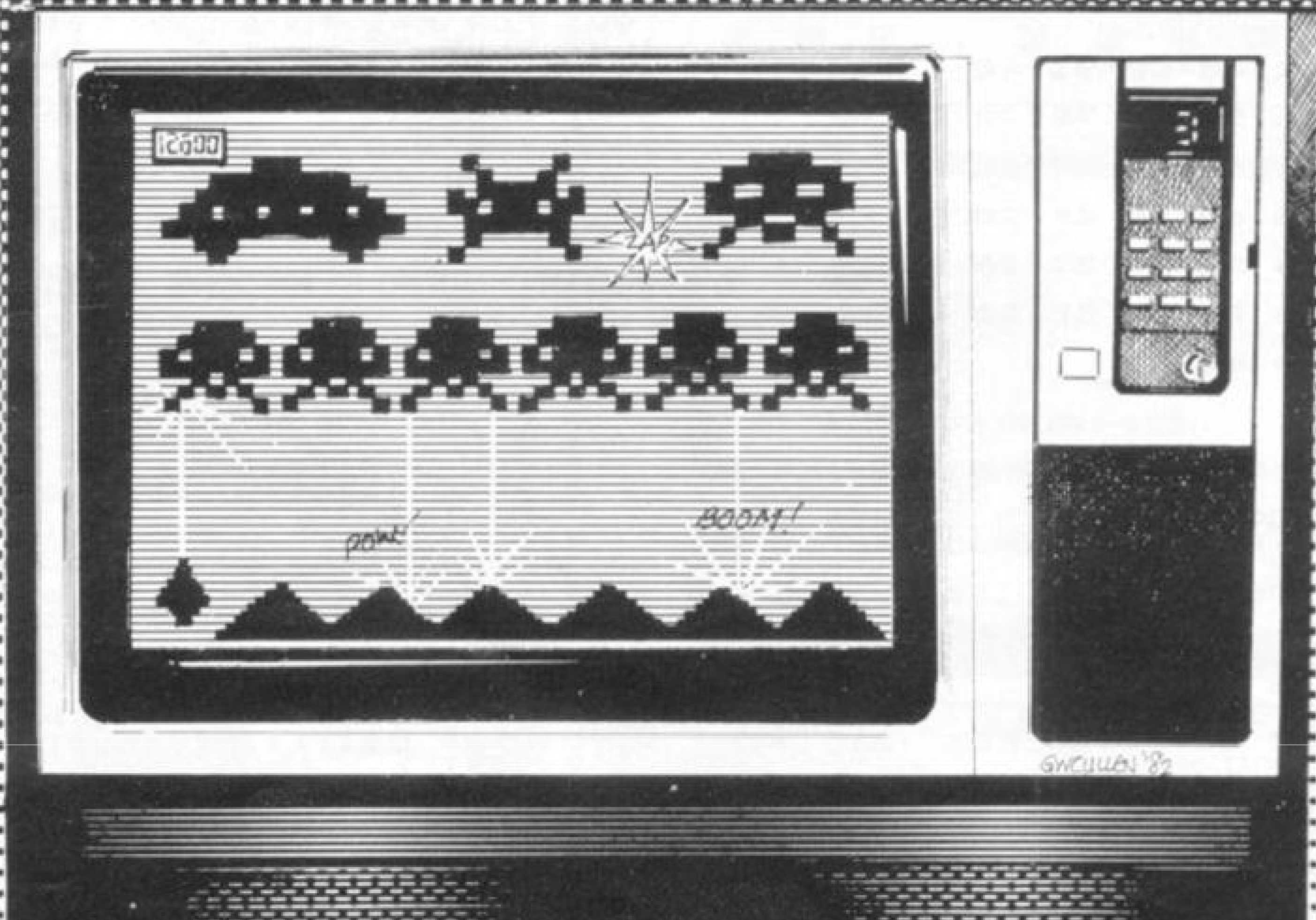
LARGE CHARACTERS

David Kelsall of St. Albans has provided programs to print wide and double height characters on the ZX printer. The programs can be combined to produce 16 large letters per line.

```
REM STATEMENT IN HEX

21 7C 40 CD 91 40 41 CD

91 40 ED 43 7B 40 C9 0E

FF ED 6F CB 47 20 04 CB

81 CB 89 CB 4F 20 04 CB

91 CB 99 CB 57 20 04 CB

A1 CB A9 CB 5F 20 04 CB

B1 CB B9 C9
```

```
WIDE    CHARACTERS

   1 REM 5?RNDLN █RNDINKEY$LN █R
ND GOSUB ??RNDTAN : COPY GOSUB ?
ACS ?4▪ACS █ACS █ACS ?4▪ACS █ACS
█ACS ?4▪ACS █ACS █ACS ?4▪ACS █A
CS █TAN
   2 IF PEEK 16389=124 THEN GOTO
 5
   3 PRINT "RESERVE MEM (POKE 16
389,124)"
   4 STOP
   5 FOR I=0 TO 112
   6 POKE 31744+I,PEEK (2161+I)
   7 NEXT I
   8 POKE 31800,63
   9 POKE 31857,201
 100 LET B$="WIDE  CHARACTERS"
 110 DIM A$(32,8)
 120 FOR X=2 TO 32 STEP 2
 130 FOR Y=1 TO 8
 140 LET P=PEEK (7679+8*CODE B$(
X/2)+Y)
 150 POKE 16508,P
 160 RAND USR 16514
 170 LET A$(X-1,Y)=CHR$ PEEK 165
08
 180 LET A$(X,Y)=CHR$ PEEK 16507
 190 NEXT Y
 200 NEXT X
 210 REM
 220 REM COPYRIGHT D.J.KELSALL
 230 REM JANUARY 1982
 240 REM
9988 REM PRINT A$
9990 FOR J=1 TO 32
9991 FOR K=1 TO 8
9992 POKE 32255+K+8*(J-1),CODE A
$(J,K)
9993 NEXT K
9994 NEXT J
9995 FOR H=0 TO 31
9996 POKE 16444+H,H
9997 NEXT H
9998 LET HPRINT=USR 31744
```

```
*PRINT DOUBLE HEIGHT CHARACTERS*

   1 IF PEEK 16388+256*PEEK 1638
9=31744 THEN GOTO 5
   2 REM POKE 16389,124
   3 PRINT "MEMORY NOT RESERVED"
   4 STOP
   5 FOR I=0 TO 112
   6 POKE 31744+I,PEEK (2161+I)
   7 NEXT I
   8 POKE 31800,63
   9 POKE 31857,201
  90 DIM A$(32,16)
 100 LET B$="*PRINT DOUBLE HEIGH
T CHARACTERS*"
 110 FOR Y=1 TO 32
 120 FOR X=0 TO 15
 130 LET A$(Y,X+1)=CHR$ PEEK (76
80+8*CODE B$(Y)+INT (X/2))
 140 NEXT X
 150 NEXT Y
9988 REM PRINT A$, 8 LINES AT A
TIME
9989 FOR I=0 TO 10 STEP 8
9990 FOR J=1 TO 32
9991 FOR K=1 TO 8
9992 POKE 32255+K+8*(J-1),CODE A
$(J,K+I)
9993 NEXT K
9994 NEXT J
9995 FOR H=0 TO 31
9996 POKE 16444+H,H
9997 NEXT H
9998 LET HPRINT=USR 31744
9999 NEXT I
```


12500
ZAP
POW!
BOOM!

Getting into the Movies

What do you do when you want aliens and asteriods to burn around your TV screen? The PRINT AT on the ZX81 gives you one solution.

You make an object move — or appear to move — on the TV screen by printing it in one position, holding the display for a moment, then unprinting the old position just as you reprint in a new position.

The PRINT AT function on the ZX81, although it is slow, makes it easy to position an object just where you want it. For our first simple program, enter PROGRAM ONE and press RUN to see it in action. What you'll see, all being well, is a black square move down the screen diagonally. Note that you can 'chain' PRINT AT commands as in line 30.

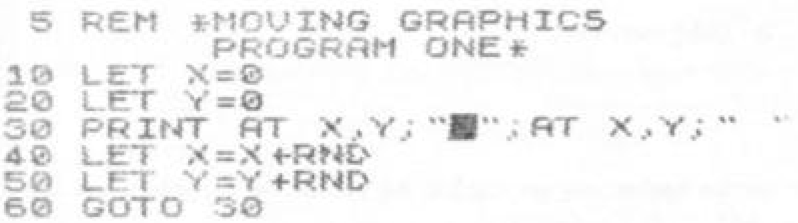
This program shows the simplest kind of moving graphics programming, and although far from satisfactory in terms of the result displayed on the screen, at least it should give you a clear idea of how to produce one kind of effect. Its main disadvantage is that the screen is blank while the computer is working out the new PRINT AT position for the black blob. As you can see from running this program, the RND function — when running the computer in SLOW — is particularly slow. As you begin to elaborate the program you're working on, and you want the computer to do more and more, you'll find this blank between subsequent PRINT ATs of the blob becomes intolerable. Try, for example, adding 35 LET Z = X**Y and see how slow the program becomes. This can be overcome to some extent by assigning two other variables to the PRINT AT position which hold it for unprinting while the computer is working out new positions.

Change the program so it reads as the listing for program two. Run this, and you'll see how much more satisfactory it is. If you want it to run forever, add 57 IF X greater than 19 THEN RUN.

Now moving a blob down the screen in an irregular slide is not of much use to anyone. Enter and run program number three, which makes a ball bounce around the screen. Although this does not use a second set of variables to unprint, it is fairly satisfactory. You'll see that lines 70 and 80 check each time through the program to ensure that the ball has not 'hit the sides', and if it has done so, ensures that the variables which determine the next position of the ball (C and D) are changed.

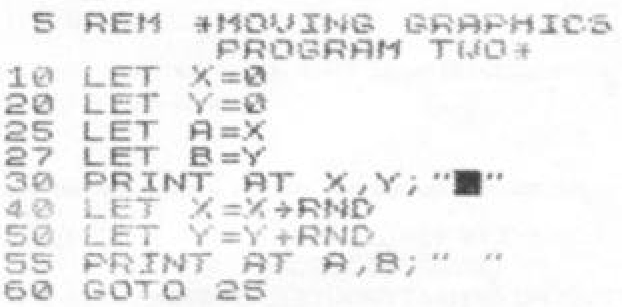
Program four does use new variables (E and F) to hold the old position while the new one is being worked out, and these are used not as an 'unprint' but rather to print a trailing series of full stops, as you can see in diagram one. Run these programs a few times, and examine the listings, and you're sure to be able to work out ways to make them more effective.

A second way of approaching the moving graphics problem is to use SCROLL to move the screen up a line, before reprinting what you need on the line it has just left. Enter and run program five, HAILSTORM, and you'll see how it works in practice. In this program, you are the black block, PRINT ATted in line 30. You control the position of the blob by using INKEY$, pressing on keys "8" (to move the blob right) and "5" (to move it left). The whole of line 80 ensures that the blob will not move off the screen, and saves valuable memory by placing the interpretation of both INKEY$ and checking the screen limits on either side, within one line. Line 990 prints



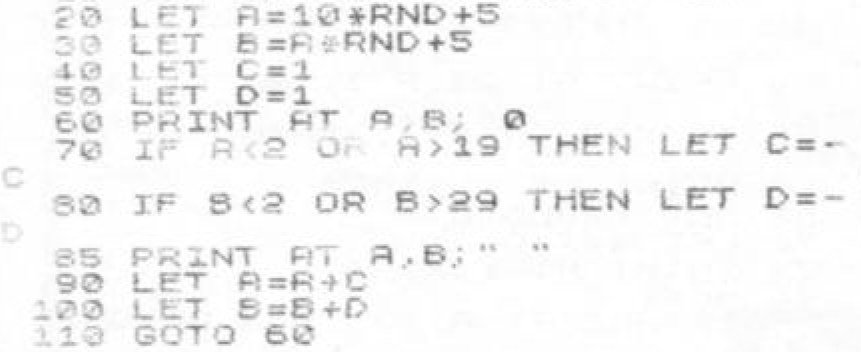
```
  5 REM *MOVING GRAPHICS
          PROGRAM ONE*
 10 LET X=0
 20 LET Y=0
 30 PRINT AT X,Y;"■";AT X,Y;" "
 40 LET X=X+RND
 50 LET Y=Y+RND
 60 GOTO 30
```

● Program one



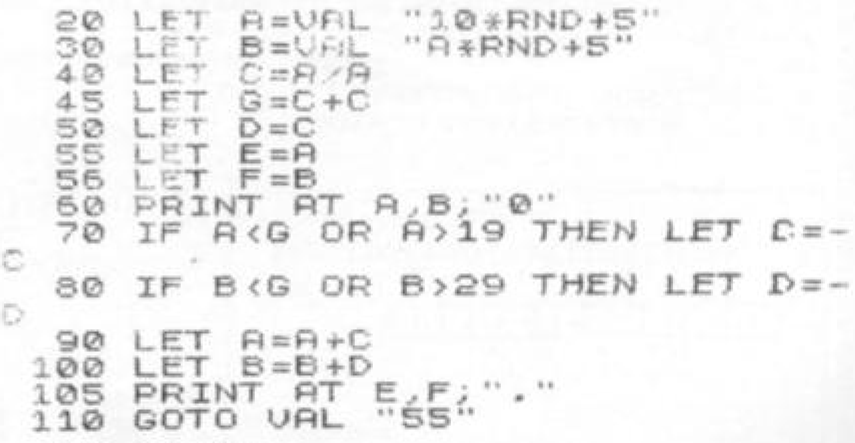
```
  5 REM *MOVING GRAPHICS
          PROGRAM TWO*
 10 LET X=0
 20 LET Y=0
 25 LET A=X
 27 LET B=Y
 30 PRINT AT X,Y;"■"
 40 LET X=X+RND
 50 LET Y=Y+RND
 55 PRINT AT A,B;" "
 60 GOTO 25
```

● Program two



```
 10 REM *BOUNCING BALL - 1*
 20 LET A=10*RND+5
 30 LET B=A*RND+5
 40 LET C=1
 50 LET D=1
 60 PRINT AT A,B; 0
 70 IF A<2 OR A>19 THEN LET C=-
C
 80 IF B<2 OR B>29 THEN LET D=-
D
 85 PRINT AT A,B;" "
 90 LET A=A+C
100 LET B=B+D
110 GOTO 60
```

● Program three



```
 20 LET A=VAL "10*RND+5"
 30 LET B=VAL "A*RND+5"
 40 LET C=A/A
 45 LET G=C+C
 50 LET D=C
 55 LET E=A
 56 LET F=B
 60 PRINT AT A,B;"0"
 70 IF A<G OR A>19 THEN LET C=-
C
 80 IF B<G OR B>29 THEN LET D=-
D
 90 LET A=A+C
100 LET B=B+D
105 PRINT AT E,F;"."
110 GOTO VAL "55"
```

● Program four

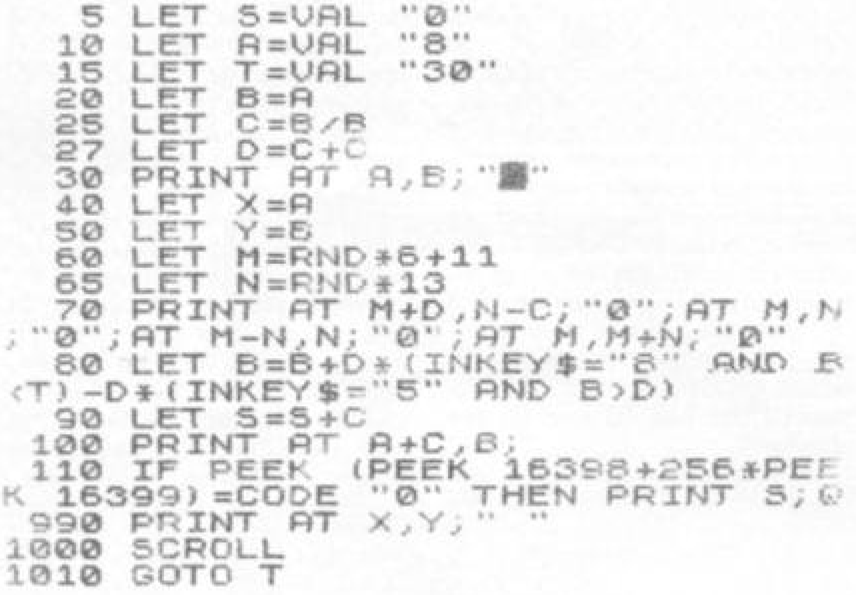
the blank to remove the black blob. This blank is printed just before the screen SCROLLs upward, to move all the zeroes upward. Your aim in the game — needless to say — is to avoid the hailstones, as you can see from diagram two. You'll remember that at the start of this article we mentioned that the RND function was fairly slow. To minimise the delay, we've used a single pair of random numbers (generated in lines 60 and 65) four times in line 70.

Lines 100 and 110 are very interesting, and are crucial for producing a worthwhile program using the SCROLL facility. Line 100 moves the PRINT AT position to A+C, B which is a line down from where the black blob is at present. Using line 110, the computer then looks up this position in the display file, to see if anything is there. If it finds the code for "0" in that position — that is one line below the black blob — it knows the black blob is going to hit a hailstone, and prints out your score, the value of the variable S. The Q at the end of this line causes the ZX81 to stop with a 2 error code, meaning an unassigned variable has been encountered. This is 'cheaper', in memory terms, than running the line twice, with THEN STOP at the end of the second line, or of saying IF. . .THEN GOTO 1020, where line 1020 reads PRINT S. Other memory-saving devices you can glean from this listing include the use of VAL a string, instead of using a number. Believe it or not, LET P = 1 uses up more memory, because of the way the ZX81 memory works, than LET P = VAL "1", and CODE "0" is cheaper than the number 28.

In our final program, SQUASH (program number six), a ball (which looks remarkably like an asterisk) is bounced off a black blob at the bottom of the screen, which you move right and left by using the "Q" and "P" keys. If you manage to 'bounce it', your score (shown in the top right hand corner of the screen) increases by one, and the game continues. Miss the ball, and the fun is over. Enter and run this program to see how it works. Note that because a random number does not have to be generated between successive reprints off the ball, the time the screen is blank is very short (only while the computer processes lines 65 and 70), so a second pair of variables to 'hold the ball' are not needed.

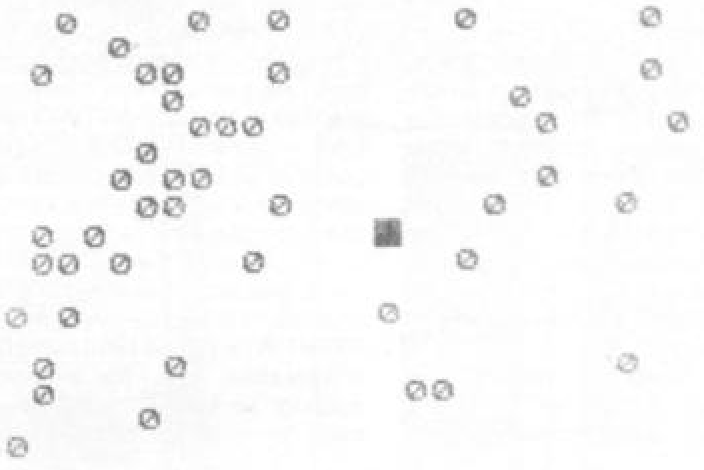
Line 5 initialises the vertical position of the ball, and line 10 sets up the player (black blob) starting position. Line 15 starts the count at −1, which quickly becomes zero at line 30, before the first score is printed. Line 20 initialises the ball's horizontal position, and 25 initialises the horizontal direction of the ball. Minus one sends it left, zero sends it straight up and plus one sends it right. Line 30 increments. the score count. Line 85 checks to see if the ball and the bat are fairly close, and if they are, allows the game to continue.

There are a number of simple ideas presented in this article for you to implement in your own moving graphics games on the ZX81. We'd be pleased to see your favourite program which you've developed after reading this article.

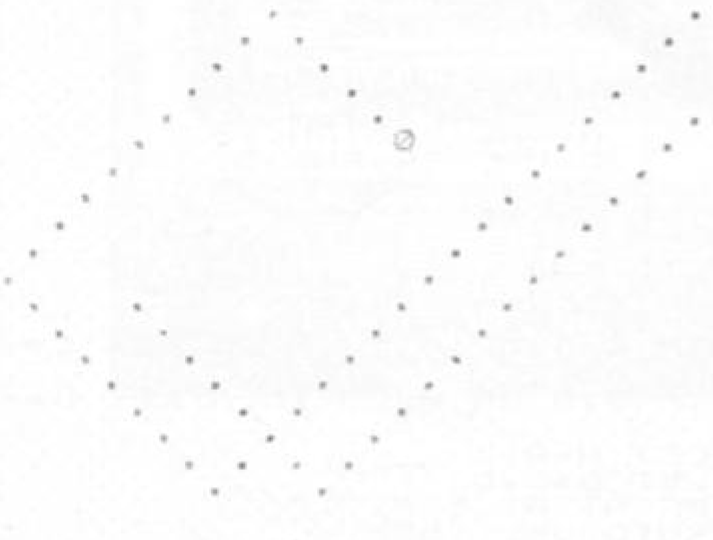


```
   5 LET S=VAL "0"
  10 LET A=VAL "8"
  15 LET T=VAL "30"
  20 LET B=A
  25 LET C=B/B
  27 LET D=C+C
  30 PRINT AT A,B;"■"
  40 LET X=A
  50 LET Y=B
  60 LET M=RND*6+11
  65 LET N=RND*13
  70 PRINT AT M+D,N-C;"0";AT M,N
;"0";AT M-N,N;"0";AT M,M+N;"0"
  80 LET B=B+D*(INKEY$="8" AND B
<T)-D*(INKEY$="5" AND B>D)
  90 LET S=S+C
 100 PRINT AT A+C,B;
 110 IF PEEK (PEEK 16398+256*PEE
K 16399)=CODE "0" THEN PRINT S;Q
 990 PRINT AT X,Y;" "
1000 SCROLL
1010 GOTO T
```

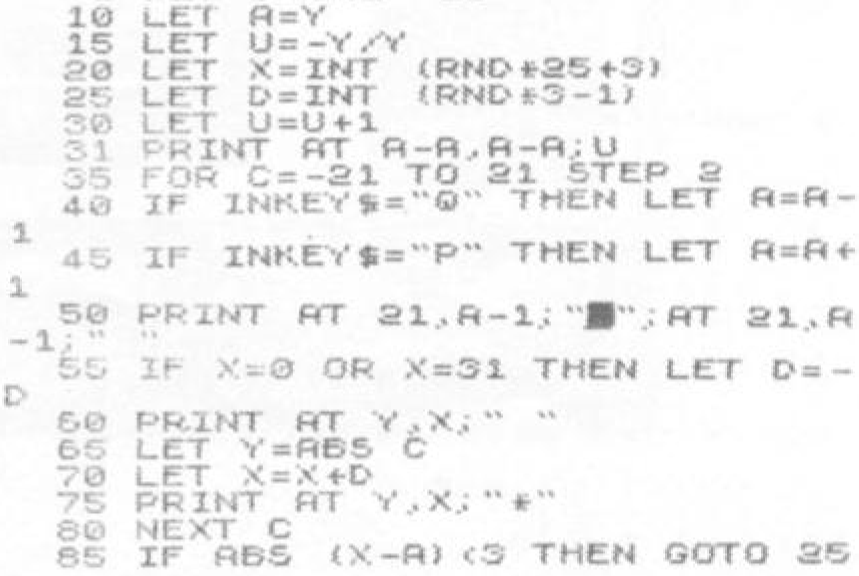
● Program five



● Diagram two



● Diagram one



```
   5 LET Y=VAL "16"
  10 LET A=Y
  15 LET U=-Y/Y
  20 LET X=INT (RND*25+3)
  25 LET D=INT (RND*3-1)
  30 LET U=U+1
  31 PRINT AT A-A,A-A;U
  35 FOR C=-21 TO 21 STEP 2
  40 IF INKEY$="Q" THEN LET A=A-
1
  45 IF INKEY$="P" THEN LET A=A+
1
  50 PRINT AT 21,A-1;"■";AT 21,A
-1;" "
  55 IF X=0 OR X=31 THEN LET D=-
D
  60 PRINT AT Y,X;" "
  65 LET Y=ABS C
  70 LET X=X+D
  75 PRINT AT Y,X;"*"
  80 NEXT C
  85 IF ABS (X-A)<3 THEN GOTO 25
```

● Program six

Specialist Books

Choosing programs for microcomputers
1980 J E Lane *£9.00*
A5 138pp P ISBN 0 85012 255 4
Looks at application packages for micros describing what they are, the benefits they offer and their use on microcomputers. Guidelines for obtaining packages and for identifying the best product are given.

Elements of BASIC
1979 R Lewis and B H Blakeley *£9.00*
A5 200pp P ISBN 0 85012 118 3
Introduces the BASIC language, covering the mathematical, non-numeric and data processing facilities. Generally machine independent with supplements to show the effect of a number of different implementations.

Graphics on microcomputers
1981 J E Lane *£4.00*
A5 44pp P ISBN 0 85012 333 X
Explores the type of graphics becoming increasingly available in low cost systems. Illustrates the facilities available and takes a closer look at graphics picture building techniques.

Information handling by microcomputers
1981 J E Lane *£4.00*
A5 60pp P ISBN 0 85012 334 8
Examines the field of information handling on microprocessors across the whole spectrum of micro applications. Aims to promote an awareness of current practices and trends.

NCC
The National Computing Centre

Introducing computer programming
1979 Reprint W G Collin *£11.50*
A5 364pp P ISBN 0 85012 210 4
A machine language independent textbook for the beginner, providing all the necessary basic information needed by someone starting on a computer programming career.

Introducing data processing
1980 NCC *£6.50*
A5 237pp P ISBN 0 85012 245 7
Covers the requirements of syllabi for introductory courses. Provides a comprehensive and accessible introduction to data processing. Assumes no previous knowledge of the subject.

Introducing microprocessors
1979 G L Simons *£9.00*
A5 177pp P ISBN 0 85012 209 0
Gives a profile of the microprocessor scene paying attention to typical application areas together with hardware and software information.

Introducing word processing
1981 G L Simons *£8.50*
A5 180pp P ISBN 0 85012 320 8
Describes the main characteristics of word processing and discusses its advantages over conventional typewriting. Communication, maintenance, security and costs are considered.

Operating systems for microcomputers
1981 J E Lane *£3.50*
A5 77pp P ISBN 0 85012 277 5
Establishes the requirements of operating systems for microcomputers in both commercial and industrial application areas and examines the facilities provided in a number of current products.

Student notes on NCC DP documentation standards
1978 NCC *£5.50*
A5 100pp P ISBN 0 85012 339 9
A subset of the full documentation standards for use by students on courses where NCC standards are part of the syllabus.

The robots are coming
1974 F H George & J D Humphries (eds)
£10.00
A5 188pp P ISBN 0 85012 114 0
Gives a general background to current developments in artificial intelligence research and looks at where these developments could be leading.

Using computers — a manager's guide
1980 M Peltu *£7.50*
A5 180pp P ISBN 0 85012 241 4
Intended to help managers implement computer systems effectively in an organisation. Provides an introduction for user management covering the topics of planning and control plus human factors.

Working with computers: a guide to jobs and careers
1975 *£2.50*
A5 86pp P ISBN 0 85012 126 4
A general introduction to computing as a career for school leavers. Covers how a computer is used, what types of job exist and how to train for them.

NCC
The National Computing Centre

We are now able to offer, in addition to our usual selection of books on computers, a number of specialist titles from the National Computing Centre.

Rather than taking their entire list of some 110 titles, we have selected those most relevant to the microcomputer market and these are listed with their precis.

Ordering couldn't be simpler, just tick the boxes in the form below, enclose a cheque or postal order to the total amount (or make use of the Barclaycard and Access facility) and send it all off to:

SPECIALIST BOOKS,
ZX COMPUTING
145 CHARING CROSS ROAD,
LONDON WC2H 0EE.

If you are using your credit card to order please don't send it, just fill in the number and sign on the dotted line.
Please allow 28 days for delivery of your books.

□ CHOOSING PROGRAMS FOR MICROCOMPUTERS *£9.00*
□ ELEMENTS OF BASIC *£9.00*
□ GRAPHICS ON MICROCOMPUTERS *£4.00*
□ INFORMATION HANDLING BY MICROCOMPUTERS *£4.00*
□ INTRODUCING COMPUTER PROGRAMMING *£11.50*
□ INTRODUCING DATA PROCESSING *£6.50*
□ INTRODUCING MICROPROCESSORS *£9.00*
□ INTRODUCING WORD PROCESSING *£8.50*
□ OPERATING SYSTEMS FOR MICROCOMPUTERS *£3.50*
□ STUDENT NOTES ON NCC DP DOCUMENTATION STANDARDS *£5.50*
□ THE ROBOTS ARE COMING *£10.00*
□ USING COMPUTERS — MANAGER'S GUIDE *£7.50*
□ WORKING WITH COMPUTERS: A GUIDE TO JOBS AND CAREERS *£2.50*

Name: ....

Address: ....

....

....

Amount: ....

Make cheques payable to ASP Ltd.

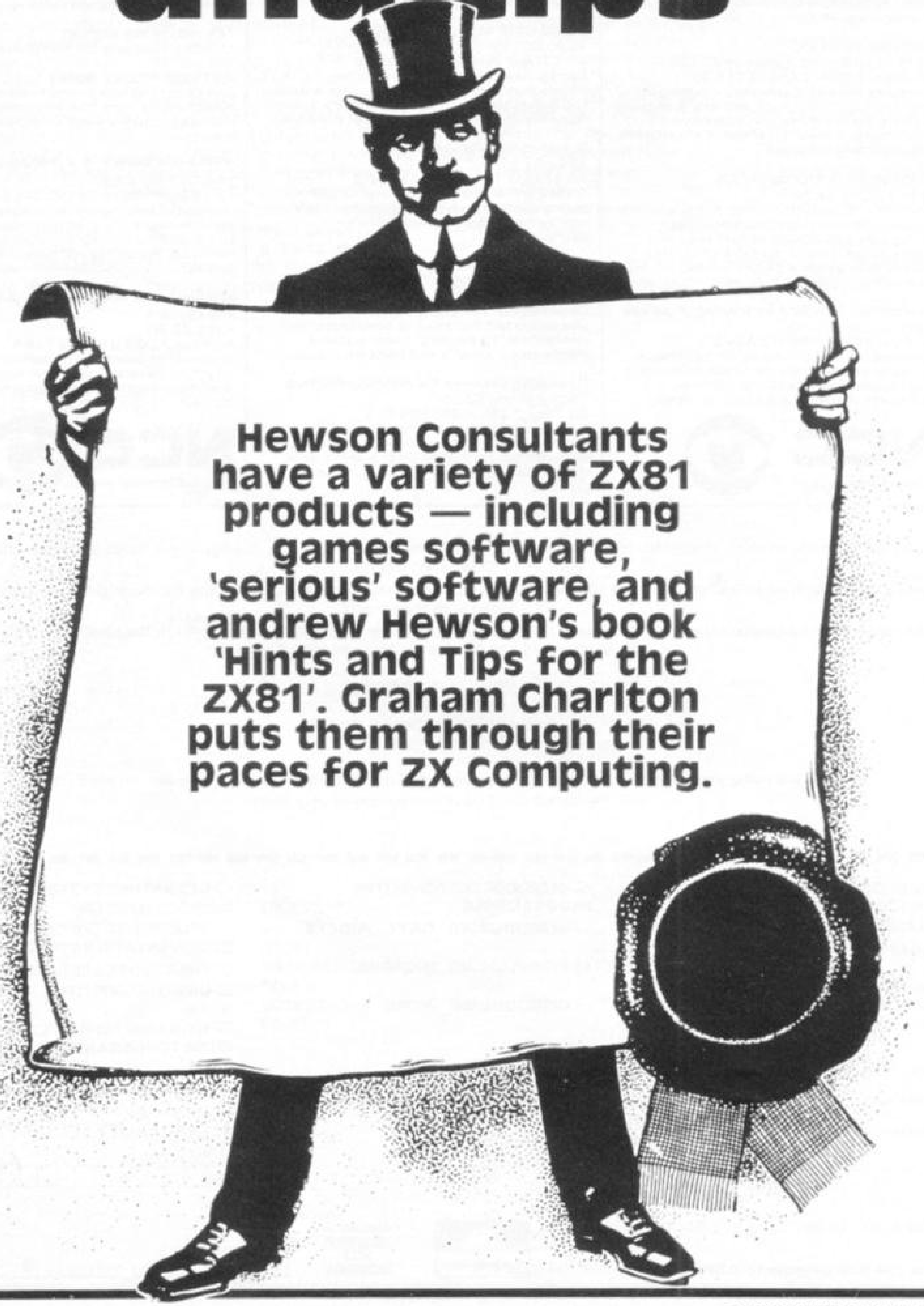
We welcome Access

BARCLAYCARD VISA

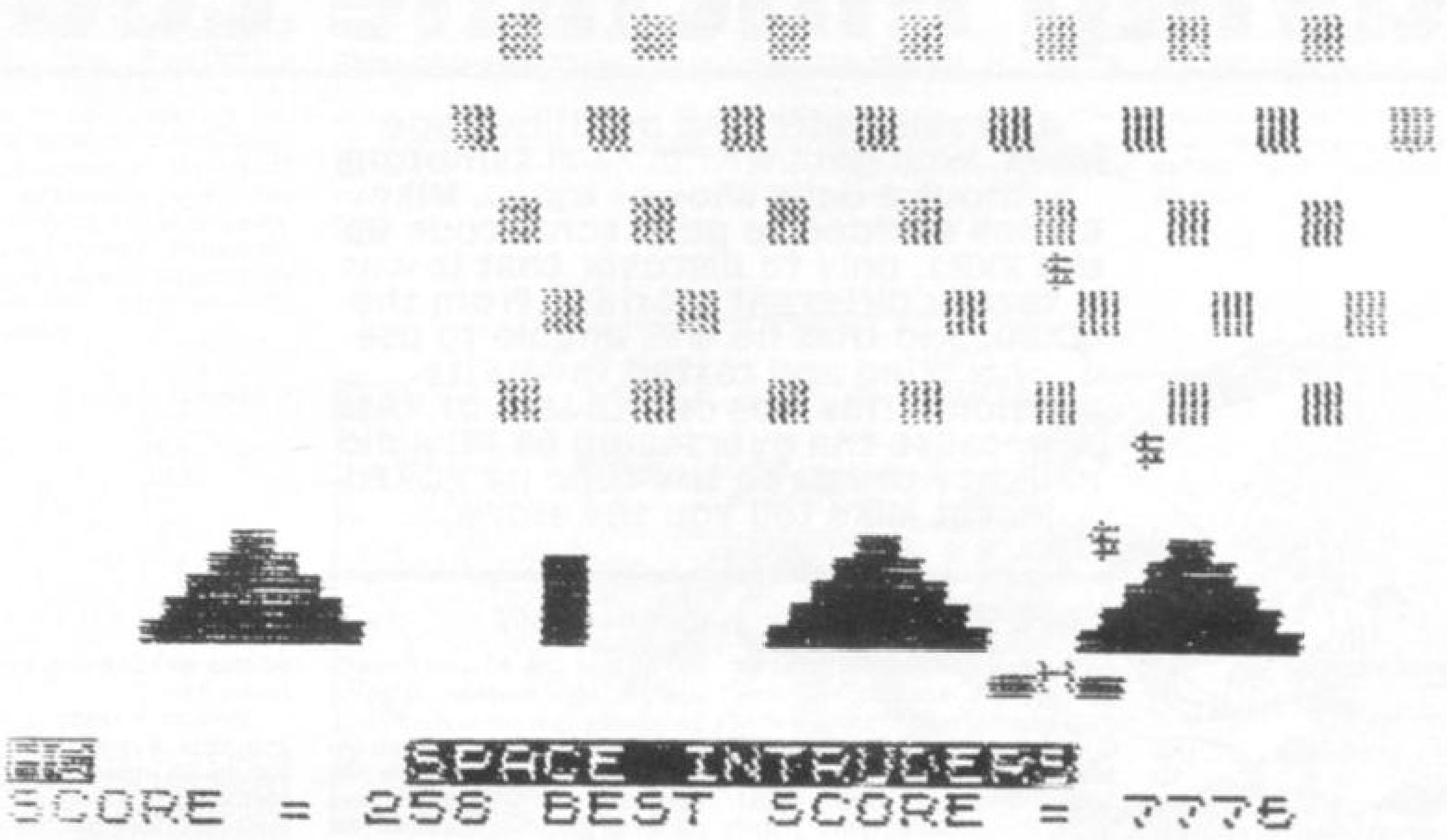
I wish to pay by
BARCLAYCARD □ ACCESS □ tick
Card No.
□□□□□□□□□□□□□□□□

Signature:

Hewson's hints and tips



Hewson Consultants have a variety of ZX81 products — including games software, 'serious' software, and andrew Hewson's book 'Hints and Tips for the ZX81'. Graham Charlton puts them through their paces for ZX Computing.



```
SPACE INTRUDERS
SCORE = 258 BEST SCORE = 7776
```

A big bundle of material from Hewson arrived at my home, and I spent a long weekend working through it. The products I looked at, and their prices, were the book 'Hints and Tips for the ZX81' (£4.25) Space Intruders (£5.95), Language Dictionary (£3.75) and Planet Lander (£3.75).

The area covered by the hints and tips book is shown clearly by the chapter titles: Saving Space; Understanding the Display File; Converting ZX80 Programs; Chaining Programs; Machine Code; plus a mixed bag of twelve 1K programs.

The book appears to be aimed mainly at those with only 1K RAM. If you have only 1K, I'm sure you realise how difficult it is to run even the smallest of programs. Therefore it is essential to understand how your ZX81 works, and know some short cuts to help you in programming. This book should help you. 'Hints and Tips' contains an extensive 27-page chapter on machine code, although it doesn't give a full list of codes. Hewson Consultants do supply, for £1.45, a full list, in which each Op Code is explained and cross-referenced. The book eases the user into the idea of using machine code, with many examples of its use in the book.

As well as this, I found the chapter on chaining programs quite enlightening. I didn't know, until I read it here, that it was possible to pass data between programs. In all, this slim volume is good value, and an essential aid to those with an unexpanded ZX81.

Space Intruders

This program, for a ZX81 with 16K, comes dubbed once on each side of a cassette, along with full instructions. Even if you'd never used a ZX81 before you got this program, the instructions would enable you to load and use the computer. Written in machine code for fast action, the program has 40 alien ships in each squadron, three laser guns and full score display. As well as this, the ZX81 can play itself — although not very successfully. Continuous firing is available, and you have to be quite quick to avoid the aliens' bombs. All in all, this is an enjoyable version of the popular arcade game.

Language Dictionary

Written for a ZX81 with 16K, Language Dictionary is an easy-to-use program, with a number of options. It can search for a word in either of the languages used, allows you to update or dump the entire contents or to save these on cassette. Although the method used to store the words packs them tightly it appears it would take a long time to find a word stored near the end when the dictionary was full.

Statistics

This pack consists of three 1K programs — Graph Plot, which does exactly as the title suggests, using data entered from the keyboard; Chi Squared Test, which "calculates the value of the chi squared statistics for comparing observed and expected values"; and Statistics, Regression, Trend which is used to calculate mean and standard deviation and a least squares regression line or trend line. I am sorry to say that these programs appear to do little more than someone with a pocket calculator and a piece of graph paper could achieve — and probably in less time than it takes to locate and load the required program. Of course, it would be valuable if a large number of similar problems had to be worked through. At the very least, the programs prove convincingly that the ZX81 can be used for more than games. Perhaps a 16K maths tape would be more useful, and would provide a better indicator of the ZX81's abilities.

Planet Lander

This tape contains four 1K programs. In Planet Lander you have to (yawn) land your spacecraft on a planet's surface. Stopwatch is self-explanatory. The third program is Space Docking, in which (yawn again) you have to dock your spacecraft with the space station. The final program is Clock, in which you set the time and then sit back and watch the display increments each five seconds. Two of these programs are available from the book 'Hints and Tips' so I suggest you buy the book. You'll learn a fair amount, and enjoy using the book far more than you will this last pack.

Saving machine code

As a self-confessed machine code freak, who gets withdrawal symptons without a daily shot of bytes, Mike Biddell decided to push some code up the ZX81, only to discover that it was a totally different machine from the ZX80, and that he was unable to use his tried and tested favourite methods. This was due to lack of RAM or because the overseeing 8K ROM did naughty things to the code he POKEd in. Let Mike tell you the story . . .

Summary Of Possible Methods

I first considered all the possibilities:

1. Establishing a dummy array with DIM A(x) and then POKE into VARS.
2. Moving RAM TOP down (as Mr. Sinclair has provided two nice little POKEable bytes 16388/16389 for just this purpose) and then POKEing the code into the available space NEW and CLS can't touch it, which is a major advantage.
3. POKEing into a REM statement. Here at least, the code would be saved and safe from attack, but I was worried about displaying the REM and the dreaded system crash that would follow when the basic interpreter found naughty, non-displayable codes for it to put on the screen.
4. POKEing above program and display file, but far enough below the stacks, to prevent collision with the "calculator stack". At least the code is immobile here, but I never liked this one with the ZX80, as it's a bit "by guess" as to whether the code gets overwritten or not.
5. Setting up a string array to contain the code and stripping it out byte by byte. This gobbles up memory at a phenomenal rate, plus the fact that you are tied to relative code as the strings shift.

The Sinclair Manual

The Sinclair Manual is as good as useless on using machine code (read Chapter 26), but to be fair, it's directed at those new to programming and in this it succeeds. The Sinclair Manual mentions methods 2, 3 and 5 with scant detail, but I began to wonder why there was no mention of a dummy array and VARS.

Experimentation

So I set out to try this method first. I have always favoured MICROMON (the ZX80 magic book) for this purpose. Micromon pokes hex code into VARS (system variable area). I converted Micromon into something the ZX81 could recognise (no TL$ on the 81 you have to use (2 to) instead). Having got the program in and running, I discovered that there were only about 50 bytes left for the code and that DIM A(25) was about all that was possible. Not only that, I could not POKE into the VARS area, all my carefully considered bytes were overwritten to zeros. Perhaps I was doing something wrong, but at this stage I stopped trying to POKE into VARS.

Method two worked as described in the Sinclair Manual, but what's the use of code you cannot save, without a special routine that consumes precious memory. So I discarded this as not being a possible favourite.

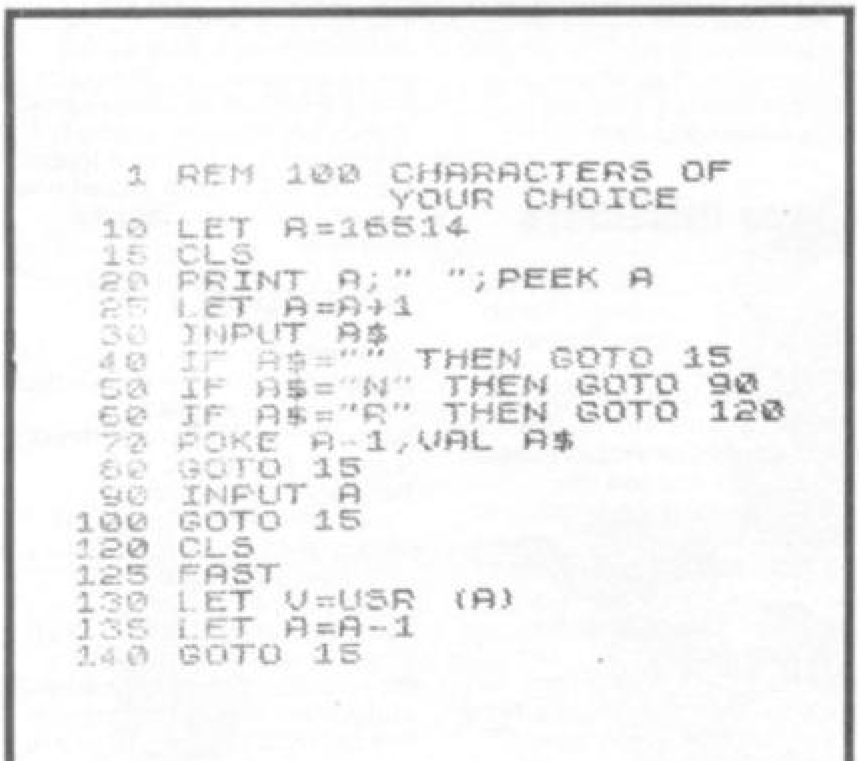
Method 4 really is a bit hit and miss and Method 5 uses too much memory. That left Method 3 POKEing into REM, a method I have not favoured in the past due to the display/system crash problem. However, not to be daunted by this, I wrote a program for experimentation.

Experimental Program

I decided that severe memory limitations on the basic ZX81 meant that the program must use decimal addressing and that the code should also be decimal. Having decided this, the program below was produced, to load 100 bytes of machine code.

Do not try to run machine code whilst compute and display is operating if you are going to use the accumulator; it won't RET. Hence line 125.

The program is started using GOTO 10 and prints the address of the first available byte after the REM statement, together with the contents, which will be the code of your chosen character. I used one hundred letter As and therefore found 38 returned at address 16514. Pressing NEWLINE only, steps through memory one byte at a time. "R" NEWLINE (RUN) runs the machine code from the address being displayed on the screen and returns to that address (assuming your code RETs successfully). "N" NEWLINE (NEW) allows you to jump to a new address in memory. Simply enter



```
  1 REM 100 CHARACTERS OF
            YOUR CHOICE
 10 LET A=16514
 15 CLS
 20 PRINT A;" ";PEEK A
 25 LET A=A+1
 30 INPUT A$
 40 IF A$="" THEN GOTO 15
 50 IF A$="N" THEN GOTO 90
 60 IF A$="R" THEN GOTO 120
 70 POKE A-1,VAL A$
 80 GOTO 15
 90 INPUT A
100 GOTO 15
120 CLS
125 FAST
130 LET V=USR (A)
135 LET A=A-1
140 GOTO 15
```

the new address (in decimal) followed by NEWLINE. I spent many happy minutes stepping through Clive's ROM after entering "N" NEWLINE, 0 NEWLINE, followed by a succession of NEWLINEs. If you have the patience you could list and disassemble the ROM (please send me a copy). Any area of RAM or ROM can be examined in this way. If you have the program operating correctly the first ten bytes of ROM should read:

Address	Decimal Content
0	211
1	253
3	255
4	127
5	195
6	203
7	3
8	42
9	22
10	64

The next step was to try some machine code that actually did something, so I decided to produce a short tone for my add on sound box. (Use any amplifier of 2 mV sensitivity plugged into mic. socket of your computer.)

Producing a Bleep

So GOTO 10 followed by NEWLINE produced 16514 38. The 38 was simply the first letter A of my REM statement. The following code was then written in:

Address	Decimal Byte
16514	14
16515	255
16516	6
16517	112
16518	211
16519	255
16520	16
16521	254
16522	6
16523	112
16524	237
16525	64
16526	16
16527	254
16528	13
16529	32
16530	241
16531	201

With some trepidation, I set the address to 16514 and pressed "R" NEWLINE. My sound box bleeped happily and I heaved a sigh of relief. (Use appendix A of the Sinclair Manual, if you want to convert the code to hex and unravel the bleep program.)

It didn't Crash

I accidentally displayed the REM statement with the code in it and to my amazement it didn't crash. Where it encountered a naughty code, clever old Clive's ROM simply told it to display a question mark.

In Summary

This will definitely be my favourite method of entering machine code since:

1. It's immobile and absolute addressing can be used.
2. It's crash proof even if REM is displayed.
3. It can be saved.

Using the experimental program also allows you to step through and examine the ROM and incidentally, run parts of the ROM, to find out start addresses of useful subroutines. Use the program and have a dabble with machine code yourself.

BIORHYTHMS

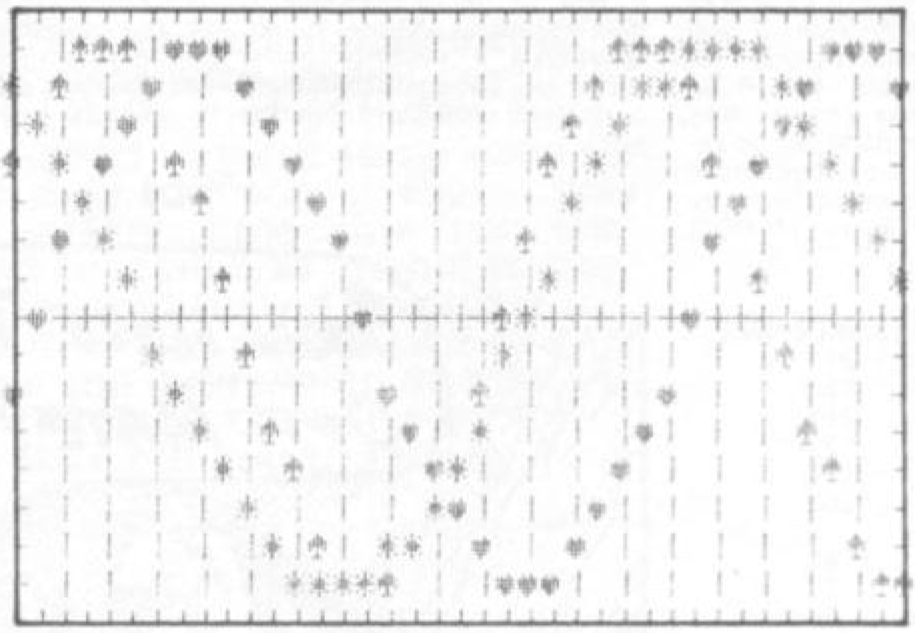
The 'science' of bio-rhythms is based on the belief that each of us is in the throes of three cycles which start at the moment of birth and continue until death.

PHYSICAL: The 23-day physical cycle relates to endurance, aggressiveness and physical strength.

EMOTIONAL: The 28-day emotional cycle governs emotions, optimism/pessimism, frustration, temper and moodiness.

MENTAL: The 33-day mental rhythm is related to logic, reasoning, ease of expression and common sense.

This program takes your date of birth, then the date of the day which interests you, and works out the 'worst point' on each of the three cycles in relation to the day you have nominated.



```
  20 GOTO 210
  30 LET T=0
  40 IF B-3>=0 THEN LET T=2
  50 IF T=2 THEN LET B=B+1
  60 IF T=2 THEN GOTO 90
  70 LET A=A-1
  80 LET B=B+13
  90 LET E=INT (365.25*A) +INT (3
0.6*B) +C
 100 RETURN
 110 LET T=0
 120 LET F=G-INT (G/D) *D
 130 IF F>D/2 THEN LET T=2
 140 IF T=2 THEN LET H=D-F
 150 IF T=2 THEN GOTO 180
 160 IF F=0 OR D/2=F THEN PRINT
"TODAY"
 170 IF F=0 OR D/2=F THEN RETURN
 180 LET H=D/2-F
 190 PRINT "AFTER ";H;" DAYS"
 200 RETURN
 210 LET Y$="THE WORST DAY IS "
 220 LET X$=" CYCLE"
 230 CLS
 240 PRINT "ENTER YEAR OF BIRTH
(AS 1950)"
 250 INPUT A
 260 PRINT "NUMBER OF BIRTH MONT
H (AS 12)?"
 270 INPUT B
 280 PRINT "DAY OF BIRTH (AS 30)
?"
 290 INPUT C
 300 CLS
 310 GOSUB 30
 320 LET J=E
 330 PRINT "NOW, TO FIND YOUR BI
O-RHYTHM","FOR A SET DAY, ENTER
THE"
 340 PRINT "DATE THAT INTERESTS
YOU"
 350 PRINT
 360 PRINT "YEAR (AS 1960)?"
 370 INPUT A
 375 LET Z=A
 380 IF A<1900 OR A>1999 THEN GO
TO 370
 390 PRINT "MONTH (AS 11)?"
 400 INPUT B
 405 LET Q=B
 410 PRINT "DAY (AS 24)?"
 420 INPUT C
 421 GOSUB 8000
 422 LET P=C
 425 GOSUB 30
 430 CLS
 440 LET G=E-J
 450 PRINT "BIO-RHYTHM FOR ";P;"
/";Q;"/";
 452 IF Z-1900<10 THEN PRINT "0"
;
 455 PRINT Z-1900
 460 PRINT
 470 PRINT
 480 LET D=23
 490 PRINT "PHYSICAL";X$
 500 PRINT Y$
 510 GOSUB 110
 520 PRINT
 530 LET D=28
 540 PRINT "EMOTIONAL";X$
 550 PRINT Y$
 560 GOSUB 110
 570 PRINT
 580 LET D=33
 590 PRINT "MENTAL";X$
 600 PRINT Y$
 610 GOSUB 110
 620 PRINT
 630 PRINT "ENTER Y FOR ANOTHER
DATE"
 640 INPUT U$
 645 CLS
 650 IF U$="Y" THEN GOTO 330
8000 CLS
8010 FOR I=1 TO 96
8015 PRINT "      ■";
8040 NEXT I
8050 RETURN
```

WHAT CAN I DO WITH 1K?

If your answer is "not much", then you must read the new book from V&H, 'What Can I Do With 1K? (40 programs and routines for the 1K Sinclair ZX81)'.

Here at last are some serious (and some fun!) programs which all run in 1K.

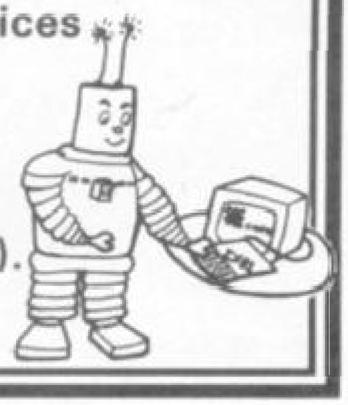
Don't be misled by the titles: 'Roulette' actually analyses gambling systems; 'Chinese Horoscope' reveals your birth sign, month sign, element and ascendant; 'Payroll' handles ALL tax bands. These are not just simple programs.

FREE OFFER!!!

A free listing of 'Chess in 1K' is included with all mail-orders mentioning this magazine!

'What Can I Do With 1K?' is available from most bookshops, computer shops, or direct from:-

V&H Computer Services
182c Kingston Rd.
Staines Middx.
Tel: Staines 58041



Price £4.95 (inc. p&p).
Also available on cassette £4.95.

ZXCHESS

TWO GREAT ZX81 16K CHESS GAMES

ZXCHESS(ENHANCED)

★ Written totally in machine code.
★ Full graphic display of Chess board.
★ Six levls of play: Two play within competition time limits.
★ Option to play Black or White.
★ Plays all legal moves including castling and en-passent.
★ Cassette routines for saving unfinished game and returning to later.
★ Displays moves of game on screen or printer for analysis.
★ Print a copy of the Chess board onto the printer.
★ Board can be set up in any position, you can even swap sides midgame.
★ clear whole board with one command: for end game analysis.

£6.50

ZXCHESS II

We believe the strongest ZX81 Chess game as no other has beaten it!!

★ All the features of ZXCHESS plus:
★ Book of 32 opening moves.
★ Seven levels of difficulty: FOUR play within competition time limits.
★ A move is suggested by the ZX81 if wanted.
★ Optional Full Graphic version using the QS CHRS Board.

£12.99

2 copies supplied on cassette with full instructions.
Cheques and postal orders payable to:

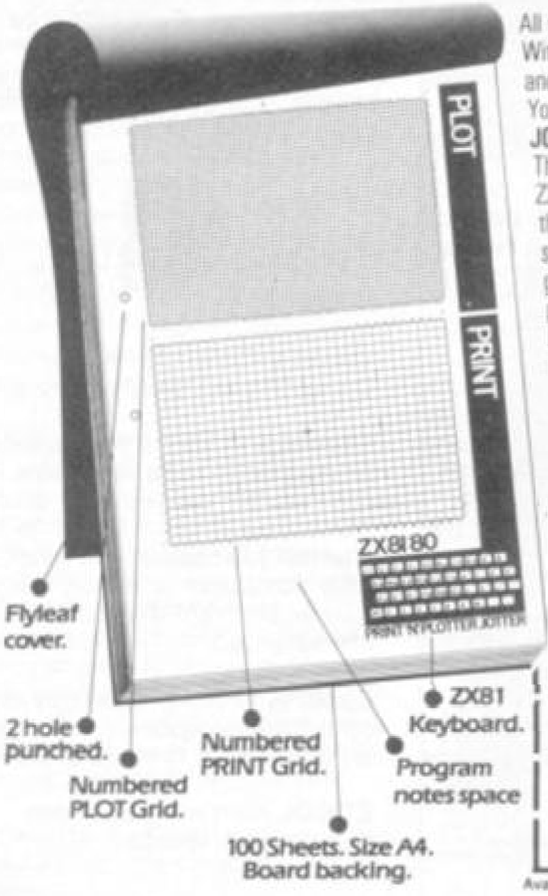
ARTIC COMPUTING
396 James Reckitt Avenue
Hull, North Humberside
HU8 0JA.

ARTIC COMPUTING

VISA

Access

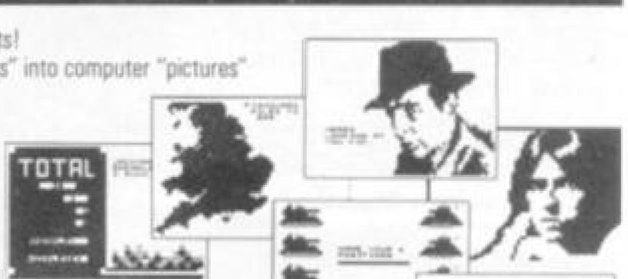
TURN 'LISTINGS' INTO DYNAMIC ZX GRAPHICS.



All you need is a pencil, ingenuity, and Print 'n' Plotter Products! Within hours you'll discover all sorts of ways to change "words" into computer "pictures" and produce really dynamic programs!

You can do it easily with the help of a **PRINT 'N' PLOTTER JOTTER** and **FILM.**

The **JOTTER** is a 100 page Graphics Pad designed for the ZX81/80. Every page has a PRINT and PLOT grid showing the screen locations. It tells you **any** part of the screen in seconds, and gives you a *'sketchpad' to produce incredible* graphics for games, business programs, information panels, tabulations — everything the normal 'listing' lacks!



GOOD NEWS
FOR
PRINTER
OWNERS

PRINT 'N' PLOTTER FILM is used with the JOTTER and is a matt, transparent, film version with identically-sized grids. Use it to co-ordinate PRINT or PLOT, or use it to 'copy' photographs, maps, charts, lettering or illustrations to use in programs. You can draw on the matt surface, and rub-out or wash it and use again! Print 'n' Plotter products will help **you** produce dynamic programs, whether you're converting existing listings or writing an original. Ask for them at your local Comp Shop, or send today for mail order.

NEW!
24 page manual.
"ZX GRAPHICS PROGRAMMING MADE EASY WITH PRINT 'N' PLOTTERS"
INTRODUCTORY OFFER:
£1.00
if purchased with a JOTTER or FILM.

If you do not wish to use this coupon write, stating your requirements;
To: **Print 'n' Plotter Products** (ZX)19 Borough High Street, London SE1 9SE.

□ Please send me . . . . JOTTERS @ £3.50 □ Please send me . . . . FILMS @ £2.25

□ Please send me . . . . MANUALS @ £1.00 **(£1.50 if ordered without JOTTER or FILM)**

□ Remittance enclosed □ Bill my Access/Visa No: ____________________

Name ____________________

Address ____________________

____

Note: Orders are usually posted within 48 hours of receipt. If you require 1st Class post please add 50p to order.

Print'n'Plotter Products©

Available at • **Buffer Micro Shop** 374a Streatham High Rd. London SW16 • **Denny's Booksellers** 2 Carthusian St. London EC1 • **Microware** 131 Melton Rd. Leicester

COMPUTING GLOSSARY

A

Address The number, eg 62768, identifying a place in *memory*.

Aluminised (paper) Printer paper surfaced with a thin deposit of metal. The characters are formed by darkening caused by electric current flowing onto the paper from the pins of a matrix print head.

ASCII American Standard Code for Information Interchange – representing letters, numbers etc by 128 permutations of a 7-bit code.

Assembler Program which converts the low-level mnemonic instructions of assembly language to the binary machine language instructions required to drive a *central processor*.

B

BASIC Beginners' All-purpose Symbolic Instruction Code – a popular high-level programming language developed at Dartmouth College, USA.

Batch (Processing) A method of computer working in which a large number of transactions are grouped together before processing (so that control totals, etc can be taken) and which are then passed through the various stages of processing as a group or batch. This was the original method of data processing for commercial work and contrasts with *interactive* and demand processing.

Baud A rate of data transmission commonly, though strictly not correctly, taken as synonymous with bits per second.

BCD Binary Coded Decimal – a 4-bit system for representing the 10 decimal digits.

Benchmark A standard computing task used to measure the relative speeds of different processors.

Binary Numbering system with the base 2, using the digits 0 and 1 instead of the decimal series 0 to 9. All digital computers work on data and instructions presented as binary numbers.

Bit Binary digit (contraction). Must be 0 or 1.

Block A sequence of *data words* or *bytes* treated as a unit, especially when working with magnetic tape.

Boot An instruction or very short program which will initiate a computer's operating system (short for bootstrap).

bps Bits Per Second – a rate of data transmission between devices. Eg 300 bps is a popular rate for some terminals, roughly equivalent to 30 characters per second (cps or chps).

Bubble memory A compact, high-capacity random access *memory* device which holds *data* as minute magnetic domains or 'bubbles'. The data is not lost when power is removed.

Buffer (1) An area of memory designated to hold *data* being transferred between devices working at different speeds, eg the fast processor and the slower keyboard, printer or disc.

(2) An electronic device in a signal path designed to allow signals to pass in one direction but to hold back unwanted reverse voltages which might damage the sending apparatus.

Bug An error in *software*.

Bus (sometimes spelt Buss) Basically means the multiple wiring common to several parts of a computer and the number of channels therein – eg a 16-bit bus addressing 64K memory locations or a 20-way bus addressing 1 megabyte. *Bus* is now generally identified with the pattern of connections to the plugs and sockets whereby optional units (eg more memory) may be connected to a computer.

Byte A unit of data *8 bits* long.

C

CAD/I/L Computer Aided Design/Instruction/Learning.

Cartridge A protective carrier of magnetic tape (a variant of the familiar cassette) or *disc*.

Central processor The heart of a computer in which the actual program instructions are effected.

Chain A process whereby one computer program automatically follows another.

COBOL Common Business Oriented Language.

Compiler A program whose function is to read another program written in a *high-level language*, such as COBOL or FORTRAN, and convert it to machine code which a computer can obey.

CP/M Control Program/Microprocessor. A popular disc-based operating system for microcomputers using the 8080 and Z80 processors.

cps Characters (rarely cycles) Per Second (sometimes chps).

CUTS Computer Users' Tape System — a standard for recording data on cassette tape.

D

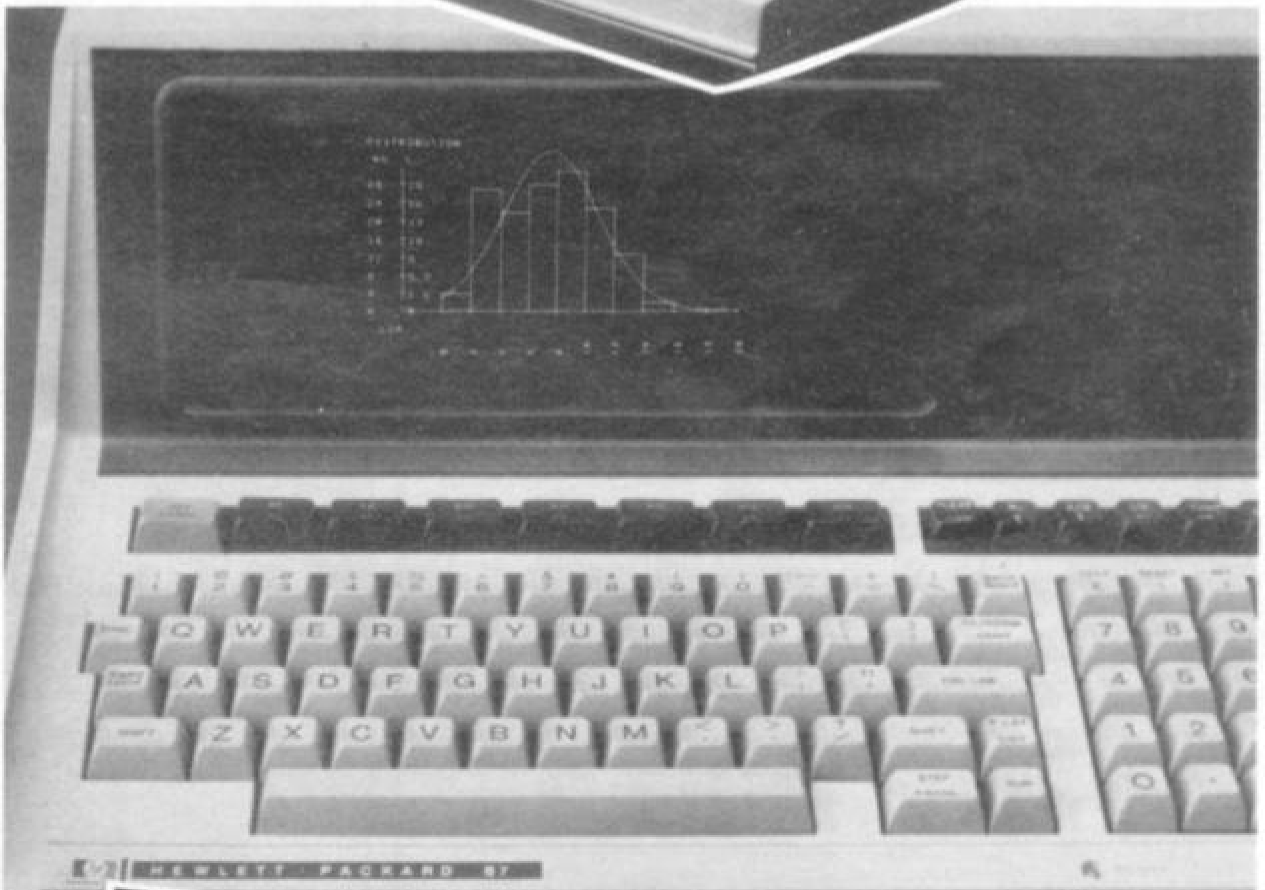
Daisy Wheel The typehead component of a sequential printer — like the *golfball* but faster — whose characters are held on the periphery of a serrated plastic disc.

Data base A system for organising elements of information in a *machine code* file so that a *program* can readily select from this *data* any particular abstraction or combination of information that may be called for. For instance, a customer data base might include full details of all customers (as required for service and distribution departments as well as sales and marketing) and also of every service call and delivery as well as each item invoiced to these customers during a year or longer. A suitable program could access that data base to answer such questions as "identify the customers buying more than £1,000 of item 'A' in less than five deliveries and receiving less than two service calls in the year."

Debug To correct the errors in a program.

Disc (Disk) Magnetic storage device allowing fast random access to any selection from a large volume of *data*. A full-size hard disc will hold say 5 megabytes or more, a smaller *floppy* disc typically holds from 80 to 250 kilobytes but in either case the capacity is being increased all the time.

Diskette A *floppy* disc, especially the smaller 5¼" size.



DOS Disc Operating System — a computer *operating system* held on magnetic *disc* rather than in *ROM*. An *initialisation* process will copy the operating system into *memory* whenever the computer is first turned on. Also an operating system which controls the disc themselves and may supplement, rather than replace, the computer's original operating system.

Duplex A mode of *data* transmission where each station can send and receive simultaneously.

Dynamic (Memory) Random Access Memory *(RAM)* requiring constant *refresh* signals but normally using less electrical power than *static* memory.

E

EAROM Electrically Alterable Read Only Memory. Typically taking 10 mSec to erase and 1 msec to write, this non-volatile storage might be better considered as 'Read Mostly Memory' as the write capability is likely to be limited to say 100,000 cycles.

Edit Alteration of text in *program* or *data* files. Often necessary, some systems make editing easier than others.

EPROM Erasable Programmable Read Only Memory. Writing typically takes one minute and erasing, by ultra-violet light, 10 minutes or longer.

F

Firmware *Instructions* or *data* permanently stored in *ROM.*

Floppy (disc) A mass-storage device comprising a soft (floppy) plastic disc with megnetisable surface on which data is recorded and may be accessed rapidly by a moving read/write head. The disc, either 8" or 5¼" diameter, rotates inside a protective cardboard sleeve.

FORTRAN Formula Translation, an early and still popular *high-level* programming *language*, mainly used for scientific purposes.

G

Golfball A type of typewriter (or the print head from which it gets its name) in which the print characters are embossed on the surface of a sphere very similar in size to a golfball. Rotation of the sphere brings the appropriate character into line for each required impression. The process is usually slow (15 cps) but of good quality.

H

Hard Copy A computer printout or listing on paper.

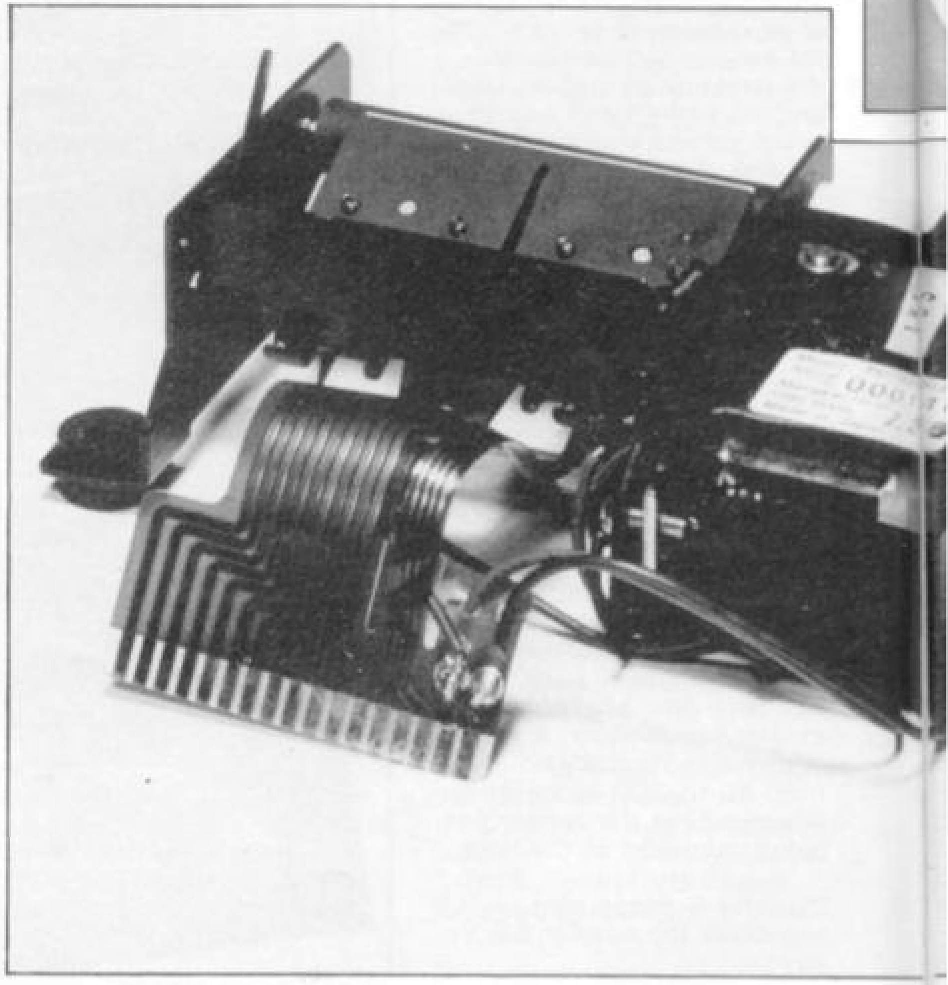
Hardware The physical elements of a computer (contrasted with software).

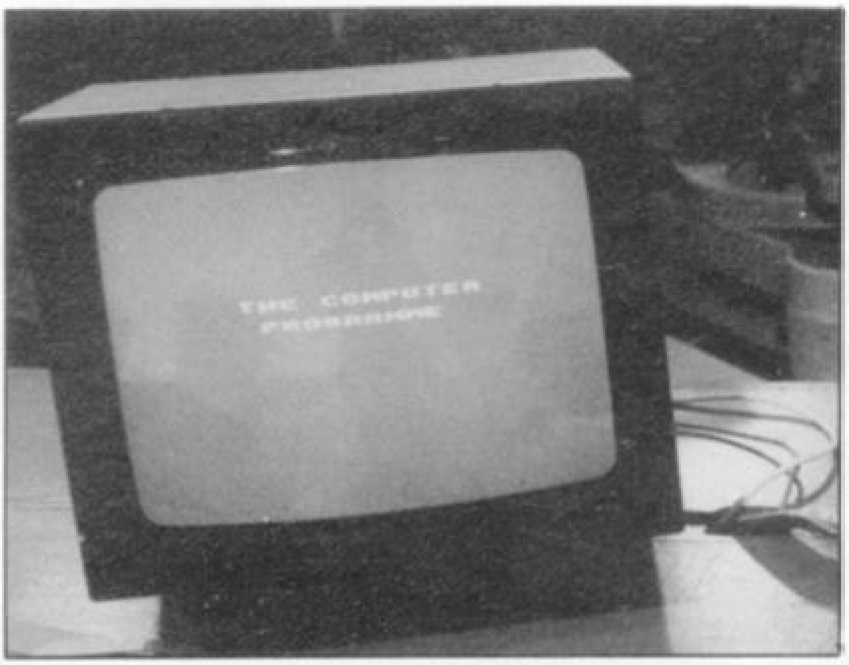
High-Level Language Programming language usually claimed to resemble a natural language,and with powerful *instructions*, each generating several *machine language instructions*. Examples include *BASIC, COBOL* and *FORTRAN.*

I

Intelligent terminal An input/output device which includes its own logic circuits and memory so that, for instance, data may be validated or changed in format before transmission to the main computer.

Interactive A working arrangement under which the computer reacts immediately to respond to any mistakes which may be made by the user





or to reply to his enquiries as soon as they are expressed. In some business activities, as also in program writing, this leads to much faster progress than would otherwise be possible.

Interface The interconnection arrangements between a computer and devices, such as printer or modem, attached thereto.

Interpreter A program to translate a high level language (typically BASIC) to machine language and to execute each instruction, line by line, immediately.

Interrupt A signal which suspends processing to allow some other command to be obeyed.

I-EEE Institute of Electronic and Electrical Engineers (in USA) — a body which has set a number of standards for more orderly interchange of information between various electronic devices, including computers.

I/O Input/Output.

Impact (printer) One which forms characters by striking a ribbon on to paper and can therefore produce carbon copies.

Integer BASIC Concerned only with whole numbers, cutting off any fractions or decimal parts.

K

Kilo (K) Abbreviation of Kilo, normally meaning 1000, but 1024 (2^{10}) when referring to *memory*.

Kansas (City) A standard for recording *programs* and *data* on cassette tape, named after the city where a conference was held, at which the standards were agreed.

L

Light Pen A stylus with a light sensor which allows a computer to identify the point at which a Video Display Unit *(VDU)* screen is being touched.

Line Printer A computer *peripheral* which prints a whole line at a time, instead of doing each character sequentially.

Load To copy a program (eg from tape or disc) into *memory*, ready for execution.

LSI Large Scale Integration — the combination of circuit elements in a small silicon chip.

M

Machine Language (code) The lowest (and tediously detailed) level of *program instructions*. All higher level coding must be converted to *machine language* (by *compiler* or *interpreter)* before a *processor* can obey it.

Mainframe A relatively large computer.The term derives from times before integrated circuits, when *processors* were wired up with large numbers of separate components mounted on circuit cards or boards which were in turn mounted in metal racks or frames enclosed in one or more large metal cabinets.

Matrix (printer) *Printer* whose characters are formed by selecting a pattern of dots from a matrix typically five dots wide and seven high.

Memory Immediate access *data* storage, directly addressable by a central processor and typically comprising a combination of *RAM* and *ROM*.

Micro (also u) Prefix signifying one millionth. Also used descriptively of something very small, though not as small as nano- or pico-.

Microprocessor An LSI chip holding a complete processor (arthmetic logic unit and control unit).

Microprogram A very low level of programming, normally implemented in ROM by the processor's manufacturer, to increase in effect the set of instructions which the processor can obey.

Minicomputer A somewhat vague term for the middle range of computers. Machines addressing up to 64K bytes or words of *memory* tend (at the present time) to be called *microcomputers* and machines able to address more than 64K memory locations tend to be called minicomputers unless they separate into distinct parts, in which case it may be called a *mainframe.*

Mini-floppy The smaller size of *floppy disc,* 5¼" in diameter.

Modem Acronym for MOdulator/DEModulator — a device adapting computer *data* for transmission by telephone line and vice versa.

Monitor The first level of computer *operating systems:* the *program* which turns *machine code* commands into action, managing input, output etc.

O

Operating System The computer's resident program which determines how instructions, input and output devices, etc are managed.

Overlay A program too long for the available *memory* may be entered and processed by instalments, each segment overlaying or replacing the code previously stored while the various values allotted to common variables would continue from one program to the next.

P

Package A set of programs designed to perform a common task, eg payroll, generalised to suit a variety of users. A turn-key package may comprise both the programs and the equipment on which they run.

Pascal Program language, designed to facilitate structured programming especially on small interactive machines. Named after Gabriel Pascal.

Patch A small piece of computer program inserted in a longer program to remedy some bug or defect in it.

Peripheral Device attached to a computer, eg printer, plotter, disc unit, but not necessarily essential to its use.

PILOT A programming language for small computers, designed to be particularly appropriate for teaching in schools.

Plotter Computer-driven graphic display using pen on paper.

PROM Programmable Read Only Memory.

R

RAM Random Access Memory. Read write memory. Data may be written to or read from any location in this type of memory.

Reset (button) A switch whereby computer control is returned to the monitor or low-level operating system and all internal variable values are changed to zero. This may be the only way of getting out of some endless loop which has arisen from a programming error.

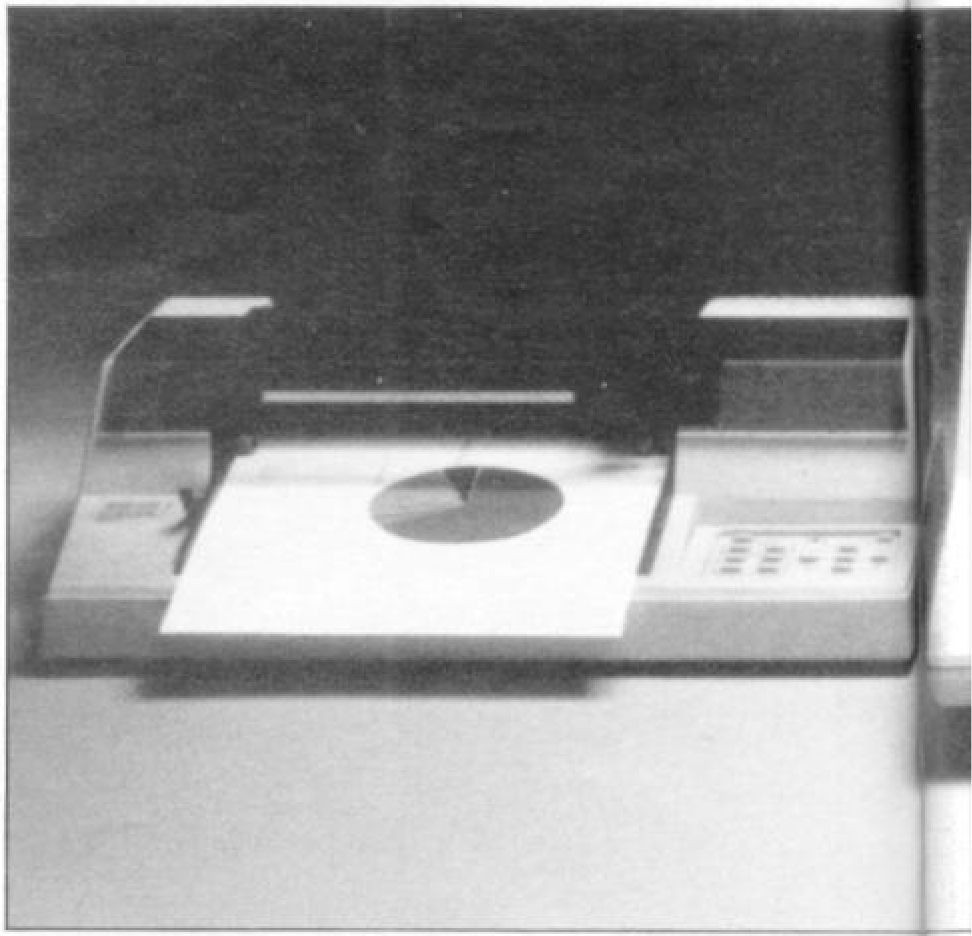
Return Return the key and corresponding computer instruction which sends the contents of keyboard buffer into a computer's memory for execution (term derives from 'carriage return' on a typewriter).

ROM Read Only Memory.

RS232 A communications interface used for modems and for serial printers.

Run The instruction to execute a program.

n-sec Nanosecond, one-thousand-millionth of a second.



S

S-100 Name of a bus or connection standard shared by many manufacturers and employing 100 connection positions. Unfortunately, there are some minor variations between different manufacturers' versions of the S-100 bus but the I-EEE has now defined a universal standard for it. Primarily designed as a *memory* bus and not for general purpose use.

Software The different kinds of program *required to work a* computer.

Source code A program written in one of the high-level languages and requiring compilation into machine language before use.

Static RAM Random Access Memory which does not require continuous refresh signals but tends to use more power than Dynamic RAM, and still loses its contents when power is removed.

String A sequence of *alphanumeric characters.*

T

Terminal A device, normally remote from the computer, at which data can enter or leave a communication network — eg a tele-typewriter working over telephone lines.

Thermal (Printer) A matrix printer wherein the print impression is made by heating a selected pattern of wires within a matrix (say 5 x 7) so that the heat causes points on the specially-treated paper to darken, to form the selected character.

Time-Sharing A method of operating a computer whereby two or more users apparently enjoy simultaneous access to and control of the machine. In practice what is happening is that the computer is attending to the users one at a time, but in a sequence of time intervals so short that none is normally aware of any delay.

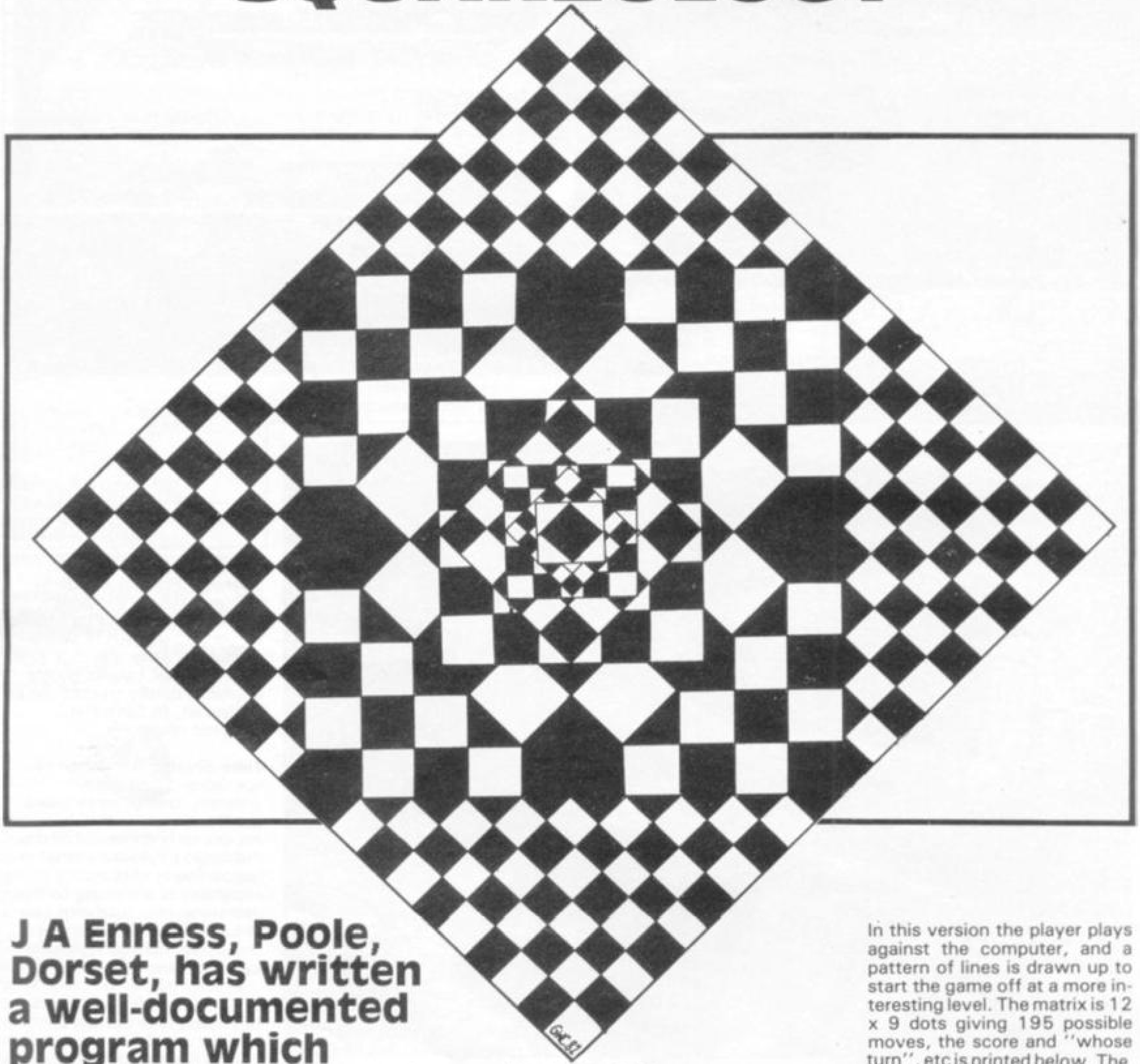
VDU Visual Display Unit — a television-type screen on which computer messages can be displayed.

W

Word The specified number of *bits* that a computer is organised to process as a group — eg 16-bit word: but the popular *8-bit word is* called a byte.

Word Processor A computer with software for entering, editing, storing, formatting and printing text, rather than processing figures.

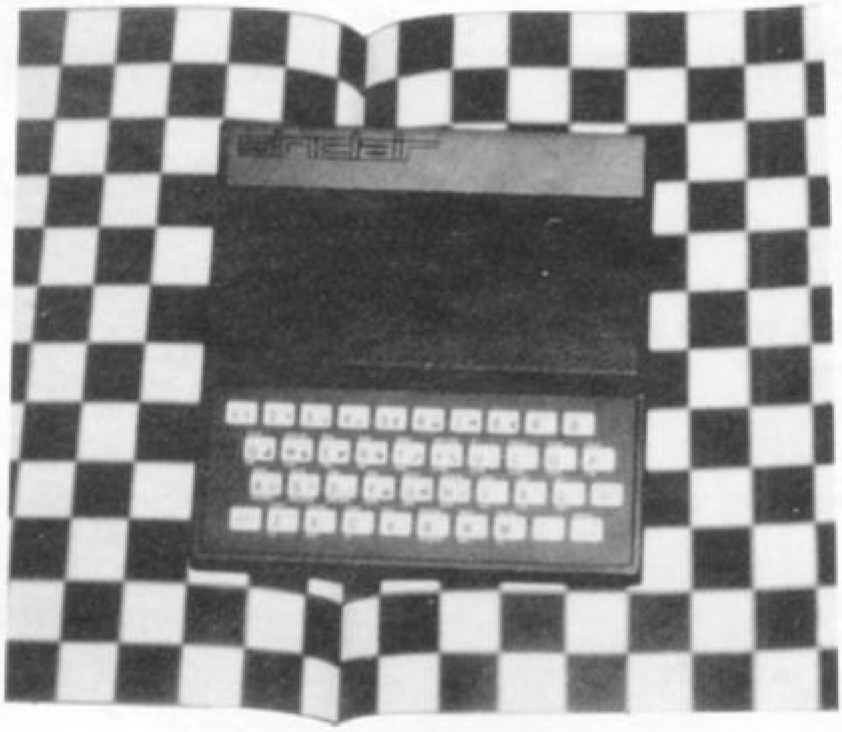
SQUAREOLOGY



J A Enness, Poole, Dorset, has written a well-documented program which plays a game based on the old favourite where two players take turns to link dots in a matrix, with each player trying to avoid placing a line in such a manner as to allow the other to complete a square. If a player completes a square, it is marked for the player, who then gets a second turn.

In this version the player plays against the computer, and a pattern of lines is drawn up to start the game off at a more interesting level. The matrix is 12 x 9 dots giving 195 possible moves, the score and "whose turn", etc is printed below. The pattern choice is not random in any manner and the same choice of pattern number will result in the same pattern.

The patterns 1-9 will fill up 60 or so moves in the matrix and then return to "Your move". Patterns from A-Z may or may not start with "Your move", and some squares may be "claimed" by the pattern. These will be marked with an "/" in the centre and are not counted on the score. If 0 is the choice of pattern then no pattern will be produced and the game will start with "Your move".



After taking your move the computer will react "My move" followed by "Search". The search routine then checks all 195 possible moves to see if it can claim any squares. If it can, it will complete the square and start again with "My move". Unfortunately the search takes approximately 12 seconds. This routine may be altered to a FAST mode by inserting a new line 153 FAST followed by a new line at 167 SLOW.

This means that the screen will blank out for a couple of seconds during the search, which in my opinion is not so good. If the search is unsuccessful the word "Search" is followed by an "X" after which it makes its choice of move. This is reasonably fast and normally takes only a second or two depending on the complexity of the situation. It does this by the simple reasoning: if one of its choices is impractical then it will not take that move into account again.

When the grid becomes so full that it is impossible to move without giving any squares away, the program is not able to take into account how many squares you can claim from its move. For this reason the computer scores two points for every square it takes.

The ultimate object being not only to win, but to beat the computer 88 to 0, which is almost impossible. If, however, you beat it by 50 points (which is very difficult), the coveted "Congratulations" line prints up.

The BREAK key will stop the program at any time. (This can be useful for sampling the patterns by simply: BREAK, RUN, another choice of pattern number.) Working out where the computer is going to move next has been deliberately made difficult to follow after its first couple of moves.

The Program

Lines 2-10 set up the instructions. Note:- Double words, (eg *fora*) are separated by the end of one line and the beginning of the next.

Line 12 goes to an inkey$ trap. (Sub 400.) The program will not continue until a key is pressed. The value of the key is recorded in Z$. Not used in this case.

Lines 16-26 set up the variables. A:- Keeps a record of the 195 possible moves. V:-Records the score. C:- Defines "whose move". F:- Starts the computer's turn position. G:-Indicates if there are any safe moves left. I:- Is used to search/ modify the data in A.

Lines 28-52 set up the display grid etc.

Lines 54-60 set up D which is used for the pattern routine.

Lines 62-64 set up "My score", "Your score". These look better in inverse characters. Sub 420 simply blanks out the line at 19, 0; and sets the next print position again for this line.

Lines 66-72 sort out whose move. (The pattern starts at 220. The "My move" starts at 150.)

Lines 74-100 set up all the variables and traps for "Your moves" choice. X = Value for the position across the grid. This can be from 0 to 11. (0 for A, 1 for B and C, 2 for D and E etc.) This, together with information of whether an odd or even line number down, sorts out the print position. This arrangement prevents illegal entries on the dots or in the middle of the squares. Y = the up/ down position.

Line 102 goes to the last line of Sub 470 to get the odd or even number which is stored in E. E=Ø for even, 1 for odd.

Lines 104-106 again prevents illegal entries and prevents entering a move already taken.

Llnes 108-110 store and print a move.

Line 112 Q= The number of sides to be taken into account in Sub 440. Sub 430 simply zeros J, K, L and M in Sub 440 these variables change to 1 if any of the four directions possible from a position has Q number of sides altogether (eg in this case Q=4 thus any four sides in any four directions will let J, K, L or M at least equal 1).

Lines 118-126 If the four sides of a square are present, then the program will go to various parts of subroutine 480. This routine works out by J, K, L or M, the direction of the print statement, which will 'black' the squares and insert an "*" a "–" or a "/" depending upon whose move claimed the square.

This simply takes the value of whose turn (in C) adds 22 and prints the CHR$. U is simply used to help simplify the print statements.

Line 128 If there is no pattern or winning move then let C=ABS (C–1). This changes the player's turns. If C=1 it becomes Ø and vice versa.

Lines 130-134 sort out the score. If all 88 squares are taken then the game finishes at line 144 where pressing any key starts another game. Otherwise line 136 starts the next turn by jumping back to line 64.

Lines 150-166 starts the search routine during "My turn" using part of Sub 440, only this time Q=3. If L or M=1 then its found a three-sided square. In this case line 200-210 loop around until the missing side position is found. Since "My turn" looks directly at a position (I) and doesn't choose X, Y co-ordinates then sub-routine 470 supplies X, Y, and E, it then jumps back into the original program at Line 106 having worked out its move.

Lines 168-196 constitute the computer's turn if the search for three sides is unsuccessful. This will choose any untaken position other than one which will make up more than two sides in any square. (Q=2 this time.) It starts at position two (I) and jumps in steps of 60, to find a suitable position. If I>195 then let I=I–193 keeps this within the "A" boundary. This tests each number from 2 to 195 without repeating, looking reasonably at random.

Once 195 is reached G=1 and Q=4 allowing the next available position to be taken. This time at line 196 we return to the main program 104, to allow a little longer to display the "My moves" choice (line 194). (X*2+37+E) works out the relevant letter.

Note: If you wish to set up the pattern fast then insert Line 221 FAST and line 69 SLOW.

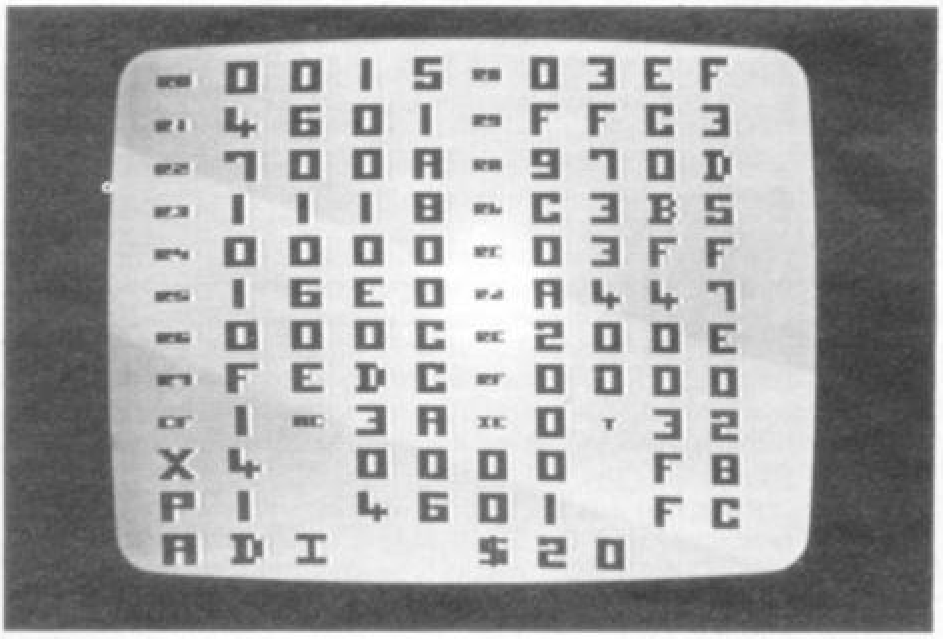
```
 2 CLS
 4 PRINT "SQUAREOLOGY",,,"BY J. ENNESS"
 6 PRINT AT 3,3;"THE OBJECT OF THIS GAME IS TOJOIN
   THE DOTS AND COMPLETE AS","MANY SQUARES AS
   POSSIBLE. THE"," COMPUTER ALSO TRYS THIS: AFTER
   ","  MAKING THE FOURTH MOVE ON ANY","THIRD
   SIDE."
 8 PRINT " [ ] [ ] [ ] THE COMPLETION OF A SQUARE
   ","WILL BLACKEN THE GREY LINES AND","PUT AN
   """*""" OR """–""" IN ITS CENTRE"," NOTE:- THE
   COMPUTER SCORES TWO","POINTS PER SQUARE YOU
   LOSE."
10 PRINT " [ ] [ ] [ ] TO START SELECT A PATTERN
   1-90R """0""" FOR NO PATTERN OR A-Z FORA
   MYSTERY START. TO ENTER A MOVEPRESS THE
   LETTER THEN NUMBER/S","OF THE REQUIRED
   POSITION. THE","WINNER OF A SQUARE HAS A
   SECOND MOVE.",," [ ] [ ] [ ] [ ] PRESS ANY KEY
   TO START"
12 GOSUB 400
14 CLS
```

```
 16 DIM A(195)
 18 DIM V(3)
 20 LET C=2
 22 LET F=-58
 24 LET G=0
 26 LET I=0
 28 PRINT "[ ] [ ] [ ] [ ] [ ]";
 30 FOR A=38 TO 60
 32 PRINT CHR$ A;
 34 NEXT A
 36 PRINT
 38 FOR A=2 TO 18 STEP 2
 40 IF A<10 THEN PRINT "[ ]";
 42 PRINT "[ ] [ ] [ ]";A;"■ ■ ■ ■ ■ ■ ■ ■ ■ ■
    ■ ■",
 44 IF A=18 THEN GOTO 52
 46 IF A+1<10 THEN PRINT "[ ]";
 48 PRINT "[ ] [ ] [ ]";A+1,,
 50 NEXT A
 52 PRINT AT 19,0;"PATTERN REQ. 1-9: OR MYSTERY
    A-Z"
 54 GOSUB 400
 56 LET D=CODE Z$
 58 IF Z$="0" THEN LET C=1
 60 IF Z$>"0" AND Z$<"A" THEN LET D=VAL Z$*4+38
 62 PRINT AT 21,0; "MY SCORE"; AT 21,15; "YOUR
    SCORE"
 64 GOSUB 420
 66 IF C=1 THEN PRINT "YOUR MOVE:";
 68 IF C=2 THEN GOTO 220
 70 IF C=0 THEN PRINT "MY MOVE:";
 72 IF C=0 THEN GOTO 150
 74 GOSUB 400
 76 IF Z$<"A" OR Z$>"W" THEN GOTO 74
 78 PRINT "[ ]"; Z$;"-";
 80 LET X=INT ((CODE Z$-37)/2)
 82 LET Y=0
 84 GOSUB 400
 86 IF Z$<>"1" THEN GOTO 94
 88 LET Y=10
 90 PRINT "1";
 92 GOSUB 400
 94 IF Z$<"0" OR Z$>"9" THEN GOTO 64
 96 LET Y=VAL Z$+Y
 98 PRINT Z$
100 LET I=X+(Y-2)*12-INT ((Y-2)/2)  '
102 GOSUB 474
104 IF Y<2 OR Y>18 OR E=0 AND X=0 THEN GOTO 64
106 IF A(I)>0 THEN GOTO 64
108 LET A(I)=1
110 PRINT AT Y-1,X*2 +4+E;"#"
112 LET Q=4
114 GOSUB 430
116 GOSUB 440
118 LET U=X*3+4
120 IF M=1 THEN GOSUB 486
122 IF L=1 THEN GOSUB 484
124 IF K=1 THEN GOSUB 484
126 IF J=1 THEN GOSUB 480
128 IF C<2 AND J+K+L+M<1 THEN LET C=ABS (C-1)
130 LET V(C+1)=V(C+1)+J+K+L+M
132 PRINT AT 21, 9; V(1)*2; AT 21, 27; V(2)
134 IF V(1)+V(2)+V(3)=88 THEN GOTO 138
136 GOTO 64
138 GOSUB 420
140 PRINT "I WON — HARD LUCK"
142 IF V(1)*2<V(2) THEN PRINT AT 19,3; "YOU WON —
    WELL DONE"
143 IF V(1)*2+50<V(2) THEN PRINT AT 19,0; "WON BY
    OVER 50 """CONGRATULATIONS""""
144 GOSUB 400
146 GOTO 14
148 REM MY TURN CHOICE
150 GOSUB 430
152 PRINT AT 19,12;"SEARCH";
154 LET Q=3
156 FOR A=24 TO 34
158 FOR I=A TO 172 STEP 46
```

```
160 GOSUB 452
162 IF L OR M=1 THEN GOTO 200
164 NEXT I
166 NEXT A
168 PRINT "X";
170 LET Q=2
172 LET F=F+60
174 IF F=195 THEN LET G=1
176 IF F>195 THEN LET F=F-193
178 IF A(F)>0 THEN GOTO 170
180 LET I=F
182 GOSUB 470
184 IF G=1 THEN GOTO 194
186 GOSUB 430
188 GOSUB 440
190 IF J+K+L+M>0 THEN GOTO 170
192 GOSUB 430
194 PRINT "[ ]"; CHR$ (X*2+37+E);"-";Y
196 GOTO 104
198 REM SEARCH SUCCESS
200 IF L=0 THEN LET L=-1
202 IF A(I)=0 THEN GOTO 212
204 LET I=I+11*L
206 IF A(I)=0 THEN GOTO 212
208 LET I=I+1*L
210 GOTO 202
212 GOSUB 470
214 GOTO 106
218 REM PATTERN CHOICE
220 IF I>0 THEN NEXT B
222 IF I>0 THEN LET C=0
224 FOR B=1 TO 60
226 LET I=I+D
228 IF I>195 THEN LET I=I-193
230 GOSUB 470
232 GOTO 106
398 REM INKEY$
400 IF INKEY$ <>"" THEN GOTO 400
402 IF INKEY$ ="" THEN GOTO 402
404 LET Z$=INKEY$
406 RETURN
418 REM PRINT BLANKS
420 PRINT AT 19,0;"[ ] [ ] [ ] [ ] [ ] [ ] — Line of 32
    spaces — [ ]"
422 PRINT AT 19,0;"[ ] [ ] [ ]";
424 RETURN
428 REM ZERO SEARCH VAR.
430 LET J=0
432 LET K=0
434 LET L=0
436 LET M=0
438 RETURN
439 REM SEARCH AROUND I
440 IF E=0 THEN GOTO 452
442 IF X>10 THEN GOTO 448
444 IF A(I)+A(I+1)+A(I+12)+A(I-11)=Q THEN LET J=1
446 IF X<1 THEN RETURN
448 IF A(I)+A(I-1)+A(I-12)+A(I+11)=Q THEN LET K=1
450 RETURN
452 IF I>172 THEN GOTO 458
454 IF A(I)+A(I+11)+A(I+12)+A(I+23)=Q THEN LET
    L=1
456 IF I<24 THEN RETURN
458 IF A(I)+A(I-11)+A(I-12)+A(I-23)=Q THEN LET
    M=1
460 RETURN
468 REM FIND E.X.Y. FROM I
470 LET Y=INT ((I+((I+11)/23-1))/12)+2
472 LET X=I-((Y-2)*12-INT ((Y-2)/2))
474 LET E=INT (Y/2+0.5)*2-Y
476 RETURN
478 REM PRINT WINNER
480 IF K=1 THEN LET Y=Y-1
482 LET U=U+2
484 LET Y=Y+2-E
486 PRINT AT Y=-,U;"■"; AT Y-2,U-1;"■"; AT Y-2,
    U+1;"■"; AT Y-3,U;"■"; AT Y-2,U;CHR$ (C+22)
488 RETURN
```

WHAT'S ON NEXT?

IN HOBBY ELECTRONICS



```
0 0 1 5   0 3 E F
4 6 0 1   F F C 3
7 0 0 A   9 7 0 D
1 1 1 8   C 3 B 5
0 0 0 0   0 3 F F
1 6 E 0   A 4 4 7
0 0 0 C   2 0 0 E
F E D C   0 0 0 0
1   3 A   0   3 2
X 4   0 0 0 0   F 8
P 1   4 6 0 1   F C
A D I     $ 2 0
```

The HE MicroTrainer.

PROJECTS FOR EVERYONE

* Light-Operated Power Switch
* Auto-Wah Effect
* Greenhouse Monitor
* Telephone Timer
* Power Supply Design Project

Next month we begin a brand new computer hardware project, the HE MicroTrainer. Based on the 1802 microprocessor, it is simple to construct and easy to operate, the perfect machine to introduce the 'nuts and bolts' of microcomputer technology. The MicroTrainer has been designed specifically for Hobby Electronics' readers, and offers a number of distinct advantages over other development systems:

- Low cost.
- Modulated video output for direct input to a domestic television receiver, permitting more information to be displayed than is possible with LED readouts. The 12 line x 12 character display shows the complete set of 1802 registers or 32 bytes of program memory.
- Displays the current instruction, in mnemonic form.
- Unique single-step operation from RAM or ROM based software.
- Cassette tape storage of user programs.
- Twenty command functions, eg RUN, STEP, STOP, RESET, INTERRUPT, INSERT, DELETE, SAVE, LOAD,
- Twenty key keypad for direct input of Hexadecimal data or command functions.
- Includes 1.5 KBytes of RAM for user programs.
- Optional 24 line I/O port
- High quality, double-sided through-hole-plated printed circuit board simplifies construction.

PLUS

A New Feature Series
The Electronic Revolution
The development of electricity and electronics, from Vota to Video.

AND

All our regular stars . . . . .
MONITOR
POINTS OF VIEW
INTO RADIO
CLEVER DICK

The June issue of Hobby Electronics will be in your newsagents on May 14th
DON'T MISS OUT . . . . PLACE YOUR ORDER NOW
Simply fill out the coupon and hand it to your local newsagent so that your copy will be reserved for you.

Please reserve . . . . . copies of the June issue of

Hobby Electronics

for
Name . . . .

Address . . . .

. . . .

. . . .

MACHINE SPECIFICATIONS

ZX80

Dimensions
Width 174mm (6.85 in)
Depth 218mm (8.58 in)
Height 38 mm (1. 5 in)
Weight 300g (10.5oz)

Microprocessor/Memory
Z80A 3.25 MHz clock
ROM: 4K bytes containing BASIC
RAM: 1K bytes internal, externally expandable to 16K bytes.

Display
Requires an ordinary domestic black and white colour TV. The lead supplied connects between the ZX80 and your TV's aerial socket. The display organisation is 24 lines of 32 characters per line showing black characters on a white screen. The ZX80 does not connect to a printer.

Programming
Programs can be entered on the keyboard or loaded from cassette. The ZX80 has automatic "wrap round" so lines of program can be any length but not multi-statement lines.

Syntax check
The syntax of the entered line is checked character by character. A syntax error cursor marks the first place the syntax breaks down if there is an error. Once any errors have been edited out the syntax error cursor disappears. Only syntax error-free lines of code are accepted by the ZX80.

Graphics
Total of 22 graphics symbols giving 48 x 64 pixels resolution consisting of 10 symbols plus space and inverses. Includes symbols for drawing bar charts. Under control of your BASIC program any character can be printed in reverse field.

Editing
The line edit allows you to edit any line of program or input including statement numbers. The edit and cursor control keys are EDIT, RUBOUT, HOME.

Arithmetic
Arithmetic operators +,–,x, ÷ exponentiate. Relational operators <, >, = , yielding 0 or – 1. Logical operators AND OR NOT yielding boolean result. Relational operators also apply to strings. ZX80 BASIC uses 16 bit two's complement arithmetic (± 32767).

Variables
Numeric variable names may be any length, must begin with a letter and consist of alphanumerics. Every character in the name is compared thus an infinity of unique names is available.

String variables may be assigned to or from, shortened but not concatenated. String variable names are A$ – Z$. Strings do not require a dimension statement and can be any length.

Arrays have a maximum dimension of 255 (256 elements) each. Array names consist of a single letter A–Z.

Control variable names in FOR. . . NEXT loops consist of a single letter A–Z.

Expression evaluator
The full expression evaluator is called whenever a constant or variable is encountered during program execution. This allows you to use expressions in place of constants especially useful in GOTOs, GOSUBs, FOR. . . NEXT etc.

Immediate mode
The ZX80 will function in the "calculator mode" by immediately executing a statement if it is not preceded with a line number.

Cassette interface
Works with most domestic cassette recorders. The transfer rate is 250 baud using a unique tape-recording format. Other systems are not compatible with the ZX80's. The ZX80 also SAVEs the variables as well as the program on cassette. Therefore you can save the data for updating next time the program is executed. The ZX80 does not support separate data files. The lead supplied with the ZX80 is fitted with 3.5mm jack plugs.

Expansion bus
At the rear has 8 data, 16 address, 13 control lines from the processor and Ov, 5v, 9-11v, Ø and internal memory control line. These signals enable you to interface the ZX80 to your own electronics, PIO, CTC, SIO if you want I/O ports etc.

Power supply
The ZX80 requires approximately 400mA from 7–11v DC. It has its own internal 5v regulator.

TV standard
The ZX80 is designed to work with UHF TVs (channel 36)and is the version required for use in the United Kingdom. The ZX80 USA is designed to work with a VHF TV(American channel 2. European channel 3) and is the version required for the American TV system, also for countries without UHF.

ZX81

Dimensions
Width 167mm (6.32 in)
Depth 175mm (6.80 in)
Height 40 mm (1.57 in)
Weight 350 gms (12.15 oz)

Microprocessor/Memory
Z80A 3.25 MHz clock
ROM: Containing 8K BASIC interpreter
RAM: 1K bytes internal, externally expandable to 16K bytes.

Keyboard
40 key touch-sensitive membrane. Using function mode and single press key-word system, this gives the equivalent of 91 keys and also graphics mode allows an additional 20 graphical and 54 inverse video characters to be entered directly.

Display
Requires an ordinary domestic black and white or colour TV. The aerial lead supplied connects the ZX81 to the TV aerial socket. The display is organised as 24 lines of 32 characters with black characters on a white background.

Two mode speeds
The ZX81 can operate in two software-selectable modes - FAST and NORMAL. FAST is ideal for really high-speed computing. In NORMAL mode however the ZX81 allows continuously moving, flicker-free animated displays.

Printer

The 8K ROM will permit instructions (LPRINT, LLIST and COPY) to drive the Sinclair ZX Printer.

Programming
Programs can be entered via the keyboard or loaded from cassette. Programs and data can be saved onto cassette so that they

are not lost when the ZX81 is turned off.

Syntax check

The syntax of a line of program is checked on entry. A syntax error cursor marks the first place the syntax breaks down if there is an error. The syntax error cursor disappears when errors have been corrected. Only lines free from syntax errors will be entered into the program.

Graphics

Apart from the 20 graphics characters, space and its inverse, the display may also be divided into 64 x 44 pixels, each of which may be 'blacked' in or 'whited' out under program control.

Editing

A line editor allows you to edit any line of program or input, including program line numbers. Lines may be deleted, increased or decreased in size.

Arithmetic

Arithmetic operators +, –, x, ÷, exponentiate. Relational operators =, < >, >, <,< =,>=, may compare string and arithmetic variables to yeild 0 (False) or 1(True). Logical operators AND, OR, NOT yield boolean results.

Floating-point numbers

Numbers are stored in 5 bytes in floating-point binary form giving a range of $\pm 3 \times 10^{-39}$ to $\pm 7 \times 10^{38}$ accurate to 9½ decimal digits.

Scientific functions

Natural logs/antilogs; SIN, COS, TAN and their inverses;SQR; e^x.

Variables

Numerical:	any letter followed by alphanumerics
String:	A$ to Z$
FOR-NEXT loops:	A–Z (loops may be nested to any depth.
Numerical arrays:	A–Z
String arrays:	A$ to Z$

Arrays

Arrays may be multi-dimensional with subscripts starting at 1.

Expression evaluator

The full expression evaluator is called whenever an expression, constant or variable is encountered during program execution. This powerful feature allows use of expressions in place of constants and is especially useful in GOTO, GOSUB etc.

Command mode

The ZX81 will execute statements immediately, enabling it to perform like a calculator.

Cassette interface

Works using domestic cassette recorders. The transfer rate is 250 baud and uses a unique recording format not compatible with other systems. The ZX81 will save the data as well as the program to avoid the need to re-enter the data when the program is next loaded.

ZX81 will search through a tape for the required program). The cassette leads supplied have 3.5 mm jack plugs.

Expansion port

At the rear, this has the full data, address and control buses from the Z80A CPU as well as OV, +5V, +9V,$\overline{\emptyset}$ and the memory select lines. These signals enable you to interface the ZX81 to the Sinclair 16K RAM pack and ZX printer.

Power supply

The ZX81 requires approximately 420mA at 7–11V DC. It has its own internal 5V regulator. The ready assembled ZX81 comes complete with a power supply. The ZX81 kit does not include a power supply.

TV standard

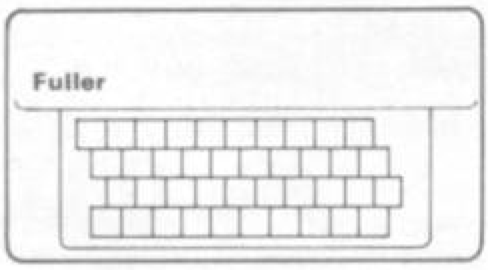
The ZX81 is designed to work with UHF TVs (channel 36) 625 lines.

ZX COMPUTING — AD INDEX

FULLER FD SYSTEM FOR ZX80/81

THE MOST VERSATILE SYSTEM FOR EXPANDING YOUR ZX

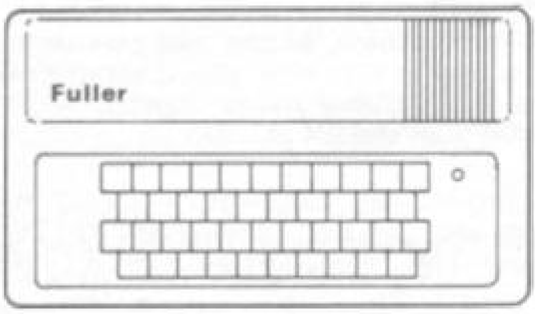
STANDARD KEYBOARD AND CASE



This splendid keyboard and case houses your ZX81 printed circuit board, which is simply screwed into place, the keyboard plugs into the ZX. You can now enter data with ease. The 40 key switch board is a custom unit not made up out of other manufacturers parts. The keytops are our own design and have the ZX Qwerty and functions foil printed onto them. Access to the user port, TV, MIC, and ear sockets are as per the ZX case.

Built keyboard and case £36.70 or £30.70 as a kit plus £2.10 postage and packing.

EXTENDED KEYBOARD AND CASE



The case is designed to house not only the keyboard and ZX but also our motherboard, power supply, RAM cards and two other boards, not necessarily of our manufacture. The injection moulded case measures 200 mm x 350 mm x 60 mm and houses a 42 keyswitch board, the extra keys can be assigned to other functions. The case is supplied with a "Power On" LED.

Built keyboard and case £39.95 or kit £33.95 plus £2.50 postage and packing. Motherboards £15.95 plus 80p postage and packing. 16k RAM board £35.95. 64k RAM board £79.95.

Keyboard Only Available!
Built £24.95
Kit £18.95 (+P.P. 80p)

Send SAE for details to:-
FULLER MICRO SYSTEMS, The ZX Centre,
Sweeting Street, Liverpool 2.

a must for your ZX81

PEEK
POKE
&BYTE
&RAM!
SHIVA
BASIC PROGRAMMING FOR THE
ZX81
By Ian Stewart & Robin Jones
Guaranteed Reader Friendly

Authors: Ian Stewart & Robin Jones

The reader-friendly guide to getting started with the Sinclair ZX81. Includes an introduction to looping and branching, graphics, subroutines, and debugging techniques, with over 50 programs designed to run on the standard 1K memory.

Approx. 130 pages Paperback £4.95

ISBN 0 906812 17 8

Published by: Shiva Publishing Ltd., 4 Church Lane,
Nantwich, Cheshire CW5 5RQ Telephone: (0270) 628272

Please supply me with ______________ copy/copies of: PEEK, POKE, BYTE & RAM

Price. £4.95 per copy. Cheques should be made payable to Shiva Publishing Ltd.

NAME (Capitals please) ______________

Full postal address ______________

____

For payment by Access/American Express:

Card No. ______________

Signature ______________

Order from your Bookseller, or fill in the coupon.

LEADERS IN COMPUTER GAMES

FIRST CHOICE FOR ZX: SUPPORT:

QS

ZX-81

ZX-80

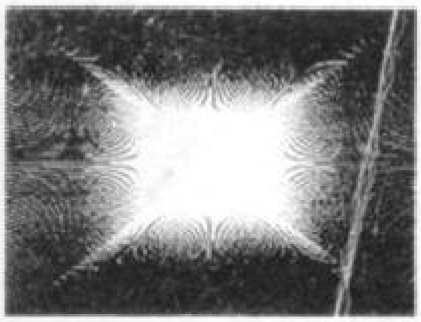
QS DEFENDER.
UP – DOWN – THRUST – FIRE
First and only full screen display. Software to drive QS SOUND BD. Moving Planetary surface. Up to 84 fast moving characters on screen at once. On screen scoring. Ten missiles at once. Increasing attack patterns. Requires 8K ROM, and 4K min of RAM. **£5.50.**

QS SOUND BD.
A programmable sound effects board using the AY-3-8910. 3 TONES; 1 NOISE; ENVELOPE SHAPER; + TWO 8 BIT I/O PORTS. Easily programmable from BASIC, the AY chip does most of the work leaving your computer free for other things. Signal O/P via 3.5 mm Jack socket Ports O/P via a 16 pin I.C. Socket. **£26.00.**

QS CHRS BD./
A programmable character generator giving – 128 SEPARATFLY PROGRAMMABLE CHARACTERS. ON/OFF 3WITCH. 1K ON BOARD RAM. Enables creation and display of your own characters to screen or printer. Demo cassette of fast machine code operation routines and lower case alphabet included. See below for ZX PRINTER listing. **£26.00.**

QS - LOWER CASE
abcdefghijklmnopqrstuvwxyz

QS INVADERS.
LEFT – RIGHT – FIRE
13×7 INVADERS; High score; 3 levels of play; RND saucers; Bonus base; Drives Sound bd. & CHRS bd. Requires 7K RAM, 8K ROM + Slow. **£5.50.**



QS HI-RES BD.
A Hi-res graphics board giving – 256 × 192 PIXELS. 6K ON BD. RAM. SOFTWARE SELECT/DESELECT. MIXED TEXT AND GRAPHICS. 2K ON BOARD ROM. Resident fast machine code graphics software (in ROM) provides the following HI-RES Commands. – MOVE x, y; PLOT x, y; DRAW x, y; BOX x, y; UP; DOWN; LEFT; RIGHT; PRINT A$; SCROLL; BLACK; WHITE CLEAR COPY. See above for ZX PRINTER listings using COPY. **£85.00.**

QS ASTEROIDS
LEFT – RIGHT – THRUST – FIRE
Software to drive QS SOUND BD. Multiple missiles firing in 8 directions. On screen scoring. Increasing number of asteroids. Full mobility of ship to all areas of the screen. Two asteroid sizes. Bonus ship at 10,000 points. Requires 8K ROM, 4K min of RAM + SLOW function. **£5.50.**

QS 3K RAM Bd.
An extremely reliable static RAM Bd. which combines with the computer's memory to give 4K total. Plugs direct in to the rear port on your ZX Computer. **£15.00.**

QS MOTHER BOARD BD. & QS CONNECTOR.
A reliable expansion system allowing a total of any RAM pack plus two other plug in boards to be in use at once. On board 5V regulator drives all external boards. Fitted with two 23 way double sided edge connectors. Connector is 2×23 way edge conns soldered back to back. Expansion can operate in two ways – (1) COMPUTER ↔ CONNECTOR ↔ Any QS add on bd. (but no extra RAM pack). (2) COMPUTER ↔ CONNECTOR ↔ MOTHER BD ↔ ANY RAM PACK. (2 bds to fit in mother bd.) Mother board **£12.00** Connector **£4.00.**

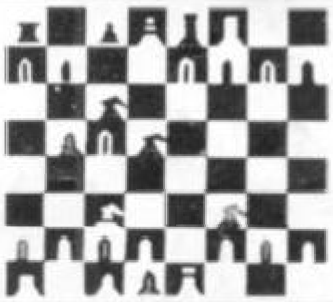
Special offers & news

(1) QS Mother bd. + connector + CHRS bd. + The special Graphics version of **ARCTIC COMPUTING'S ZX CHESS 11. £45.00.**
The strongest chess program with 7 levels of play.

(2) QS MOTHER BD + CONNECTOR + either SOUND or CHRS bd. **£40.00.**

We will be at the following shows . . . Come and see us . . .

THE COMPUTER FAIR, Earls Court, 23rd to 25th April
3rd ZX MICROFAIR, Central Hall, Westminster.
30th April & 1st May



POSTAL AND MONEY ORDERS TO: ALL PRODUCTS FULLY GUARANTEED.

QUICKSILVA: 95, UPPER BROWNHILL RD. : MAYBUSH : SOTON : HANTS : ENGLAND.

Please state Type of machine, Which ROM, Memory size, when ordering.