PERSONAL COMPUTER WORLD

50p U.S.$2.00

JUNE 1979

EXHIBITIONISM

—

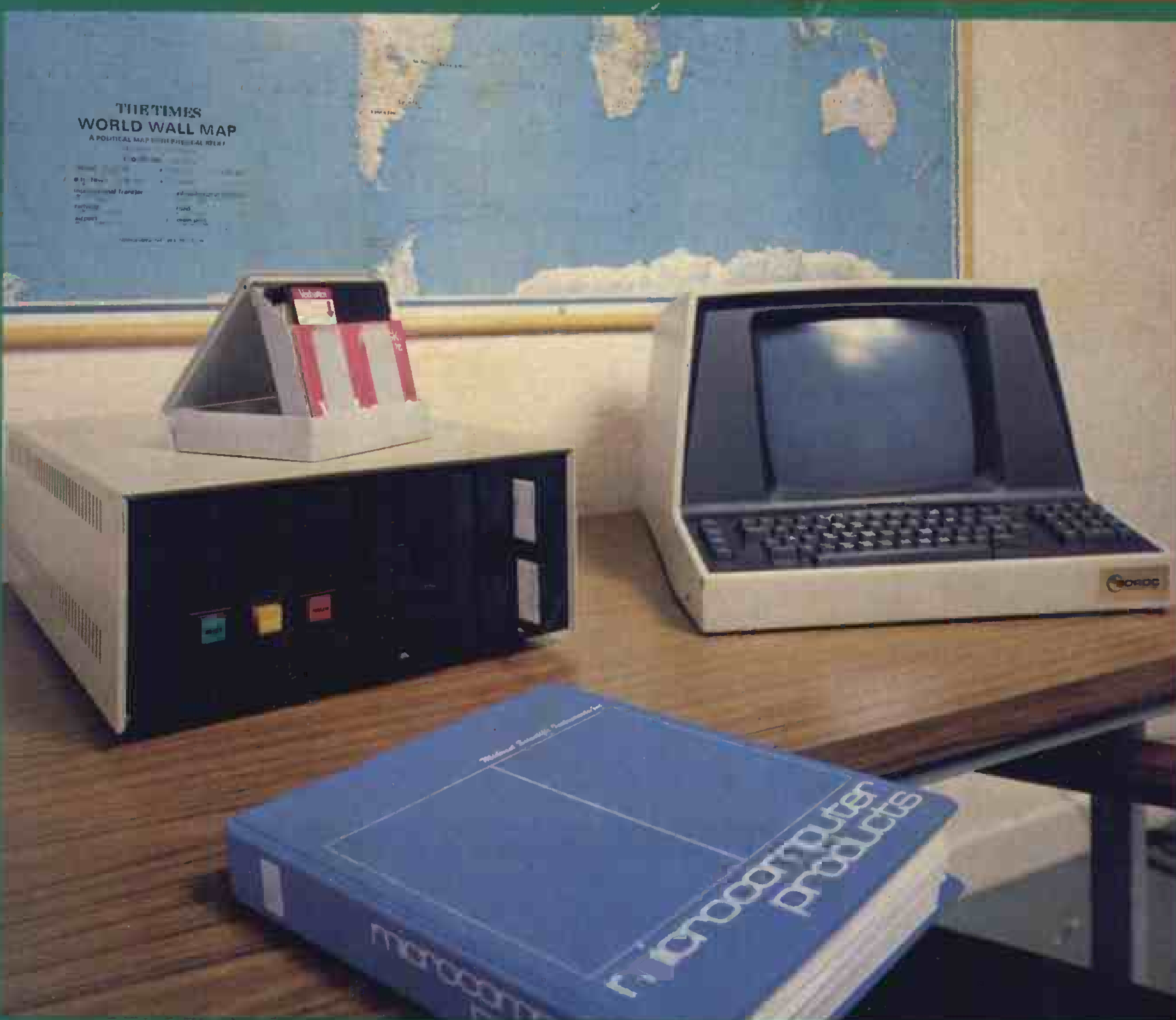
SMALL COMPUTERS FOR SMALL ORGANISATIONS

Europe's first magazine for personal computers for home and business use

Applevision

The Multilingual Machine

Assembly Point



PCW Review: The MSI 6800

Come to the Great British Electronics Bazaar

"ALLY PALLY"
Thursday-Saturday
28th-30th June.

FREE!

(AND WAIT TILL YOU SEE OUR SEMINAR PROGRAMME)

The Great Big 'Bazaar' for the hobbyist, amateur, and small buyer.

There's never been an event like this before.

First, the very scale of the exhibition is huge. Virtually all the companies you're used to hearing about (and buying from) will be there. Companies like Fluke and Gould showing off their low cost multimeters; smaller but important manufacturers like Lektrokit and Chromasonics; and even the R.S.G.B. who will have a station 'on the air' throughout the 'Bazaar.'

Then there are the suppliers of low-cost components and equipment. Plus almost all the journals in the business. Plus, oh, so many more interesting people catering for your needs (including computer kits!).

And you get in FREE if you send an s.a.e. (see alongside).

THE GREAT BRITISH ELECTRONICS BAZAAR

Our Symbol. We think it tells you just what the Bazaar is all about.

The Seminars.

If you would like to hear just what the experts have to tell you, a *season ticket* for three whole days can be yours for only £1.50.

Send an s.a.e. and we'll give you all the information (just use the coupon).

Our home for three days – Alexandra Palace, where it all began. (Our seminars are sited alongside the organ – for those who know this unique hall.)

SEMINAR TICKETS

£1.50.

I'd like to sit in at your seminars. (And like a free ticket to the exhibition.) Send me full details, please, and I enclose a large-ish s.a.e.

Name: ....

Address: ....

....

....

Post to: 'The Bazaar,' 34/36 High Street, Saffron Walden, Essex.

When?

Between Thursday to Saturday 28th–30th June.

You'll be in very good company; some ten thousand enthusiasts – and over a hundred stands displaying all that you want to see.

You'll come?

Eyes down for the appropriate coupon.

ADMISSION FREE

(or 50p on the door).

I'd like to see 'The Bazaar' FREE. I enclose a large-ish s.a.e. and will receive by return a ticket and full information.

Name: ....

Address: ....

....

....

Post to: 'The Bazaar,' 34-36 High Street, Saffron Walden, Essex. If you'd rather just pay 50p, go to Wood Green Tube Station and take a bus (every 3 minutes) to Alexandra Palace. We're open 10 am-6 pm daily, Thursday to Saturday, 28th–30th June.

PERSONAL COMPUTER WORLD

Europe's first magazine for personal computers for home and business use

UK 50p
US $ 2.00

Vol. 2 No. 2
June 1979

ISSN: 0142 0232

CONTENTS

David Hebditch's *On The Line* will be continued in the next issue.

Editorial and Advertising Office:
62A Westbourne Grove, London W2
Phone: 01-229 5599 (Publisher)
01-727 8758 (Editorial)
01-229 5589 (Advertising & Subscriptions)

Publisher:
A Zgorelec
Editor:
Meyer N. Solomon
Editorial Assistant:
David Bannister
Policy Adviser:
Peter Crofton-Sleigh, FRAS

Consultants
John Coll, Mike Dennis, Neil Harrison, Charles Sweeten, Patrick Sutton, Michael James, R. W. Davy, David Hebditch, Sheridan Williams, Stephen Castell, Dr. D.J. Hand.

Art: Sauveur Laurent Sant, Kathryn Hamme
Secretarial:
Vanessa Blackburn Kiddle
Layout Consultant:
David G. McFarlane
PCW Photography: Yoshi Imamura, Peter McGee
Typesetting & Artwork: Gilfillan
Cover Photo: MSI 6800

CONTRIBUTORS:

We welcome interesting articles written simply and clearly. You need not be a specialist to write for us. MS should not be more than 3000 words long, lines double spaced, with wide margins. Line drawings and photographs wherever possible. Enclose a stamped self-addressed envelope if you would like your article returned.

Manufacturers, suppliers and dealers are welcome to contribute technical articles, and send product information, but we are pledged to an independent viewpoint and will publish evaluations and reasoned criticism or praise, space permitting. Naturally there will be right of reply. Views expressed in articles are not necessarily those of Personal Computer World.

We may make arrangements to offer our readers products at special prices, for a limited period, in line withthe policy outlined above.

Published monthly by Intra Press, 62A Westbourne Grove, London W2. Phone 01-229 5599. Contents fully protected by copyright. All rights reserved. Subscription rates: Britain £8 for 12 issues. Prices include postage. USA – $10 for six issues. $20 for 12 issues. Continent and elsewhere £9.80 for twelve issues. Prices include postage.
Printed by Keats Printing Limited, 12 Fouberts Place, London W1V 1HH. Sole UK Distributors Seymour Press Ltd., 334 Brixton Road, London S.W.9., England. Distribution to specialist shops by Intra Press.

Editorial

This really should be headed "appeal". I am receiving a flood of articles, a lot of them publishable, but many handwritten. It goes against the grain to reject an article because it is not neatly typewritten or does not have a printed listing; but, really, some submissions are a typesetter's nightmare. If you do have to write, please use block letters. Don't let what I've just said discourage you from submitting articles. Only, remember the poor typesetter at our end.

SUBSCRIPTIONS

When PCW started publication, we had a special six-issue offer. When these subscriptions expired, we sent out reminders.
The renewal rate was 70%!
PCW reader loyalty is becoming a byword in publishing. If you're having difficulty in obtaining PCW at your newsagent, take our subscription. You can find the details at the foot of P.5.

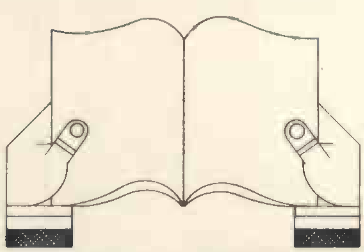
PCW Vol 1 BACK NUMBERS

Sold Out – No. 9
Sold Out – No. 7
Sold Out – No. 5
Sold Out – No. 4
Just a few left of – Nos. 3, 6.

If you think No. 10 will be available in 2 months' time – don't count on it. Nos. 1, 2, and 8 are still in good supply.

Think of the future – Look Backwards!

Losers - Weepers



BINDERS - KEEPERS

Half the worried people you meet today are not preoccupied with pollution, perversion or persecution. It's worse than that – they've lost a copy or two of PCW and don't know where to find replacements.

So keep *your* copies of PCW in a beautiful bright blue binder. £2.95 worth of smart security.

Binders are now available.
Orders already received get priority.

Payment by cheque or postal order £2.95,(includes p+p) made out to Intra Press. Please allow 28 days for delivery. "Binders", PCW, 62A Westbourne Grove, London W2.

Publisher's Letter

Dear Reader,

I am glad to announce that Computer Trader, our new publication, has had a good reception. It deals mainly with "trade" news, of the kind found in PCW's Tidbits, and is packed with information such as the full details of the MAPCON Scheme, and a monthly updated list of exhibitions. As such, it is increasingly in demand from those who have to or want to keep tuned to the micro business. If you're interested in seeing a sample copy, simply write in and ask for one.

STOP PRESS! £1,500 CASH – FIRST PRIZE

Second PCW Microprocessor Chess Tournament

Following the success of the first **PCW Chess Tournament** last year, we are pleased to announce that our second tournament will take place at the PCW show in London, November 2nd - 4th 1979.
It is hoped that some financial support may be available for private entrants from outside the U.K., to defray travelling expenses, and there will be at least one cash prize. The highest placed programs will be eligible to compete in the first **World Microprocessor Championships** which will be held at the 1980 PCW Show.

Detailed rules and entry forms will be available in due course. Prospective entrants are requested to write to David Levy (c/o Personal Computer World, 62a, Westbourne Grove, London W2) who will be acting as commentator and tournament manager.

SPONSORSHIP INVITED FOR 2nd & 3rd PRIZES


WE'LL SHOW YOU EVERYTHING
Because we've got the biggest and widest range of micro-computers, there's more for you to bite on at a Byte shop.
So you're not only sure of finding exactly what you want: you can take the opportunity to experiment before you buy.
And because we're backed by the huge financial resources of a major investment group, we'll be here tomorrow as we are today. All over the UK.
So whether you want a micro-computer for your home, your business, for industry, for education — or if you'd just like to find out which model you get on best with — you'll find a visit to a Byte Shop a
ew and invaluable experience.
You can call at a Byte Shop any time from Monday to Saturday.
Z-2D Computer System
the BYTE SHOP ltd
The Byte Shop
426/428 Cranbrook Road,
Gants Hill, Ilford, Essex
Telephone: (01) 554 2177
Telex 897311
Branches at:
LONDON (WEST END):
48 Tottenham Court Road —
tel (01) 636 0647
NOTTINGHAM: 92a Upper
Parliament Street —
tel (0602) 40576
MANCHESTER: 7-8 Corn
Exchange Building,
Fennel Street —
tel (061) 834 0220

Letters

Reader's ingenuity

I am writing to you in reply to your request for letters from micro-computer owners, since I own a Pet 2001-8.

I would like to offer the following information: I have not seen it mentioned anywhere previously.

It is widely known that if you require lower case letters on the Pet, you cannot have graphics characters simultaneously. This is not strictly true since you only lose the graphics characters which are accessed by:

[Shift, "any letter of the alphabet"]

Characters which do not have a lower case equivalent, e.g. numbers, punctuation marks, retain their shifted graphics character. In addition, four new graphics characters can be obtained whilst in the lower-case mode. These are:

shift colon, which gives a "tick" character i.e.

shift ↑ , which gives a 4 x 4 check pattern

shift) [close bracket], which gives a diagonally striped pattern, and finally

shift ←, which gives a similar pattern sloping the other way.

I would like to compliment you on the high standards set by your magazine, which despite the recent appearance of other rival magazines, I still regard as the best.

If I could make a few suggestions for future articles, I would like to see an article (or series of articles) on Assembly programming for absolute beginners, explaining how to put together an Assembly language program. Also some articles on ideas for games, not necessarily listings of games programs, but suggestions for games, for readers to write themselves; how about reviews on the American magazines, which can be obtained from some mail order firms, or reviews of easily obtainable software for the most popular machines.

Finally, being an optimist (I expect Commodore to release a printer for the Pet in my lifetime), I hope to write a Word Processor for the Pet.

Paul Hallam,
Wadham College, Oxford OX1 3PN.

Basically good

Your correspondent Mr. Greg Trice (Letters, April) condemns as cretinous the widely accepted BASIC language. Whilst this was never intended to compete with more sophisticated programming schemes it maintains its popularity because of the ease with which it may be implemented on small and simple computer systems.

If Mr. Trice were to own a very small computer, which is what a large proportion of **PCW**'s readers aspire to, I should be very interested to know whether he would choose to program in BASIC or machine-code.

Mark Morgan Lloyd
Lenton Hall, University Park, Nottingham NG7 2RB.

Defender of the here and now

Sometimes, I'm really annoyed by your correspondents who make such condescending remarks about the Z80; trivial programs; and of course, machine-code programming.

Mr. Randal wants some amazing system I can't afford, but isn't going to do the work; he would help, of course. Does he have a machine at all? He's not saying.

Mr. Trice doesn't approve of "frivolous" applications, such as might cause enjoyment for the owner of the system. No, he wants computers to be for the drudgery of repetitive calculations. For a modest outlay he could have a Sinclair Programmable. (I was near to saying TI57, but that would have caused bowls from their user group, and rightly so!)

Those of us who did what you suggested, and made 1978 the year we bought a computer, know that no program is ever really trivial; we also know that games programs (chess, for instance) are far more complex than stuff like accounting systems, or even trig. routines.

I use a Nascom. It's a good machine, and when the expansion arrives it will be even better. I haven't used Basic, but I know the useful pages of the Z80 programming manual by heart. As a computer, the Z80 is pretty nifty, especially in its ease of use — for decimal calculations. Floating point multiplication on numbers of up to 46 digits was easy to write (more digits if you want them, but I thought one line of screen per argument would do) in machine code. Conversely, a Star Wars Death Star trench simulation in under 1K is quite tricky. Which, we should ask, is the more worthwhile program?

Chris Blackmore,
31 Herne Rise, Ilminster, Somerset, TA19 0HH.

PCW *Intel, Zilog, Motorola and other companies deserve our thanks for being in the forefront of the microprocessor revolution. And they are bringing out more advanced products* **PCW**

Logging it in

I agree with Mr. Rodwell (Letters, Jan) that the substitution of 2.3 for LOG(10) in my coding to align decimal points will give faster execution. I should have put in my documentation that I usually calculate this once LET A = LOG(10) and then use A in place of LOG(10) in the coding.

However, I hope no-one attempts to use Mr. Rodwell's coding alone, unless they are absolutely certain that zero will NEVER be printed, as attempts to evaluate LOG(10) should cause argument errors. To sum up, the published coding:

```
100 IF ABS(N) < THEN 130
110 PRINT TAB (T—INT (LOG (ABS (N))/LOG(10)); N
120 GO TO 140
130 PRINT TAB T; N
```

may be modified by:

a) replace LOG(10) by 2.3 in line 110 or;
b) add 90 LET A = LOG(10) and replace LOG(10) by A in line 110

Note: The calculation in line 90 need only occur once in the entire program or;

c) if positive numbers only will occur, ABS may be omitted from line 110 or;
d) if numbers in the range $-1 < N < 1$ NEVER occur, line 110 can be used alone — without ABS if numbers are always positive.

Sydney A Leleux,
South East Derbyshire College, Field Road, Ilkeston, DE7 5RS

Radiant Rubbish

I tried to program my TI57 with the program on page 57 of the March issue "Turning the Tables" but all that happened was flashing rubbish.

I followed the steps carefully and found an error towards the end of the keying sequence, where you have printed 12; +; =; INT; STO1; INV SPR, I inserted a '1' between + & = and the program runs perfectly.

I hope this may help others who had problems with this.

J.T. Rowley,
76 Kidderminster Road, Brunsgrove, Worcs.

A rich hunting ground for articles.

May I, as a newcomer to the world of computing, say how much I am enjoying this new (hobby at the moment) interest.

I look forward to each issue of your magazine, although many items are still above my head, or of no interest due to their specialization.

My main reason for writing to you is to give you an idea of some of the types of articles I, as a beginner, would like to see in your magazine. After discussions with other friends who are also new to the field, it seems that we are in general agreement about the difficulty of finding certain techniques clarified.

Whereas some beginner articles such as you publish are good, they tend to be brief and limited, very suitable for the 'earliest beginner'. Then there is a gap, with the next stage seeming to cover much more complex programs or hardware ideas.

I know that it is probably true that you can only publish what is submitted to you, but perhaps if there is evidence of what is needed, you could ask some of your 'regulars' to provide manuscripts.

To this end, perhaps you could include a 'survey' in a future issue to see how your readers divide.

To give you an idea of the sort of problems that have puzzled me (or still are), and which could be subjects for articles:—

Software

1) A brief resume or summary of what each step is doing where this is not obvious — certainly not needed on a 100% basis — could be restricted to short programs or those for minimum (beginners) systems.
2) A series explaining how monitor software works — I have NASBUG T2 listing for my NASCOM but I cannot at present make full use of the subroutines avaiable as I am unable to see the 'wood for the trees'.
3) I am at present trying to write simple 'Games' software for my NASCOM. I have not yet found out (probably tied up with 2, above) how to interact with a running program, in order to, shall we say, 'wipe out' targets on screen. An article or series, on methods of changing displays, programs, etc in execution, would probably be of considerable interest.
4) Where programs listings are given using routines, commands or functions that are not common, could the authors be asked, where practicable, to give alternative commands, so that the program may be 'translated' to less powerful versions of the language.
5) General articles on 'Basic' are usually rather limited in their treatment of some of the more specialised commands. It would be nice to see a series where a particular command

(function etc.) is given a more extensive treatment. This is obviously unnecessary with elementary commands etc, but I have in mind items such as:

Arrays, Matrices, Print and all its variations, Use of Strings, Complex loops, when to use INPUT, DATA statements – etc.

Hardware

Fairly well done but perhaps a few articles on logic design, Memories, Registers, Flip Flops, Multiplexers, I/O's etc. would help overcome some of the problems associated with the understanding of it all.

I hope the foregoing will be taken in the spirit in which it is offered. I wish your magazine all success in the future.

C. Bowden.
Tregwyn, Stithlans, Truro, Cornwall.

Fuzz Buzz?

Re: Buzzwords, Feb 1979. Peter Reynold's attempt to reduce confusion over references to thousands of bits and bytes, although highlighting a serious problem area, has in fact made matters worse. The suggested abbreviation of 4k8 for '4,000 bytes each of 8 bits' is regrettable since 4k8 also has the standard connotation of 4,800!

Writers at present crave for abbreviations and readers have to compile a dynamic glossary of abbreviations. Perhaps in this instance just writing bits or bytes would suffice, leaving a mandate on the writer to explain himself if his bytes are not 8 bits. Because of differing word lengths, in general, quoting of store size in words leads to confustion, often because the number may be interpreted as bytes, or because the input device (ie reader) has not been programmed with the word length.

How about this for those who must abbreviate:—

k = 1000 b - bits
K = 1024 B - bytes (of 8 bits by default)

Martin W. Baker
17 Cockshot Road, Reigate, Surrey RH2 7HD

Micro Musings

I must thank Mr. Roberts for his analysis of my Mk 14 experiment (PCW April 79). The OF2F instruction is marked 'speed' in D. Johnson-Davies' program, and otherwise unexplained in the early version of the training manual, hence my (stated) difficulty in tracing the exact route of an empirical result.

I have written some more advanced poems for the Mk 14, and utilize in these another mysterious instruction shift, which generates equally interesting results. If these are published I would welcome Mr. Roberts' analytical extrapolation. It might even be the first time a poet enlists a critic to help him (the poet) to understand the complete meaning of the poem!

Eric Finlay.
c/o Chelsea School of Art, Manresa Road, Chelsea, London SW3

The Young Dimension

I am 14 and became interested in computers through a club at school, where, instead of having a machine of our own, we use some at Imperial College, part of a big computer network. After school, we hand-punch our programs to be taken to the centre and run. We don't use BASIC, because if you can take the extra trouble to learn it, FORTRAN has more functions and is not interactive.

Imperial College has a number of interesting subroutines, one being 00XX3D. (A good name for 3D noughts and crosses). They use the same method explained in Bill Davy's article on the game in the September 78 issue, (each vacant cell is allocated points, dependent on the number of noughts or crosses in the winning lines going through that cell), and although I think the idea is ingenious, the system of actually allocating the points is not so good. Instead of going through all 76 lines each time another move is made, why not just examine the lines going through the new entry? There are 13 possible directions of such lines, and it is quite easy to write a program to look at all directions and discard any which don't exist or contain less than four cells.

I have written another program to do all this, but haven't sent it because there's a snag – it's in FORTRAN. I don't know enough BASIC to be able to translate it, and there's plenty of food for thought for anyone interested in programming the game.

L.J. Barker,
26, Boniface Road, Ickenham, Uxbridge, Middlesex.

Bill Davy comments

Yes. This is a valid improvement of the given algorithm. However, it raises a number of interesting points. Firstly, in assembly language the program is so fast and short that speed improvements are scarcely necessary. Secondly, it is hoped that the program structure as presented is sufficiently clear that modifications and improvements (such as the one suggested) can be made easily. Finally, effort spent finding a better algorithm (eg tree search with alpha-beta pruning) would be more rewarding than improving the crude algorithm given in the article (which amounts to 1 ply and evaluation).

PCW *Bill Davy wrote to the editor: "Are they all this bright nowadays?"* **PCW**

A program naught can cross

In reply to the correspondence about the 3D Noughts and Crosses program published in the December '78 edition: I would like to add my ideas for the arrays NØ and N1.

N1		NØ	
C	V	C	V
ϕ	ϕ	ϕ	ϕ
1	0	1	0
2	1	2	3
3	49	3	16

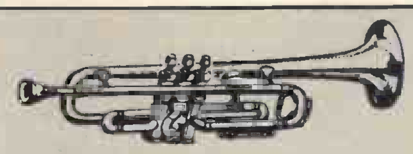
The original values were "aggressive" but were easily foxed. So the new values were defensive blocking every attempt at a line by the human, then eventually, 3 x's will be unnoticed and even if there is another point which would block 3 rows with 3 o's (3 x 16), it will make the move to make 4 x's. (49)

I wrote this program in machine-code for a MIKBUG compatible M6800 system which has only 1k of store (plus about 20 bytes remaining from the stack), luckily all the variables, constants and data are always less than 255 (and integers) so 1 byte suffices. After about 10 or 20 games I have not been able to beat it, the best results being a draw (after about ½ hour).

I then altered the program so that it reversed the o's and x's after each move and jumped over the input routine, thus the machine took on itself! The result being a draw after 11 seconds (this must be much faster than a similar BASIC program).

I would be glad to hear from anyone who manages to beat a machine with the values given. If anyone is interested in having a copy of the two programs send 50p and a SAE to my address.

B. Grainger
219 Kingsbury Road, Erdington, Birmingham, B24 8RD.



TRUMPET, VOLUNTARY

Responding to TIDBITS information or an advertisement?
Say you saw it in PCW
Blow your own trumpet.
Mention your magazine.

NewBear

Computing Store Ltd.

New - Newbear DB 80 Printer £595

Serial and parallel ports optional extras - 64 Key ASC11 Keyboard.
alternative character sets.

Introducing - Newbear 700

low cost VDU ex stock **£299**
V24 interface, upper case only keyboard and monitor available separately

Sym 1 6502 based micro from Synertek - fully assembled and tested
Down in price now only **£160**

Newbear now stocks O.K.Tools

Horizon (S100 bus Z80 based micros)
16K RAM with single disc drive **£1265**
32K RAM with double disc drive with 2 serial and 1 parallel port **£1983**

Schugart SA 400

5¼in disc drives down to **£190**

Now in stock BASF 5¼in disc drives £

Books

The Users Guide to North Star Basic R.Rogers £10.00 .75 pp

Basic with Business Applications Hayden £5.56 .50 pp

Basic Basic (An introduction to computer programming in Basic language) J.S.Coan £5.40 .50 pp

Practical Microcomputer Programming(6800) Weller £17.56 1.00 pp

Volume 0. The Beginners Book A.Osbourne £5.95 .50 pp

Volume 1.Basic Concepts A.Osbourne £5.95 .50 pp

Volume 11.Some Real Products A.Osbourne £18.95 1.00 pp

Volume 111.Some Real Support Devices A.Osbourne £11.95 1.00 pp

Understanding microcomputer and Small Computer Systems Scelbi £7.56 .50 pp

Book orders over £40 p&p free

Send for our complete book list

All prices subject to 8% VAT
Carriage by arrangement.
Barclaycard and Access welcome.
Minimum official order £10.

Callers welcome Mon - Sat 9.00 to 5.30 but please phone us first on 0635 30505

Visit our Northern office at
2A Gatley Rd.,
Cheadle, Cheshire.
Tel: 061-491 2290.

All Mail Order to Newbury.

24 HOUR DELIVERY IN MOST CASES

**Our new address is
Newbear Computing Ltd.
40 Bartholomew Street,
Newbury, Berks.**

Tid Bits PRODUCTS . . . . COMPANY NEWS . .

New Company
Micro Gems are a new company specialising in TRS-80 software. They offer an increasingly wide range of titles all of which, they state, will be of interest to PCW readers.
Further information from:
Micro Gems,
32 Buckingham Avenue, Hucknall, Nottingham NG15 8ET.

Staggeringly Dynamic
The first 64K bytes memory card for Cromemco systems has been introduced in the UK by MicroCentre, the Edinburgh Cromemco distributor.

Priced at £1026 the new card replaces four 16K byte cards previously costing up to £450 each, a saving of £774 for 64K bytes which "represents a staggering 43 per cent price drop".

Details from:
MicroCentre,
132 St. Stephen Street, Edinburgh EH3 5AA.
Tel: 031 225 2022 Telex: 72165 Ref: W582

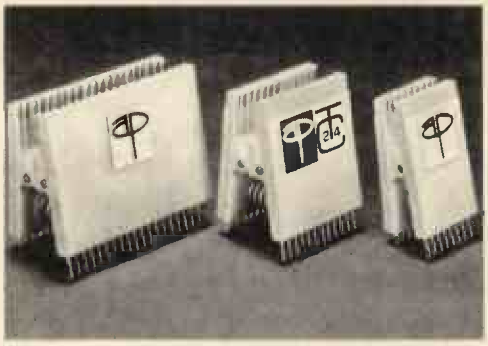
Rairing to go

RAIR have added the Decwriter IV to the range of DEC terminals they supply. The Decwriter is under £900, is a 300 baud desk-top terminal printing at burst speeds up to 45 cps, employing a 9 x 7 dot matrix to produce the complete 128 character ASCII set. It can be firmware configured from the keyboard or the host computer. Character width, vertical spacing, tab stops and left and right margins are all adjustable.
RAIR Ltd., is at:
30–32 Neal Street, London WC2H 9PS.
Tel: 01-836 4663

Giving Viewdata a Fillip
After four years of development, Philips Data Systems have announced a private Viewdata system designed specially for the business market. It is a new, inexpensive method of recording, storing and accessing business information, and is expected to make the benefits of electronic data handling available to everyone in user companies from Boardroom to shop-floor. The system is very comprehensive and is designed for medium to large companies.
Details from:
Adrian Wheeler, 01-589 3422

The MKB.01 miniature alphanumeric keyboard from Apex Microsystems is available in chassis or stand-alone versions. Measuring 165 x 93 x 14mm, it is "particularly suited to applications where size is vital", such as portable or mobile data entry systems, or as an alternative to conventional keyboards in small computer systems. Tactile response miniature keyswitches are spaced at 12.5mm centres.

Apex Microsystems,
27 Cowbridge Road, Pontyclun, Mid-Glamorgan.
Tel: Llantrisant (0443) 225578.



A Selection of Lektrokit's IC Test Clips for Dual-in-Line Packages. Lektrokit claims these clips offer a simple and inexpensive means of accessing any IC pin or lead.
Details from:
Lektrokit Ltd.,
Sutton Industrial Park, London Road, Earley, Reading, Berks RG6 1AZ

The New ADDS Regent 20, an "economy class VDU", has many standard options, claimed to be unusual in the dumb terminal market. It's a conversational device, with half duplex and local echo or full duplex and no echo working. The display is 24 x 80 on a 5 x 8 matrix with lower case descenders.
For full details contact:
Roger Crumpton, TDS Ltd., at (0254) 662244.

The Melody Eight door chime from Chromatronics, illustrating the bubble pack designed for the retail market. Price is £9.95.
Details obtainable from:
Robin Palmer, Chromatronics.
Tel: Harlow 418611

Small Systems, Big Ideas
Small Systems Engineering have brought off a coup. They are now marketing the famous Pickles and Trout P & T-488, which interfaces S100 bus computers to the IEEE-488 instrumentation bus; functions as a 488 controller, talker or listener; and has three software packages available: North Star DOS/BASIC interface; CP/M interface, and a custom systems interface package.
Small Systems are at:
62 New Cavendish Street, London W1.
Tel: 01-637 0777

CIS Cobol (Version 3) for North Star Systems is available from Microfocus. Supplied on two minifloppy discs, one of which contains the compiler and run-time system and the other utilities CONFIG and FORMS plus some demonstration programs.
Further details from:
Paul O'Grady, Micro Focus Ltd.,
58 Acacia Road, St. John's Wood, London NW8 6AG.
Tel: 01-722 8843 and 8847/8

Filling the Gap
Mutek, microprocessor systems specialists, have come forward with a system close to OSI Superboard capabilities by using readily available OSI products. The package consists of 1) The 500 CPU board with 4K user RAM, 8K Basic, several user ports 2) The 540 interface board with a 64 x 32 video display capability modified to European TV Standards; cassette interface and keyboard ports. 3) Encoded ASCII keyboards. 4) An 8-slot backplane assembly with 6 free slots for expansion.
Details from:
Dave Graham at The Studio, Quarrey Hill, Box, Corsam, Wiltshire SN14 9HT.

Basically Tiny
CC Soft have developed two versions of a Tiny BASIC for Nascom users. Level A is for use on a Minimum Nascom System. Level B is available on cassette for systems with additional RAM from address 1000 Hex; or in two 2708 PROMs for systems with PROM Sockets addressed from F000 Hex.
Details from:
M.B. Scutt, CC Soft,
83 Longfield Street, Southfields, London SW18.
Tel: 01-870 4891

The Thinker's Disc Drive
The Computhink Disc Drive offers up to 800K Bytes of on-line mass storage, is for the PET, and has "a wide and ever-extending range" of professional software such as a disc operating system, assembler, a linking loader, assembler-editor, Fortran Compiler, PLM compiler. Software under development includes an accounting package and a data base.
Details:
John Chew, Kingston Computers Ltd.,
Scarborough House, Scarborough Road, Bridlington.

The Winter of Content
Grama (Winter) Ltd., offers hardware systems tailored to requirements and a "complete business package" free with the purchase of a 32K–40K computer system. Programs are integrated, are twenty-four in number and functions are selected by number. As an example, program 9 is "examine Sales Ledger", and allows: a) list all sales b) monitor sales by stock code c) invoice search d) amend ledger files e) total all sales.
Further details from:
Tony Winter, 21B Dryden Chambers, 119 Oxford St. London W1

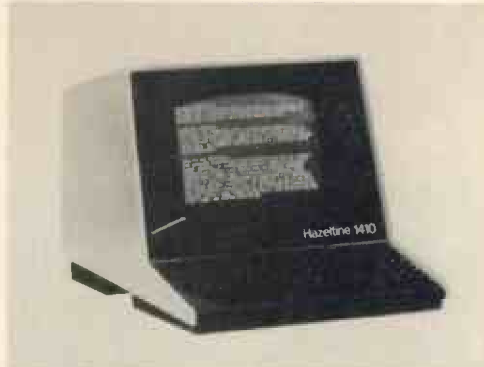
This Chip is UP
National Semiconductor, using an advanced process called XMOS, has developed an enhanced version of the industry-standard 8048 family of single chip eight bit microcomputers. Centre-piece of this new family is the (proprietary) INS 8050. It contains 4K ROM, 256 RAM and the CPU on the same Chip: National claims this is twice the capacity of any similar microcomputer chip now on the market, and that they are ideal for Original Equipment Manufacturers (OEMs).

The development by Petsoft — (micro software subsidiary of bureau group Applied Computer Techniques) — "of versatile, low price business programs", has made personal computers such as the Commodore PET shown here, an increasingly popular accounting tool for small businessmen. Programs range in price from £5 for a telephone charge monitoring program to £49.50 for a powerful Percentage Costing Package.
Details from:
ACT Ltd., Petsoft Division, 5/6 Vicarage Road, Edgbaston, Birmingham B15 3ES. Tel: 021 454 5348 Telex: 339396

Intel RAMpant
A new 16K Dynamic RAM from Intel runs from a single 5V supply, is pin-compatible with both current 16K devices (except for power supply) and future 64K devices. The new device is the 2118, housed in a 16 pin package, and is intended for use with systems requiring a 100 nsec access time memory. Power consumption is impressively low: 130mW operating, 15 mW on standby.
Details:
Intel (UK) Ltd., 4 Between Towns Road, Cowley, Oxford OX4 3NB.

This Hazeltine terminal, the H141D VDU, is designed to optimise real time operations. Features an ANSI standard bit paired keyboard with a separate integral numeric pad.

The microprocessor design of the H1410 is claimed to provide a low component count terminal giving reliability projections "several times greater" than those obtained with conventional designs.



The Hazeltine H1410 is supplied by RAIR, is available ex-stock at £590 (or on rent at £37 a month).
Contact: Howard Sales at RAIR Ltd.,
30–32 Neal Street, London, WC2H 9PS. Tel: 01-836 4663

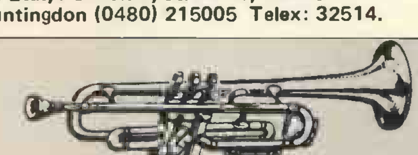
This oscilloscope is available from **Scopex Sales, Pixmore Avenue, Letchworth, SG6 1JJ. (Tel: Letchworth 72771).** *Price is* **£188** *exc. VAT.*

The 4D10B Dual Trace Oscilloscope features full XY operation and Z modulation, an accuracy of ±3%, DC-10 MHz band width and incorporates CMOS technology.

Go to Comart
Comart Ltd — specialists in S100 microcomputers — have launched two new standard configurations of the popular Cromemco Z80A based microcomputer system.

The Computer System 3/64 featuring dual 8" diskette drives, Z80A processor and 64K bytes of 4MHz memory and including console and printer interfaces with language support of Macro Assembler, Fortran IV, Extended Basic, COBOL and Multi User Basic is announced at a price of £4385.

The Computer System 2/64 offering mini-diskette drives and 64k bytes memory is listed at £3050.
More Information from:
Comart Ltd., PO Box 2, St. Neots, Cambs.
Tel: Huntingdon (0480) 215005 Telex: 32514.

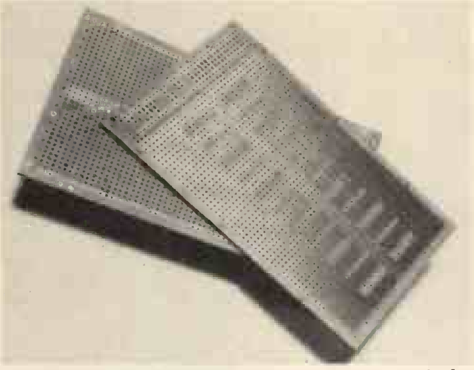


TRUMPET, VOLUNTARY
Responding to TIDBITS information or an advertisement?
Say you saw it in PCW
Blow your own trumpet.
Mention your magazine.

High Bred product from Vero: The Hybrid Board AB082

Newly available from Vero Electronics Limited, is this prototyping board.

Intended for general purpose R & D, the 'Hybrid' Board is available to single Eurocard dimensions, 100 x 160mm, and will accommodate up to twelve 14 or 16-way ICs or sockets and any discrete components required.



With provision, via a copper pattern at one end, for a 64-way connector to DIN 41612 (not supplied) the board is manufactured from 1,6mm epoxy glass.

Further details:
Vero Electronics Limited, Industrial Estate, Chandler's Ford, Eastleigh, Hampshire, SO5 3ZR. Tel: (042 15) 69911

The giants ITT and Centronics have pooled resources and know-how to market the latter's printers. Also distributing the printers are Sintrom Electronics, Cable & Wireless, Dacoll (Scotland) and Cara (Ireland).

Signing the Centronics/ITT contract are, from left to right: **Keith Williams**, *Divisional Manager, ITT Electronic Services;* **Chris Gill**, *Systems Manager, ITT Electronic Services;* **Mike Boyle**, *Distributor Sales Manager, Centronics Data Computer (UK) Limited.*

Further information from:
Marilyn Holmes. Telephone: Bracknell (0344) 54471

LOGIC MONITOR LM2

The Continental Specialties Corporation call this the "poorman's logic analyser". Price is £68.95. It monitors the logic high state of each of the 16 pins of a standard DIP IC through the lighting of a small LED.

Details from:
CSC, Shine Hill Industrial Estate, Saffron Walden, Essex CB11 3AQ Tel: 0799 21682

This illustration of Centronics' printers will appear on the front cover of a new brochure being produced and distributed by ITT Electronic Services.

Further information from:
Marilyn Holmes. Telephone: Bracknell (0344) 54471.

the Faces Behind the Places

Photographs *by Yoshi Imamura*

Peter Norman of Computer Centre. *He's a very competitive man — and his customers can thank that instinct in him, because it's reflected in his prices. It will be no surprise to* PCW *if Peter Norman becomes a major figure in the small computer world. In the photograph is Computer Centre's own system, of which Peter Norman is immensely proud. Address: 9 De la Beche Street, Swansea, West Glamorgan.*

Roland Perry *(standing),* **Nigel Day** *and* **Karen Duerden** *of* **Sintrom Microshop.** *While some of the flamboyant figures of the small computer world appear and disappear across the scene in nanoseconds, Roland Perry is hard at work building a solid image of dependability for Sintrom. The Vector MZ (see* PCW, *April 79) is its stock-in-trade. Sintrom is at 14, Arkwright Road, Reading, Berks RG2 0LS. Tel: (0734) 84322. Telex: 847395. Cables: Sintrom Reading.*

Barbara Hall of American Data, *the marketing company for Ohio Scientific Instruments. OSI is a very cost-conscious company, and the C2-4P that readers see advertised is proof of that. The Challenger 1P in the photograph is a packaged version of the OSI Superboard — a computer-on-a-board with outstanding specifications. The Superboard is not yet available in quantity in Britain*

David Bannister *and* **Vanessa Blackburn Kiddle of PCW.** *David claims to help the editor and Vanessa claims to manage them both. If you look very closely you can also see Phreakstein. And if you don't know* PCW*'s address by now you must be even more bemused than its editor.*

EXHIBITIONISM

LONDON

Microforum Europe '79 Wembley Conference Centre.
Contact: Business Equipment Trade Assoc. 109 Kingsway, WC2B 6PU.
JUNE 26 – 29

OFFICE '79 (Office and Business Systems Ex), Belle Vue, Magnum Exhibitions Ltd., 157 Station Road, East Oxted, Surrey RH8 0QF.
JUNE 11 – 14

The Great British Electronics Bazaar Alexandra Palace.
Contact: The Evan Steadman Communications Group 34-36 High St., Saffron Walden, Essex.
JUNE 28 – 30

1979 Microcomputer Show (Incorporating the DIY Computer Fair) Bloomsbury Centre Hotel.
Contact: Online Conferences Ltd., Cleveland Road, Uxbridge
JULY 5 – 7

International Word Processing Exhibition & Conference, Wembley Conference Centre.
Contact: Business Equipment Trade Assoc.
JULY 10 – 13

European Conference on Applied Information Technology
Contact: 13 Mansfield Street, London W1M 0BP.
SEPT 25 – 28

Personal Computer World Show, West Centre Hotel.
Contact: **PCW** Exhibitions, 62a Westbourne Grove, London W1.
NOV 1 – 3

Electronics '79 Show, Olympia.
Contact: Industrial Trade Fairs Ltd., Radcliffe House, Blenheim Court, Solihull, B91 2BG.
NOV 20 – 23

Breadboard '79 The Royal Horticultural Halls, Westminster, London SW1
Contact: Trident International Exhibitions, Abbey Mead House, 23a Plymouth Road, Tavistock, Devon PL19 8AN.

ELSEWHERE IN BRITAIN

Managing Microprocessors Holiday Inn, Langley, Berks.
Contact: Thames Valley Regional Management Centre, Wellington St., Slough.
MAY 30

Leeds Electronics Exhibition, Leeds University.
Contact: Dept. of Electrical & Electronic Engineerings, Leeds University LS2 9JT.
JULY 3 – 5

Electrical & Electronics Exhibition, Exhibition Centre Bristol.
Contact: Exhibitions for Industry Ltd., 157 Station Road East, Oxted, Surrey.
SEPT 10 – 13

THE CONTINENT & ELSEWHERE

Computer Hardware, Software, Systems & Services Exhibition.
Contact: SDL Exhibitions, 68 Fitzwilliams Sq., Dublin.
MAY 29 – 31

EDP/USA '79 Milan.
Contact: U.S. Dept. of Commerce, Room 1014 Washington DC 20230.
JUNE 12 – 15

International Microcomputers Minicomputers, Microprocessors Exhibition and Conference Geneva.
Kiver Communications S.A., 171/185 Ewell Road, Surbiton, Surrey.
JUNE 19 – 21

Computing Techniques, Business & Copying Machines Exhibition
Bucharest.
Contact: Publicom Publicity Agency, Bd. N. Balcescu 2Z, Bucharest.
JUNE 19 – 22

Microcomputers, Microprocessor '79 Exhibition, Geneva.
Contact: Kiver Communications S.A., 171/185 Ewell Road, Surbiton.
JUNE 19 – 21

Electro-Electronic Trade Fair, Sao Paulo
Contact: A.M.I. Ltd., 97/99, Park St., London W1.
JUNE 25 – JULY 1

Consumer Electronics Show, Sydney.
Contact: Riddel Exhibition Promotions Ltd., 166 Albert Road South, Melbourne 3205.
JUNE 28 – JULY 2

Euro Micro '79 Goteborg.
Contact: Swedish Trade Fair Foundation, P.O. Box 5222, S-40224, Goteborg.
AUG 28 – 30

INFO ASIA (Information Management Ex.) Tokyo. ECL (Ex. Agencies) Ltd., 11 Manchester Square, London W1M 5AB.
SEPT 5 – 8

OFFICE (Office Machinery & Equipment Ex.), Oslo Norway. Norges Varemesse, PO Box 130, Sloyen, Oslo 2.

International Exhibition of Computers & Peripheral Equipment, Munich
ECL (Exhibition Agencies Ltd.,), 11 Manchester Square, London W1.
SEPT 17 – 21

SICOB (International Office Equipment Exhibition), Paris.
French Trade Exhibitions, 54 Conduit Street, London, W1R 9SD.
SEPT 19 – 28

International Computer & Business Efficiency Exhibition, Stockholm.
Marketing Exhibitions Ltd., 113/123 Upper Richmond Road, London SW15
SEPT 27 – OCT 3

International Electrical/Electronics Conference & Show, Toronto.
Westbourne Marketing Services Ltd., Crown House, Morden, Surrey.
OCT 1 – 3

INTERBIRO – EDUCA (International Ex. of D.P., Office Equip., Teaching & Training Equip.), Zagreb, Yug.
ECL, 11 Manchester Sq., London, W1M 5AB.
OCT 15 – 20

Electronic Show Kowloon
Contact: HK Productivity Centre, Sincere Building 20/F 173 Des Vouex Road Central Hong Kong.
OCT 3 – 8

Eltro-Hobby '79 Stuttgart
Contact: G.E.S. (Overseas) Ltd., 181 Queen Victoria Street, London EC4
OCT 3 – 7

Electro Technical Trade Exhibition, Dortmund. Westfalenhalle GmbH Ausstellungen, Rheinlanddamm 200, 4600 Dortmund.
OCT 10 – 13

International Computers, Electrical Technology and Communications Exhibition, Sydney.
Convention & Exhibition Administration, 61a Hill Street, Roseville, New South Wales 2069.
OCT 16 – 19

Semicon (Japan); Tokyo,
Golden Gate Enterprises Inc., De Anza Office Center. 1307 So. Mary Ave., Suite 210, Sunnyvale; CA, USA
NOV 28 – 30

Electrical Technology & Professional Electronics – Finntec '79 Helsinki
Contact: ECL (Exhibition Agencies Ltd.,) 11 Manchester Sq., London W1.
NOV 6 – 10

International Electronics Exhibition
Brussels.
Contact: Brussels International Trade Fair, Palais du Centenaire, Parc des Expositions B-1020 Brussels.
NOV 26 – DEC 1

THE UNITED STATES

Summer Consumer Electronics Show, McCormick Place, Chicago, IL.
Contact: Consumer Electronics Two Illinois Center, Suit 1607, 233 N. Michigan Ave., Chicago, IL 60601.
JUNE 3 – 6

Semicon East, Boston, Mass. Golden Gate Enterprises Inc., De Anza Office, 1307 So. Mary Ave., Suite 210, Sunnyvale, CA 94087
SEPT 18 – 20

COMPUSIGN (Components, Equipment & Materials for Computers Ex.) Anaheim, CA. Golden Gate Enterprises, Inc., De Anza Office Center, 1307 So. Mary Ave., Suite 210, Sunnyvale, CA 94087.
OCT 9 – 11

INFO (Information & Management Systems Exhibition) New York.
ECL (Exhibition Agencies Ltd.,) 11 Manchester Square, London W1M 5AB.

National Computer Conference and Personal Computing Festival, New York City, NY.
Contact: AFIPS, 210 Summit Ave., Montvale, NY 07645. (021) 391-9810
JUNE 4 – 7

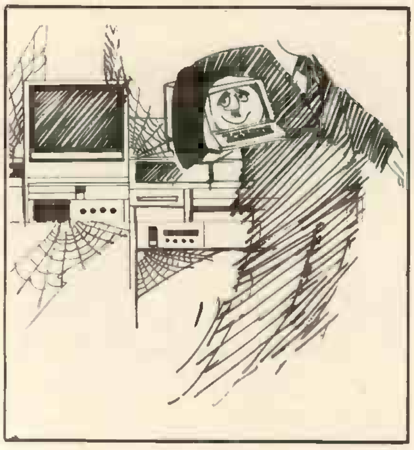
Atlantic Coast Microcomputer Show Deauville Hotel, Miami Beach, FL.
Contact: Bud Felsburg, Felsburg Associates, P.O. Box 735, Bowie, MD 20715. (301) 262-0305.
JULY 13 – 15

International Minicomputer Exposition, Dallas Convention Center, Dallas, TX.
Contact: I.M.E., 413 Carilon Tower, 13601 Preston Road, Dallas, TX 75240 (214) 271-9311.
SEPT 6 – 9

Personal and Small Business Computer Show, New York, NY.
Contact: Ralph Ianuzzi, H.A. Bruno and Associates Inc., 78 E. 56th St., New York, NY 10022.
SEPT 13 – 16

Personal Computing '79, Philadelphia Civic Center, Philadelphia, P.A.
Contact: John Dilks, Rte. 1, Box 242 242, Mays Landing, NJ 08330.
OCT 5 – 7

5th West Coast Computer Faire, (tentative), Los Angeles Convention Center, Los Angeles, CA.
Contact: Jim Warren, The Computer Faire, P.O. Box 1579, Palo Alto, CA 94302.
NOV 2 – 4.



COMPUTER LANGUAGES

The Multilingual Machine

G. J. Marshall
(Department of Electronic and Communications Engineering,
The Polytechnic of North London)

The developments and improvements that have occurred during the evolution of programming languages can be interpreted as responses to the needs of the programmer. Increasingly sophisticated languages have provided him with much needed tools and techniques for the solution of ever more difficult and complex problems. The aims of this article are to describe the development of computer languages, to give the flavour of some of them and to examine the effects of language developments on personal computing.

Introduction

The process of solving a problem with the aid of a computer can be divided into three parts, they are:-

i) specify the problem,
ii) find a method to solve the problem, and
iii) communicate the method to the computer.

The first step seems obvious. However, on occasion, superb programs have been written to provide the solution to a problem which, on further investigation, turned out to be the wrong problem. The second step is to find or devise a suitable algorithm. In a business environment, the first two steps are, essentially, what is known as systems analysis. Step three relates to the subject of this article, for the way to communicate a solution method to a computer is to express it in a language that the computer can understand. This is, in essence, what a computer programmer does (he may, of course, carry out the work involved in steps one and two as well).

The aims of this article are, then, to review the development of computer languages, to illustrate features of some of the languages and their applications, and, finally, to examine the impact of computer languages on personal computing.

From Machine Code to High-Level Languages

All computers, whether large mainframe computers or microcomputers, can only store, and operate on, binary digits. Data and instructions are stored and understood by the computer as patterns of binary digits.

Also, each computer has its own set of instructions, with one instruction for each action that the computer can perform. Thus, a computer, fundamentally, can only obey an instruction that belongs to its instruction set, and it can only understand an instruction when it is expressed as a binary pattern. Another way of expressing the same ideas is to say that a computer can only understand its own machine code. A segment of a machine code program for an *eight-bit* computer is listed as Program 1. *(See p. 22)*

It is clear that, while machine code is suitable for computers, it is far from ideal for people. We will return to the machine code program to consider its meaning shortly.

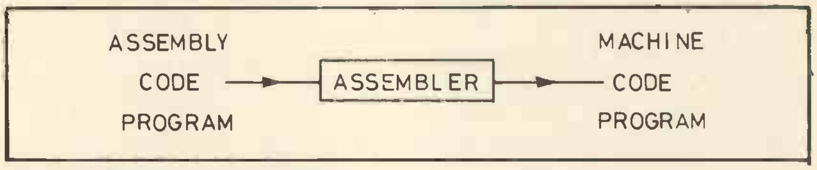


Fig. 1. Action of an assembler.

```
00001100
00101111
00100111
10110100
```

Program 1. Machine code program.

Programmers who had to write machine code programs naturally found it impossible to develop their programs from scratch as sequences of binary patterns, since mistakes were easy to make but hard to detect. They soon developed the habit of assigning each instruction a short mnemonic, such as LDA for LoaD a number into the Accumulator and STA for Store the contents of the Accumulator. Only at the final stage of loading the programs into the machine were the mnemonics replaced by their binary codes. In due course, it was realised that this coding chore could be undertaken by the computer itself and so assembly codes originated. A short assembly code program is given as Program 2.

```
LDA  12
ADD  15
ADD  7
STA  20
```

Program 2. Assembly code program.

This program finds the sum of three numbers, previously stored in the locations with addresses 12, 15 and 7, by loading the contents of location 12 into the accumulator, adding the contents of location 15 to it and adding the contents of location 7 to that: the sum is stored in location 20 so that it is preserved and the accumulator is available for other computations. The program has a degree of generality, for it can compute the sum of any three numbers that are stored in locations 12, 15 and 7.

The assembly code instructions of Program 2 are *each* equivalent to *one* machine code instruction. This is true of most of the instructions in any assembly code, although facilities available with some assembly codes, most notably macro processors, make certain instructions equivalent to several machine code instructions. Assembly code instructions cannot be executed directly, but must first be translated to machine code. This translation is performed by a special program called an *assembler:* its operation is represented diagrammatically in Figure 1. To illustrate the basic principles of operation of an assembler, Program 2 can be assembled manually to provide the equivalent machine code program with the aid of the code in Table 1. To translate the instruction LDA 12 to machine code the mnemonic LDA is replaced by its code 000 and the decimal address 12 is replaced by its (five digit) binary equivalent 01100 to give the *eight* (binary) digit pattern *000*01100. The same process applied to the other instructions converts Program 2 to Program 1.

mnemonic	code
LDA	000
ADD	001
SUB	010
STA	011
JUN	100
JEQ	101
EOR	110
STOP	111

Table 1. Three digit code for eight operations.

Trade Off

When machine code instructions are represented by patterns of eight binary digits, there is a trade-off between the number of operations that can be represented and the number of locations that can be addressed. In the example just discussed, assigning three digits for the operation code allows $2^3 = 8$ operations and $2^5 = 32$ different addresses to be represented. If two digit operation codes are allocated then only $2^2 = 4$ operations can be represented (see, for example, Table 2) but $2^6 = 64$ addresses can be accessed. It should be clear that representing instructions by patterns of eight binary digits imposes considerable restrictions. These restrictions do not apply to computers with longer word lengths. It is probably necessary to use *two* (8-bit) words to represent instructions in an eight-bit machine.

mnemonic	*code*
LDA	00
ADD	01
SUB	10
STA	11

Table 2. Two digit code for four operations.

A typical instruction set, that of the MOS Technology 6502 microprocessor (which is used in KIM and PET), is listed in Table 3. It is worth stressing that only two of the 56 instructions are arithmetic, implying quite clearly that the microprocessor is not primarily intended for 'number crunching'.

Arithmetic: ADC, SBC
Logical: AND, EOR, ORA
Load and Store: LDA, LDX, LDY, STA, STX, STY
Jump and Branch: BCC, BCS, BEQ, BMI, BNE, BPL, BVC, BVS, JMP, JSR, RTS
Increment and Decrement: DEC, DEX, DEY, INC, INX, INY
Transfer: TAX, TAY, TSX, TXA, TXS, TYA
Shift and Rotate: ASL, LSR, ROL, ROR
Compare: CMP, CPX, CPY
and fifteen others, including NOP

Table 3. Instruction set of the 6502 microprocessor.

Auto Codes

The next development in computer languages was the emergence of autocodes. The early, relatively primitive, autocodes have instructions of the form

A = A + B
X = A + C

These two instructions cause the values of the variables A, B, and C to be added and assigned to the variable X. The introduction of variable names relieves the programmer of address management chores by passing this task to the translation program. Primitive autocode instructions of this type are equivalent to three assembly code instructions — a load instruction and a store instruction — so that the programmer now handles a larger amount of computation per instruction than with an assembly code.

However, the task of adding three numbers still cannot be achieved with a single instruction. More sophisticated autocodes were developed, including Mercury Autocode, Extended Mercury Autocode and Atlas Autocode. Each of these languages is specific to a particular machine, but the latter, at least, deserves to be classified as a high-level language, in the sense that its instructions can describe a large amount of computation, being, in appearance, very similar to those of Algol. The development of autocodes overlaps the emergence of the so-called high-level languages. The most widely used ones, in scientific applications, are FORTRAN, Basic and Algol 60. In each of these languages three numbers can be added together by executing a single instruction. The instructions are:

in FORTRAN:- X = A + B + C
in Basic:- LET X = A + B + C
and in Algol:- x : = a + b + c

The trend of these developments was to throw an increasing amount of the mundane management chores onto the translation program, thereby relieving the programmer and absolving him from the need to understand the detailed workings of the computer he was using. Also, single instructions

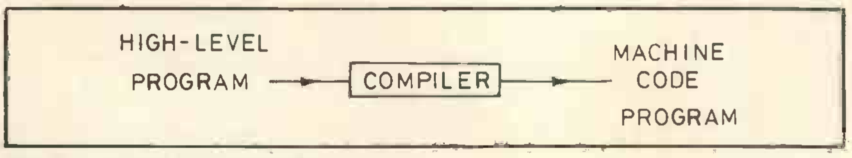


Fig. 2. Action of a compiler.

describe increasingly large amounts of computation, implying that they are equivalent to increasing numbers of machine code instructions. This gives the programmer a better chance of comprehending his entire program; in a phrase, of seeing the wood rather than the trees.

The translator for a high-level language program is called a *compiler.* Its operation is represented diagrammatically in Figure 2. Clearly, a compiler performs a much more complex task than an assembler. A compiler translates a complete high-level language program to a complete machine code program *prior* to its execution. A program to translate a high-level language for execution one line at a time is called an *interpreter.* Although the use of an interpreter increases program execution times, it makes possible interactive working.

High-level languages

FORTRAN was the first of the high-level languages, emerging from IBM in 1957. Algol 60 was formally defined by a report dated 1960, and implementations followed that date. These two languages were intended for scientific and engineering applications. Basic, which was intended as a teaching language that would be easy to learn and to use, was devised at Dartmouth College, USA, in the early 1960s. Meanwhile, COBOL had arrived as a high-level language for business applications.

The widespread use of high-level languages, and, of course, their compilers, gave rise to the 'virtual machine concept'. Broadly, this is the idea that a computer can seem to understand FORTRAN (or, indeed, any other language) when the appropriate compiler is stored in its memory. Programs written in a high-level language can then be automatically translated to machine code and run by the computer in a way that need not concern the high-level language programmer at all. Thus, by using the stored program facility to store a translation program, a computer appears to be something it really is not.

These languages, while by no means the only ones, are typical of the early high-level languages. The remainder of this section is devoted to an account and comparison of some of their features.

The features to be considered are:-

i) *Input and output*

Input, in the form of a value for a variable, A, can be obtained by the following instructions in FORTRAN, Basic and Algol, respectively:

```
    READ (1,100) A     (Fortran)
100 FORMAT (F10.2)

    INPUT A      (Basic)
    a: = read    (Algol)
```

The FORTRAN instruction reads a number from the input device assigned to channel 1. The 100 refers to the format which specifies the form of the number and its precise location, on, for example, a data card if the input device is a card reader. The Basic instruction is for interactive input from a keyboard and is obviously much easier to use. The Algol instruction is from ICL Algol and is probably unique to that implementation. The one language feature not defined in the Algol 60 report was input/output, so that the form of these instructions is left to the discretion of the language implementer and almost certainly varies from implementation to implementation. Output instruction in these languages resemble input instructions and illustrations are given below.

ii) *Conditionals*

The forms of the conditional instructions in the respective languages are illustrated by:

IF (A.GT.2.5) S = A + B . . . Fortran
IF A > 2.5 THEN S = A + B . . . Basic
if a > 2.5 *then* s: = a + b *else* s: = a – b . . . Algol

The instructions all have the general form:
'IF condition THEN conditional instruction', but with variations. During program execution, the conditional instruction is executed only when the condition is true. The FORTRAN instruction is clumsy and the use of the same symbol as a separator and a decimal point is confusing. The Basic form is more readable and the inclusion of THEN is very helpful. However, the Algol instruction with the 'else' at the end is much more powerful.

iii) *Repetition*

One of the strengths of the computer is the ease with which it can perform actions repetitively. For this reason languages must contain features that facilitate the handling of repetition. The facilities for repetition in the various languages are illustrated by short programs to print out the integers from 1 to 20. These programs also illustrate the output instructions.

```
    DO 50 I = 1,20
    WRITE (6,101) I
101 FORMAT (I3)
 50 CONTINUE      Fortran
```

```
FOR I = 1 TO 20
PRINT I
NEXT I     Basic
```

for i: = 1 *step* 1 *until* 20 *do*
begin print (i, 2, 0) *end* Algol

The shorthand for repetition in Fortran and Basic is similar, although the need in FORTRAN to identify the end of the loop with a number can be irksome. The arrangement in Algol is perfectly explicit and easily readable.

iv) *Data structures*

Arrays are almost the only facilities provided for structuring data in the early high-level scientific languages. These structures are well suited for handling vectors and matrices. The instructions in the respective languages for reserving storage space for a one-dimensional array are:-

DIMENSION V(10) Fortran
DIM V(10) Basic
real array v [1:10] Algol

The FORTRAN instruction reserves ten locations for variables named V(1) to V(10), while, in most dialects, the Basic instruction reserves *eleven* locations for V(0) to V(10). The Algol instruction is explicit.

Cobol, which is intended for use in commercial applications, has instruc-

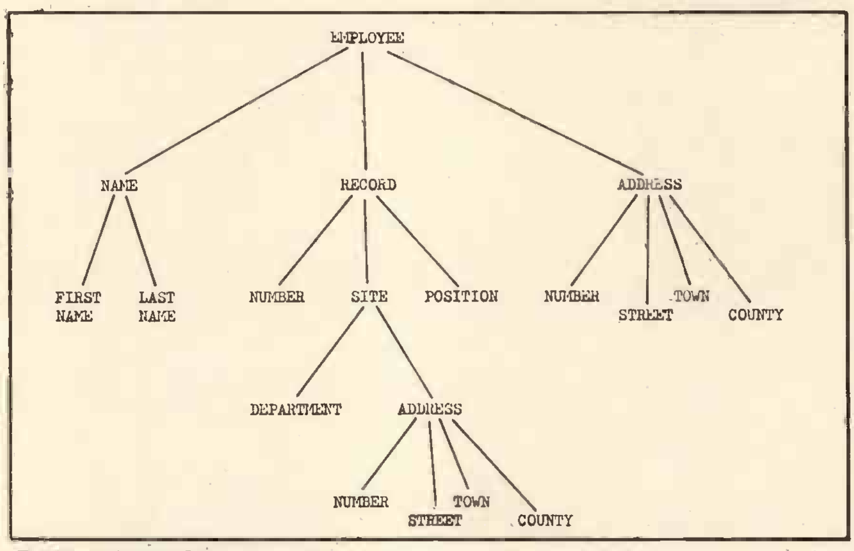


Fig. 3. Cobol Structure.

tions of a different appearance to those of the scientific languages. Only simple arithmetic facilities are necessary, the major facilities are provided for reading and updating file records and filling in forms. The language is intended to be readable. The Cobol instruction:

MOVE X TO Y

is equivalent to the FORTRAN instruction Y = X, while

ADD BALANCE TO OLDTOTAL GIVING NEWTOTAL is equivalent to NEWTOTAL = OLDTOTAL + BALANCE. Cobol has more sophisticated data structures than the scientific languages and can support structures such as the one illustrated in Figure 3. Selection of an item in such a structure is achieved by: LAST NAME IN NAME IN EMPLOYEE

Arrays of such structures can also be used, for example

EMPLOYEE (1:25).

Structured Programming and Languages for it

The next stage in language development was the emergence of languages to permit structured programming. The need for such languages emerged because programmers were experiencing intense difficulty in the development of long programs. The difficulties arose largely because language facilities were not available to permit the management of a natural flow of control in large programs, so that programmers were forced to use clumsy and unsuitable constructs.

The fundamental concepts of structured programming follow from the idea that the structure of a program should reflect the structure of the problem solution method. If programs can be written in this way they are automatically readable. *Thus, there is no real need to add comments to a program to make it readable.* More importantly, programs are made more easy to debug because the logical flow of the solution method is preserved directly by the program itself. Properly structured programs are also easier to develop and test because, as illustrated in Figure 4, they can be divided into modules, each of which has only one entry and exit point, that can be developed and tested independently.

The shortcomings of the early high-level languages arise from the fact that they do not provide features to make possible the writing of large, properly structured programs. The languages that are intended for structured programming include Pascal and Algol 68. PL/1 also has facilities for structured programming, although it was conceived as a general purpose language, combining the features of both FORTRAN and Cobol while showing the influence of Algol 60. Pascal was intended as a teaching language that would demonstrate programming as a systematic discipline and that could be efficiently implemented in a compact manner. The first operations compiler became available in 1970. Algol 68 was defined by a formal report dated 1968; it is a general purpose language developed from and updating Algol 60.

It is rather difficult to properly illustrate structured programming without examining long programs. However, the examples and programs discussed in the remainder of this section may give some flavour of it. The Basic program listed as Program 3 can be said to be a very bad one!

```
10 A = 1
20 B = 2
30 GOTO 60
40 PRINT C
50 STOP
60 C = A + B
70 GOTO 40
```

Program 3. A 'spaghetti' program.

A short examination shows that the same computational process is described by Program 4. Program 3 is a 'spaghetti' program that can be reduced to a linear program as in Program 4. (Imagine holding the ends of a piece of spaghetti and pulling until it untangles!)

```
10 A = 1
20 B = 2
30 C = A + B
40 PRINT C
50 STOP
```

Program 4. Linear program equivalent to Program 3.

Now, when programming in one of the early high-level languages it is not always possible to write a linear program to express a linear solution method. Typically, the GOTO construct has to be used in a manner that destroys the linearity. (This is equivalent to pulling the ends of the spaghetti and finding that a knot has been produced!)

To illustrate how programs in an appropriate language can reflect a solution method better than another language, we consider two problems and present programs for their solution in Pascal and in Basic. The first problem is:-

Read numbers from data until the first negative number is encountered. Then print out its position and stop.

The natural solution method is to repeatedly read numbers and count them until a negative number is read, and then to print out the value of the counter before stopping. This method can be expressed directly in Pascal as in Program 5.

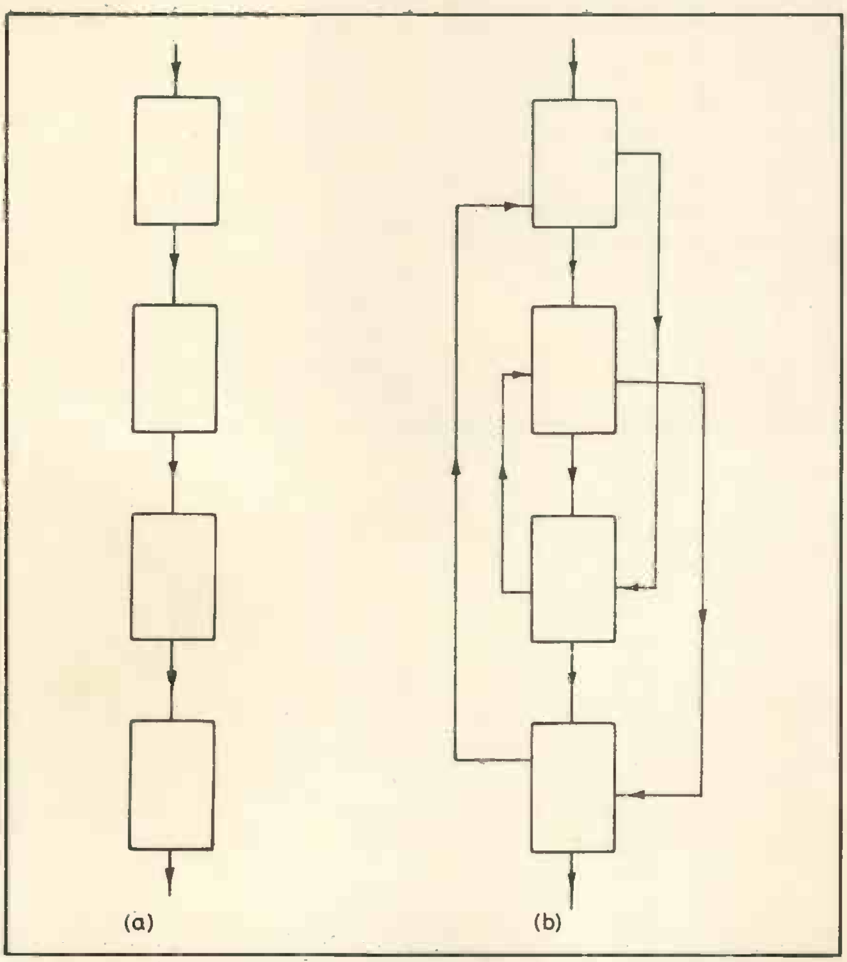


Fig. 4. Good (a) and bad (b) program structures.

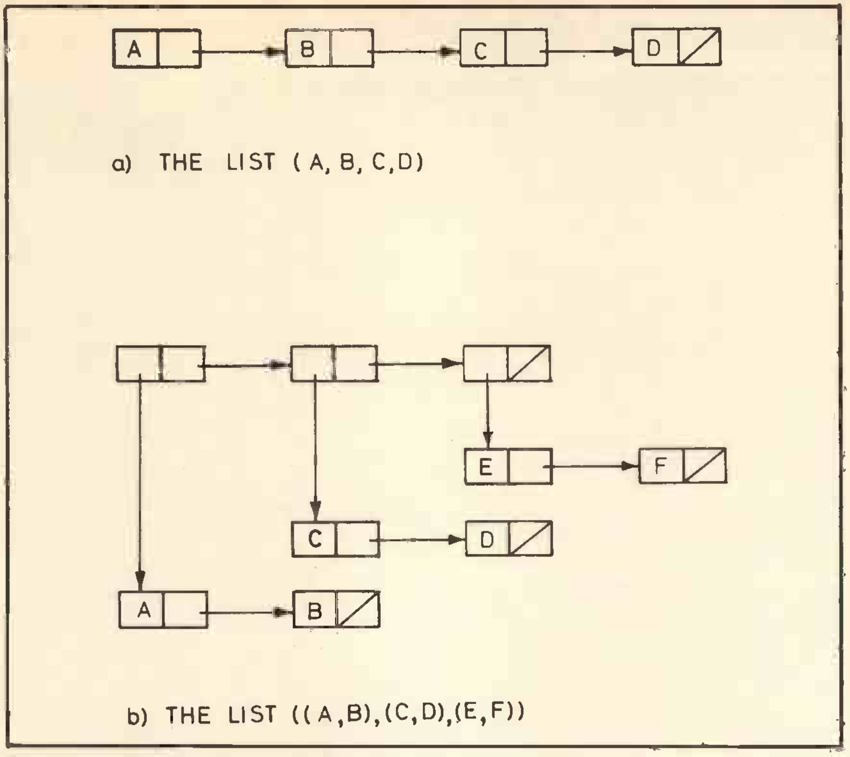


Fig. 5. Lists and their representations.

```
read    ff        ,     stay    ,        age    ;
if stay         = 1     then             af:=   0.5*ff
                        else if stay     <15 then af: = 0.7*ff
                                              else af: = ff
if age          <3      then fare: =     0
                        else  if age     <14 then fare: = 0.5*af
                                              else fare: = af
writeln (fare);
```

Program 7. Pascal program for problem 2.
(is the multiplication symbol)*

```
i: = 0
repeat read (n); i: = i + 1 until n <0
writeln (i)
```

Program 5. Pascal program for problem 1.

A Basic program to do the same thing is given as Program 6 and is a little contrived.

```
10 I = 0
20 INPUT N
30 I = I + 1
40 IF N <0 GOTO 60
50 GOTO 20
60 PRINT I
```

Program 6. Basic program for problem 1.

Now consider a second problem which concerns the calculation of rail fares:-

Reductions on the full rail fare for a journey are given according to both the length of stay and the age of the passenger as shown in the following tables:-

Length of stay (days)	Discount on full fare
1	50%
2 – 14	30%
15 or more	0

Age (years)	Actual Fare
under 3	free
3 – 13	half discounted fare
over 14	discounted fare

Given the full fare, length of stay and age of the passenger, compute the actual fare to be paid.

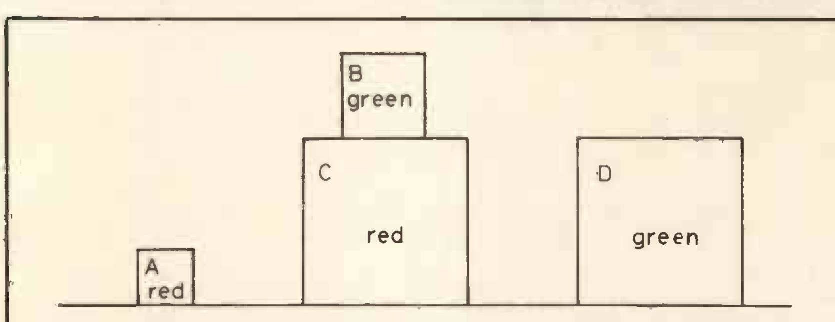


Fig. 6. The blocks world.

A Pascal program for this which captures the natural solution method is given as Program 7. The variables *ff, af* and *fare* are used to hold the values of the full fare, adult fare and the fare to be paid, respectively. It is left as an exercise for the reader to attempt the same thing in Basic or FORTRAN.

Special Purpose Languages

All the languages mentioned to this point are intended for either scientific or commercial application. However, other languages have been developed for particular special purposes. These include languages designed for manipulating particular data structures, most notably for manipulating strings and for processing lists. *Snobol* is the major language designed principally for string manipulation. It has facilities that include pattern matching and pattern replacement. A typical Snobol instruction is

```
STR "ED" : S(FOUND)
```

The effect of this instruction is to scan the string named STR for the pattern ED and if the search is successful to jump to the instruction labelled FOUND. The original string manipulation language is LISP. A typical LISP instruction is

```
((EQ (CAR LAT) A) (CDR LAT))
```

The applications of string manipulation and list processing include linguistic analysis, compiler writing, computer assisted instruction and artificial intelligence.

Two examples may show, in a limited way, how list processing can be used, but first the meaning of a list must be explained.

A list of the four elements A, B, C and D can be represented by (A, B, C, D) and the way in which it is stored is represented in Figure 5a. The figure shows that each element of the list is stored with a pointer to the next element, and this is the way that the elements of a list are linked. A more complex list is represented in Figure 5b. The first example is drawn from a system developed by Terry Winograd for the machine comprehension of natural language.

The system manipulates a representation of a set of coloured blocks, such as the one illustrated in Figure 6. The system can understand and obey instructions expressed in natural language such as "put the small red block on the large green block". We consider only how the machine can determine what is meant by "the small red block". Given a list of all the blocks, (A, B, C, D), a list of the red blocks, (A, C), a list of the green blocks, (B, D), a list of the small blocks, (A, B), and a list of the large blocks, (C, D), then the small red block must be a block that is in both the list of small blocks and the list of red blocks. Thus, the small red block is block A.

It is worth noting that *operations on sets* are being implemented by using lists, for if the list of blocks is regarded as a description of a set, the lists of blocks with a particular property are subsets and the problem of determining what is meant by the small red block becomes that of finding the intersection of two subsets.

The next example illustrates how lists and list processing can be of value in enabling a robot to automatically determine the actions necessary for it to achieve a specified objective. Consider a robot in the rather simple situation illustrated in Figure 7 which is set the task of filling the bucket. The situation in the diagram can be represented by a list (known as the world list) thus:-

((ROBOT, IN,ROOM 2) (TAP, IN, ROOM 2) (BUCKET, IN, ROOM 1) (BUCKET, EMPTY))

The situation on successful completion of the task is represented by this world list:-

((ROBOT, IN, ROOM 2) (TAP, IN, ROOM 2) (BUCKET, IN, ROOM 2) (BUCKET, FULL))

Now, each action that the robot can take alters the situation in some way and can be represented by a corresponding change in the world list. To achieve its task, the robot must determine an appropriate sequence of actions. This can be done by determining a valid sequence of changes to the initial world list which alters it to the world list representing the situation on completion of the task.

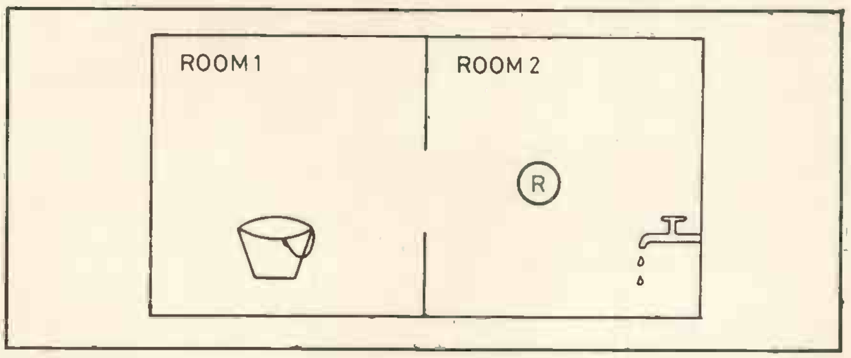


Fig. 7. A robot world.

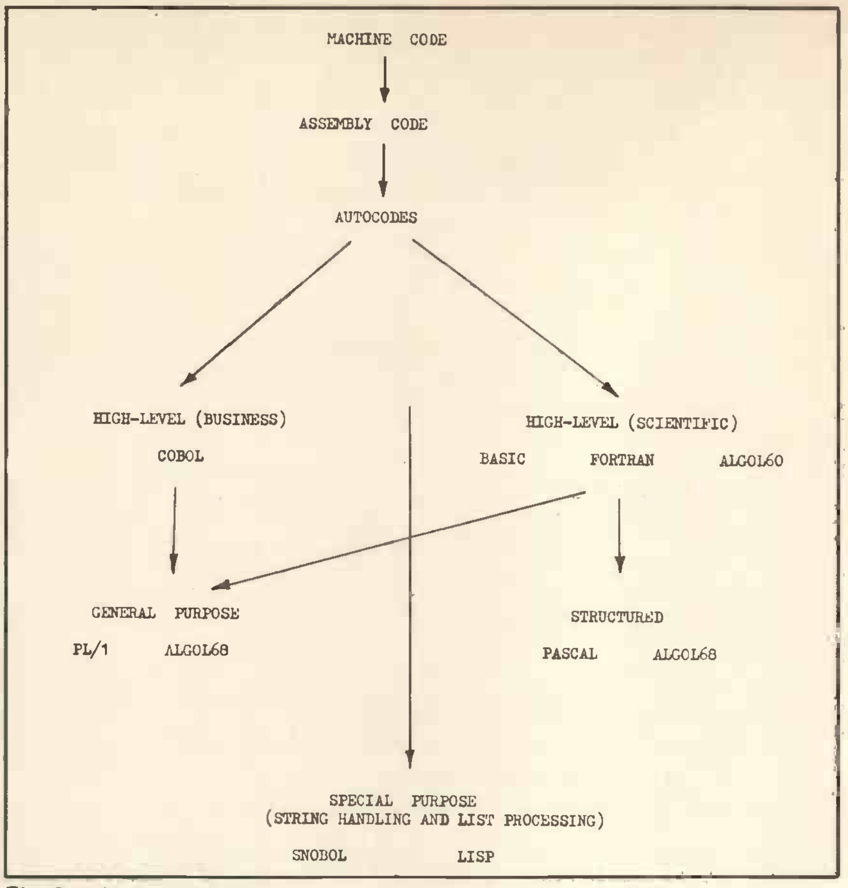


Fig. 8. Language tree.

Impact on Personal Computing

Figure 8 shows, in the form of a tree, the lines of development connecting the languages mentioned in this article. With regard to the languages that have made an impact on personal computing, assembly codes are not really in the running, since an assembly code is specific to a particular micro-processor making it difficult for one to achieve any general impact. The language that has made the greatest impact to date is Basic, and the most likely language to achieve general acceptance in the near future is Pascal.

It is interesting that both were designed as teaching languages. Their general acceptance probably stems from the fact that their principal aims fit the requirements of personal computing almost perfectly. In particular, Basic is easy to learn and Pascal is structured and capable of compact implementation. For both languages, interpreters seem to be more suitable than compilers for personal computing because the ability to interact with a program seems, in general, to be more desirable than attaining the shortest possible times for program execution.

As for the future, the possibility of natural language input, at least in a restricted form, has been touched on. Since a speech recognition unit is currently available for the Apple, perhaps programming will soon be possible by addressing the computer in almost ordinary English.

Bibliography

1. 'An introduction to the study of programming languages', D. W. Barron, Cambridge University Press, 1977. *2.* 'A comparative study of programming languages', B. Higman, Macdonald and Jane's, 1967. Both books give more details of the languages mentioned in this article and of many other besides. *3.* 'Pascal user manual and report', K. Jensen and N. Wirth, Springer-Verlag, 1975. The Pascal report and a tutorial style user manual. *4.* 'Illustrating Basic', D. Alcock, Cambridge University Press, 1977. An entertaining introduction to Basic. *5.* 'List processing', J. M. Foster, Macdonald and Jane's, 1967. A nice introduction to list processing and its applications. *6.* 'Artificial Intelligence and Natural Man', Margaret Boden, Harvester Press, 1977. A description of artificial intelligence work that stresses its human relevance and includes an account of Winograd's blocks program.

POLYTECHNICAL PROCESSING

Computer, Students, Teachers in Leicester Polytechnic

Philip Lawton[1].

The purpose of this article is threefold. Firstly to speculate about the future, secondly to describe the Leicester Polytechnic's present computers and show how they are used, thirdly to mention two favourite programs.

A Speculation

First a personal speculation about the future. A new computer laboratory consisting of 20 personal computers will be available with mainly BASIC, perhaps some PASCAL, FORTRAN, ALGOL, APL and other languages. It may be situated in the library, be open from 8 a.m. until 9 p.m. and be supervised by computer centre staff. Interactive computer programs with associated information for learning and calculation purposes will be available from the tutor on duty, these programs would be pre-recorded on cassettes, floppy discs or ROM's. Printers, graph plotters and recorders will be available to plug into any computer when required. In addition a few intelligent terminals will be on-line to a national/international network.

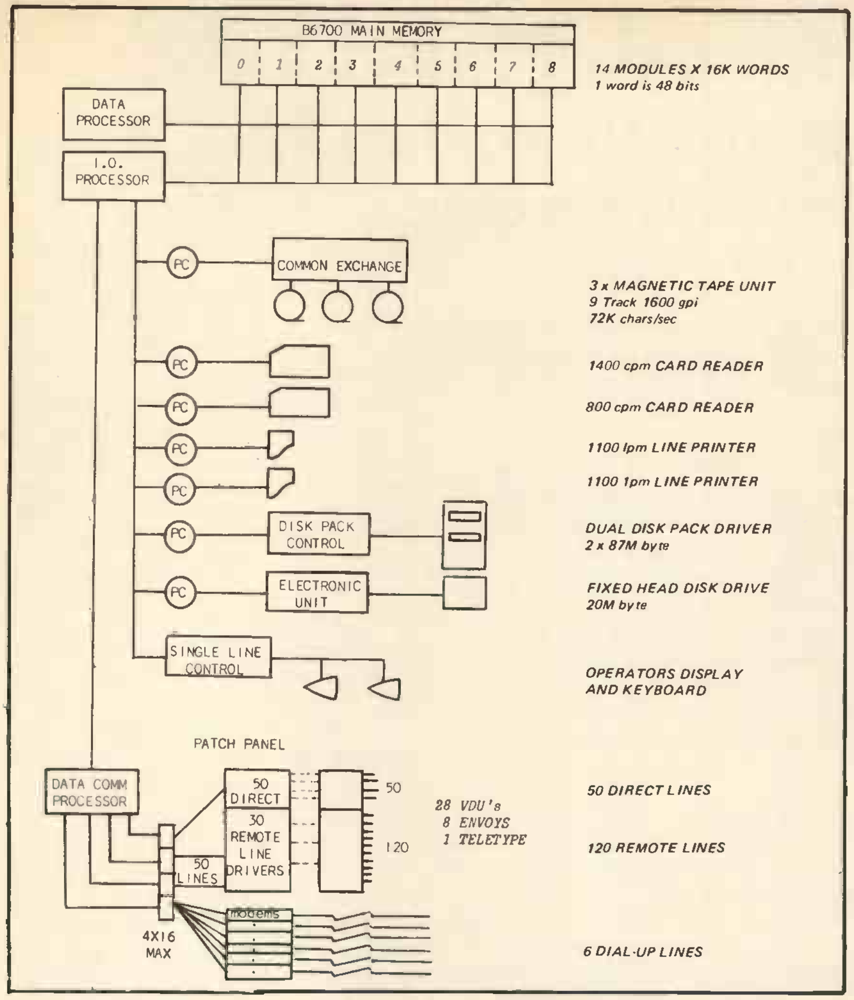
Personal computers will be available in the present laboratories for simulations and for processing the results of experiments. They will be available for use in lecture theatres with the present closed circuit television system, and for use at home.

Finally, equipment will be available to prepare demonstrations of the use of programs in problem solving; for example, 35mm camera, tape recorder, video recorder, data recorder, printer, plotter, along with the appropriate replay equipment.

The Present Set-Up

Now a description of our present computers. The central system, located in the Computer Centre, is based on a Burroughs B6700 computer with a memory of 224K words (1 word is 48 + 4 bits, 1 byte is 8 bits), 2 line printers, plus the usual collection of discs, tapes, card readers, modems and terminal lines. Connected to the computer are 28 visual display units (with edit facilities), 3 graphic terminals (2 storage, 1 refresh), 1 on-line digital graph plotter and numerous envoys and teletypes. From the user's view point the main features of the central system are that:— the visual display units (vdus) operate at 600 baud (60 characters a second) — the graphic terminals operate at 9600 baud — 6 languages are normally available — all the terminals are online to edit and modify programs –

Listen, Look, Learn?



The Central System

only 6 to 16 terminals share the data processor to run user programs at any time. In addition, a Honeywell H516 computer is available with:— a disc, graphics terminal, teleprinter, high quality graph plotter, high speed paper tape reader and punch.

The above equipment is distributed between a computer room, a terminal laboratory (16 vdus + 3 printers), a graphics laboratory, and various other rooms. It is available for general use between 10 a.m. and 9 p.m. and can be reserved by teachers for supervised student use.

In addition to the central system various Polytechnic Schools have their own computers; namely, Data General, Membrain, DEC, SWTPC's, PET's, and various microprocessor kits.

All these computers are used in two main ways, either to practice a programming language or to use an existing program. A disc based library of 250 programs for teaching and research purposes has been developed during the past 10 years. These programs are described in a software catalogue which is published by the information section of the computer centre[2]. In addition to the description in the catalogue, demonstrations of the on-line use of some of the interactive programs have been recorded by means of slide/tape, video cassette, and audio cassette where the left channel is used for data and the right channel for speech. The audio cassette is replayed via a vdu to a television monitor or direct to a television set as illustrated in the photograph.[3]

The Polytechnic was a member of the Engineering Science Project[4] of the National Development Programme in Computer Assisted Learning (1972–77)* along with many other Polytechnics and Universities. The results of this Development and the lists of programs available for transfer can be obtained from the Council of Educational Technology.[5]

Two favourites

Now a mention of two favourite programs. These are Interactive Simulation Language (ISL)[6] and Impedance and Complex Number Calculations (AC)[7].

ISL consists of 30 mathematical routines which can be interconnected in order to simulate dynamic problems[8]. The program is interactive and the results can be displayed graphically. The main routine is integration, other routines include summation, sine function, pulse, square waves, non-linear, unit delay, counters.

A favourite demonstration is illustrated in figures 1 and 2. Figure 1 shows a square wave synthesised from the first fifteen harmonics, whilst figure 2 shows an exponential decay synthesised from the resultants of the harmonics after each as "passed through" a capacitor-resistor circuit (or mathematically equivalent system)[9]. The ratio of periodic time to time constant is 10 to 1.

AC consists of 6 mathematical routines (add, subtract, divide, multiply, parallel, square root) involving complex numbers. These routines can be linked together using standard electrical equations in order to analyse simple electrical circuits. It is used as an introduction to computer aided circuit design.

ISL is introduced to students by means of a problem investigating exercise in a laboratory class whereas

(*See "The Learn Machine", PCW Vol. 1, No. 12)

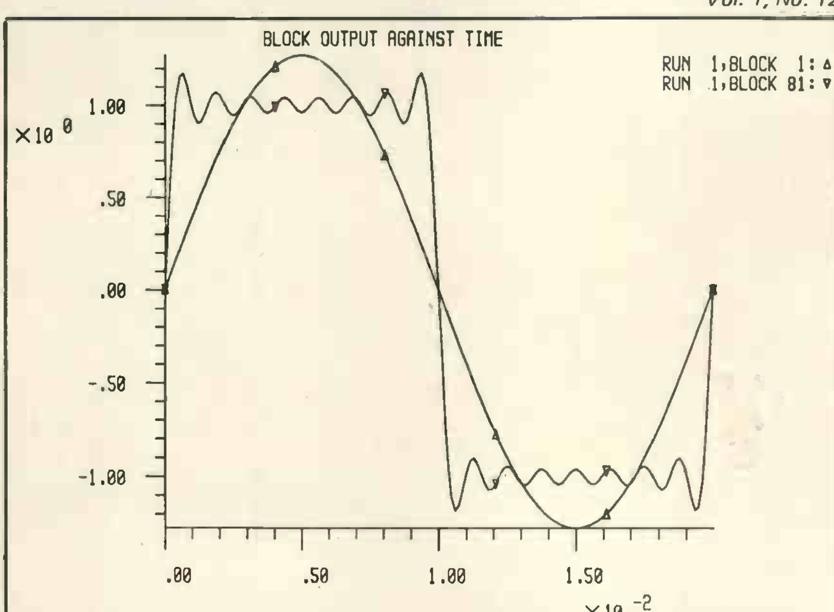


Figure 1. Square wave synthesised from first fifteen harmonics.

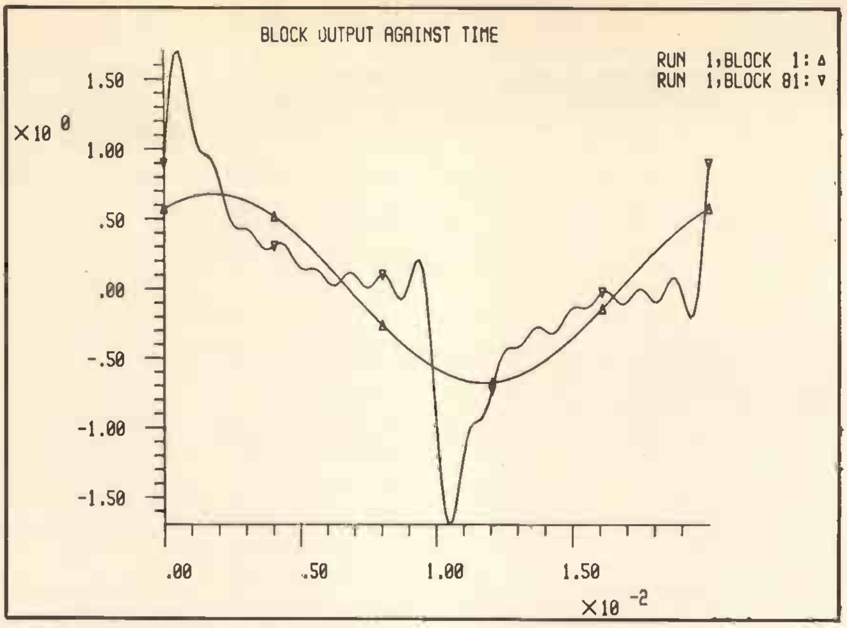


Figure 2. Exponential decay synthesised from first fifteen harmonics.

AC has been used to solve problems by a weekly tutorial class. In both cases the use of computer programs enables more experience and knowledge to be gained in the same time relative to the situation a few years ago.

In conclusion, this author eagerly supports the concept of the Mobile Computing Laboratory for Schools proposed by Mick Coleman in a PCW article[10] and which is already available in Pennsylvania[11]. One of the disappointments of the present situation is that local schools have not been able to take advantage of Computer Centre facilities because of the cost of a terminal and an acoustic coupler, and the telephone charges. But the Open University is doing an excellent job with its on-line computer system; also Prestel, Oracle & Ceefax are here to use, and given sufficient demand their cost might remain steady in spite of inflation.

REFERENCES

1. The author is a Senior Lecturer in the School of Electronic and Electrical Engineering at Leicester Polytechnic.
2. Software Catalogue September 1977, Software Catalogue Supplement 1 January 1978; Computer Centre, Leicester Polytechnic.
3. Recording of the PCW article "Basic, the First Steps" by Philip Couzens (Volume 1, Number 2) is illustrated on the screens shown in the photograph.
4. The Engineering Science Project was based at Queen Mary College, University of London and directed by Dr. Smith, of the Computer Assisted Teaching Unit. This is now the Engineering Science Program Exchange.
5. The Council of Educational Technology for the United Kingdom, 3 Devonshire Street, London W1N 2BA (01-636 4186) provides a programme of services for those interested in the application of computers in education and training.
6. Interactive Simulation Language developed by Dr. D.J. Holding, Queen Mary College, University of London.
7. Impedance and Complex Number Calculations developed by P.J. Lawton, Leicester Polytechnic.
8. PCW article titled "Cutting the World Down to Size" by Werner Sieber (Volume 1, Number 1).
9. *Wireless World* article entitled "A check on Fourier with some side lights on the controversy about phase shifts" by M.G. Scroggie (November 1977).
10. PCW article entitled "Do we want our schools to have a personal problem?" by Mick Coleman (September 1978).
11. Calnews 5 July 1976 (CET, see [5]), page 4, column 2. Department of Education at Pennsylvania State University, USA.

"Incredible sales figures — in December, January and February we were increasing turnover at about 100% a month."
"We're really serious about looking after people".
"We're the meat in the sandwich, between customer and manufacturer".

A PCW Conversation

THE EVERISS EXPRESS

The above quotes are from a PCW conversation with Bruce Everiss of Microdigital, Liverpool. The conversation ranged from his plans for expansion, to the market outlook, especially for retailing to the small business.

Bruce Everiss has a background in accountancy, a flair for promotion, and plenty of what advertisements for salesmen call "self-motivation". At first glance he seems young and diffident — but as he talks you can see how each phrase satisfies him with its aptness, to become the fuel for following phrases of increasing confidence and authority. Here is another sample of Everiss exposition: "We've been hiring new staff at the rate of one a fortnight. Microdigital has expanded to two suites of offices — we can give a red-hot service".

He seems particularly proud of his stock of books and the turnaround of his mail order service.

But Everiss, speaking from what he regards as a position of strength, doesn't skate over the problems. "We get a manufacturer's delivery date and then advertise in good faith, but there are delays sometimes. Then we get people wondering what's going on. We explain and inform. We are extremely patient because we are proud of our goodwill".

The Evangelist

Everiss is a great fan of the PASCAL language. In fact, while talking he was transformed from fan to evangelist. "PASCAL is totally portable. A PASCAL program written for the APPLE can be run on a super-machine like the CRAY with absolutely no trouble".

In most people's mouths that would sound like bombast, but it's plain that Everiss lives for what he believes in — the spread of microcomputers. In his opinion, Government is well intentioned but bumbling, with a reaction time to events which is slow because of built-in inertia. So Government can — against its will, perhaps — be insensitive. He thinks that programs like TV's "Blake Seven" are doing more to popularise computers and educate people about them than anything any Government agency does. Government support goes to the bigger companies. "It totally ignores retailers who are doing the groundwork — and taking the risks".

And as for Liverpool, with its University — there's no place like home. "Liverpool University's microcomputer laboratory is the leading centre for microcomputer education".

Later in the conversation, Everiss reverts to PASCAL. Already, he says, PASCAL is the second language of Apple, and may be put on the Sorcerer. Its time is now dawning and the Age of Pascal will be the Age of People's Programming. "Basic is an abomination. Basic is the language of Babel".

PCW person hastily lights another cigarette. PCW is full of BASIC.

And talking of PCW, Everiss says its layout is terrible. Its contents great.

Micro Business

When asked about the requirements for setting up in dealing in micros, Everiss says, "The technology is so new that a person's background is irrelevant . . . yesterday's window cleaner is today's authority" . . . If the man is intelligent enough, and interested enough, and can get proper backing.

Who's buying? "A lot of businesses, government departments, educational establishments and even multinational companies".

The small business market is there, all right — even the business owned and run by one person or partners. The potential *can* be realised, and people come to Microdigital looking for a job to be done; but Everiss says that small business applications need micros with at least one disk and file handling routines.

For him, a minimum configuration would be a system with a Disc Operating System of 10K and user memory of 32K. A printer would be needed, depending on the application. Microdigital is well placed to sell to the small business, as it's situated in the commercial centre of Liverpool — the equivalent of London's "square mile". And Microdigital's company secretary is also a senior partner in a firm of accountants. On his full-time staff there will be a commercial programmer, who's now looking at writing small business software for the Apple. The Apple, says Everiss, is "the premier machine in the market today". He cites the Trade Counter program for the Apple (at £25) as just one small example of the usefulness of the small computer in business. And people haven't yet looked closely at the practical possibilities of voice input/output, which the Apple has had for some time.

Everiss went on to describe Microdigital's set-up, in the process giving clues to what makes a successful micro retailing enterprise tick.

"This is a new industry. We find that the prime requirement for staff is intelligence rather than experience — in a totally new situation, what experience can be valid? If we took a top systems analyst and let him loose in this line of country, the man would be lost. We need intelligent, adaptable people, with no preconceptions. That is why no big company has succeeded in retailing personal and small business systems.

"The prime requirement is being fast on your feet — and a certain amount of prescience helps".

"I give my staff machines to take home at weekends. I ask them to put together kits like the NASCOM 1." You can't get hands-on experience sitting in an armchair and dozing over a book.

Professionalism

"I have no problems with staff discipline. I can do practically anything they can — that is, I can speak their language — so we're in touch. Everybody at Microdigital is so bright, anyhow. They need very little directing. Our staff find us, and not vice versa.

"I know it's easy to ramble on about customer goodwill and customer satisfaction — it looks good in print. But I believe in it. Our repair service has so far managed turnaround in a week, except", and Everiss winced, "when there's a shortage of a particular component.

"We're more professional than most. We don't believe in exploiting the keenness of an electronics hobbyist to do our repair work for us — the 'repair facilities' of some retailers are just a part-timer who's paid what his employers think he's worth; they get what they pay for; some don't pay very much. If I do anything I do it properly — that means using engineers in digital electronics; and they don't come cheap.

"We cheerfully hire out computers. A sale may be delayed while people are making friends with a computer, but we believe in the future. Then when someone buys a system, we tune it for him before he goes home. I'm not selling cars or fridges or take-away food. I don't regard people as consumers but as customers. Now I'm setting up a genuine technical consultancy service for really practical advice".

'Take the money and run' may be a cute caption for a cartoon, but it is a malevolent motto for microcomputer retailing.

The Categories

Everiss sees the market mainly in terms of its now classic categories: hobbyist, home computers, education and research, and data processing for the small business. He points out, however, that there is a vast market for small computers as adjuncts to bigger systems, in distributed data processing and scientific applications. Then, of course, there is proper electronics control using the dedicated microprocessor. There is a burglar alarm system by Chubb with a 6802 sitting in it. He is certain that the biggest single application affecting the public will be the microprocessor in cars. He has absolutely no doubt that ignition, fuel, readouts will all be controlled or implemented using the microprocessor.

He has one customer who's used the NASCOM to replace the hardwired logic in fruit machines, another using the SC/MP to control two separate machine tools. The micro is in some washing machines, radios and hifi, and will be in anything that can be controlled in discrete steps. It will be sitting there, a swift slave.

And Everiss more believes than hopes that the silicon slave will be exploited mercilessly to benefit *all* of us.

PCW Review

The MSI 6800 System One

Stuart Danton

Introduction

We live in a rapidly changing world. Only 30 or so years ago the computer entered our lives and yet now, in the High Street,you can buy a pocket calculator more powerful than the early computers and get change from ten pounds!

The computer today comes in all shapes and sizes, from the largest mainframe or D.A.P. to the most modest of micro-computers. The SYSTEM ONE is a microcomputer but it is by no means a modest one. It boasts an MC6800 microprocessor, up to 56K bytes of RAM and built-in backing storage provided by mini-floppy discs.

Even in its minimum configuration the SYSTEM ONE costs a bit more than the average British amateur would wish to pay for a personal computer; however, SYSTEM ONE falls naturally into a niche at the lower cost end of the professional market.

SYSTEM ONE is in fact a complete computer system available with single or dual disc drives and all the necessary support software. Although there are at present no standard application packages supplied specifically for the SYSTEM ONE, there is a wealth of such software that with the minimum of adaptation can be run on SYSTEM ONE to advantage due to two very good versions of BASIC available from S.E.E.D., the suppliers of SYSTEM ONE.

The evaluation kit was the dual disc version with a 2MHz MC6800 and 32K of RAM. The terminal used was the SOROC IQ120 which is also available from S.E.E.D.

The Hardware

SYSTEM ONE is a well engineered product in a pleasant, sturdy, non ostentatious box. It is not particularly small, being about 21" x 15½" x 7", but this reflects the fact that the hardware is well laid out on a solid chassis.

The box houses two Shugart SA400 mini-floppy disc drives, an

Stuart Danton is a former consultant with ICL Dataskil.

eight slot motherboard (five are used in the 32K version) and the power supply. The box is well cooled by a rear mounted fan.

The power supply is rated to give +8 Volts at 18 Amps and +15 & —15 Volts each at 3 Amps. These rails are further regulated at each board.

The motherboard uses the SWTP SS-50 bus and each board is held firmly in position by the connectors only.

The CPU board is the MSI CP-1 and is based on a 2MHz MC6800. The 6800 can address up to 64K bytes of memory, but having no input or output ports it uses some memory addresses for I/O leaving 56K bytes as a practical limitation. The board provides for up to 4K bytes of 2708 EPROM and uses an MCM6810A 128 byte static RAM. Clocks are provided for the serial interfaces as well as for the CPU. There are bi-directional bus drivers and address buffers and an MC6875 provides memory timing and DMA control.

On the 32K RAM system, two MSI RAM-16 boards are provided. These use TMS4044 4K x 1 static RAMs. Address selection is by on board DIL switches. On the 32K SYSTEM ONE the first 32K of store is RAM.

The disc drives are controlled by a powerful single board, the SSB BFD-68, which uses the FD1771B-01 single chip disc controller. This leaves enough room on the board for the disc control firmware in on-board ROM (disc bootstrap and all the disc I/O routines). Communication with the CPU is via an MC6820 PIA.

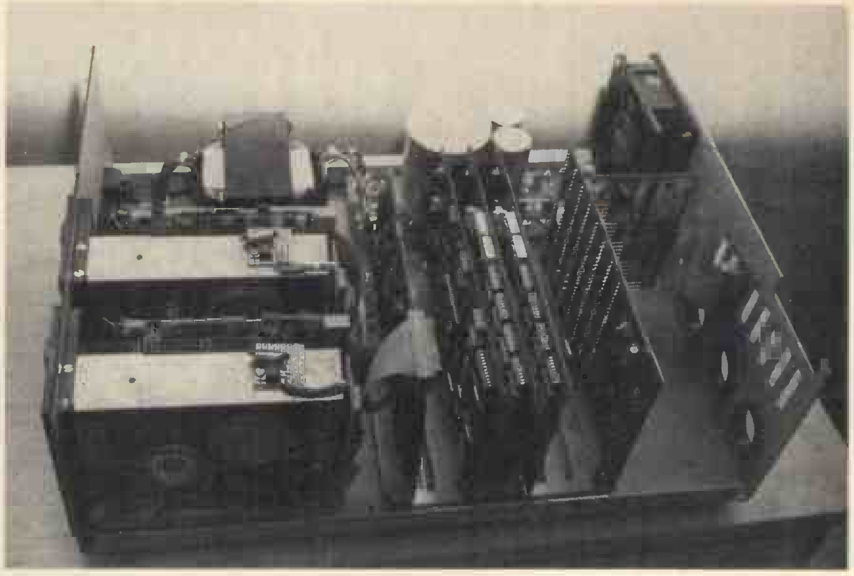
The other board in the SYSTEM ONE is the MSI A-1 interface adaptor which is capable of providing eight serial channels although on the SYSTEM ONE only one is required to connect to the terminal. The terminal governs the data rate and in the case of the SOROC IQ120 used on the evaluation SYSTEM ONE a zippy 19,200 BAUD.

Memory Organisation

The first 32K of memory in the 32K SYSTEM ONE is contiguous RAM. The locations from 0000H to 67FFH are available as user memory, locations 6800H to 7FFFH being used by the disc operating system. Locations 8000H to 83FFH are occupied by the disc firmware.

The SMARTBUG EPROM is on the CPU board and is located from E000H to E3FFH. The MCM6810A RAM is located from A000H to A07FH and is mainly used by SMARTBUG.

There are two other EPROMS on the CPU board. The first, located from E000H, is an extension to SMARTBUG provided by S.E.E.D., giving built in diagnostic aids including confidence tests, memory tests. The second is for interrupt vectors.



MSInside

Controls

One expects a computer to be a mass of flashing lights and a complicated array of switches — that is if you watch T.V.! On the modern computer these functions are performed by monitors and operating systems so on SYSTEM ONE there are just three controls.

The first is the ON/OFF switch. On power up SMARTBUG is automatically started so that SYSTEM ONE is ready for use. The second control is the RESET switch which when used restarts SMARTBUG. The third initiates the disc bootstrap and providing that a DOS68 system disc is in DRIVE 0, the disc operating system is loaded to RAM and becomes operational.

System Software

At the "lowest" level is the elegant SMARTBUG intelligent monitor which provides the SYSTEM ONE programmer with several features including all the "front panel" controls and displays and is a very capable de-bugging aid. Having created a program, the user can use SMARTBUG throughout its development.

The assembly level program may be created using the editor and assembler package (discussed in more detail later), or, for small programs, with the aid of pencil and paper and SMARTBUG itself.

On start up SMARTBUG displays the prompt character (*) on the terminal indicating that it is ready for a command. Commands are recognised on the keying of a single character in response to the prompt. Some commands require qualifying information and this is indicated to the user by a single space displayed on the terminal directly following the command. The following should give you some idea of just how useful SMARTBUG can be, and incidentally just how much software can be packed into 1K bytes.

COMMAND	FUNCTION
A	Display content of A accumulator and allow its alteration to any value specified.
B	As A but for B accumulator.
C	As A but for condition code register.
D	DOS68 warm start (if previously loaded).
G	Go to program starting at location indicated by the contents of locations A-048H & A049H.
I	Insert specified value to all locations between two specified locations.
K	As G but stop program execution when location specified is reached and display current contents of all registers.
M	Display content of specified memory location and if it is RAM allow its alteration to any value specified.
Q	DOS68 cold start i.e. simulate DOS switch by boot-strapping the system disc.
R	Display contents of condition code register, B & A accumulators, index register, program counter and stack pointer.
T	Trace through program one instruction at a time each time space keyed starting from specified location; at each step display as for R command and display instruction code.

X	As A but for index register.
4	Transfer control to location E400H. Gives access to S.E.E.D. diagnostic and confidence testing routines.

The Operating System

The SYSTEM ONE operating system is DOS68. It is a fairly typical microcomputer operating system offering most of the system monitor facilities the user should ever require at console operation and programming levels.

DOS68 is first loaded into SYSTEM ONE by placing a system disc in DRIVE 0 and pressing the DOS switch. After a few moments of disc activity DOS68 "signs on" by displaying on the terminal its name and version number followed by the DOS68 prompt character (&) on the next line. Once loaded to RAM the facilities of DOS68 are available. DOS68 may be warm started using one of the many system functions.

DOS68 is well supported by the BFD-68 System Manual supplied with the BFD-68 disc controller. It provides comprehensive information about the hardware as well as the DOS68 operating system itself. Operators and programmers guides are included in the manual.

To minimise RAM requirements only a few of DOS68 commands are actually resident in RAM all the time, the other commands, known as transient commands, are loaded only when requested. It should be remembered that the disc I/O routines are in ROM. This is a great plus to the system — many a frustrated programmer has ruined a valuable disc due to inadvertant corruption of RAM resident disc I/O routines!

System commands are run by just typing their name (plus any required parameters) in response to the monitor prompt:-

COMMAND	FUNCTION
LIST	Lists all the program and data files on the disc nominated. System files are listed using a switch parameter. Other information about disc status is also listed.
SAVE	Is used to save selected areas of memory content to a nominated (new) disc file.
GET	Loads a binary object file to RAM without executing it.
GETH	As GET for a hexadecimal file.
RUN	Loads a binary object file to RAM and then executes it.
DELETE	Removes a nominated file from the directory.
RENAME	Is used to change the name of a nominated file.
APPEND	Is used to merge nominated files.
PRINT	Displays a nominated file on the terminal.
COPY	Allows a file or a group of files to be copied from one part of a disc to another part of the same disc or to a second disc.
SDC	Is a rare but very useful utility that allows files to be copied one at a time using only DRIVE 0 so removing one of the major objections to single disc systems.
LINK	Is used to tell the bootstrapping routine which program to load and run when the RDM bootstrap is executed.
INSTAL	Allows a nominated file to become a system file and therefore be loadable without the use of RUN.
REMOVE	The opposite to INSTAL.
FIND	Is used to determine where an object file would load to memory and at what location execution would start.
VIEW	Allows source files to be examined without having to use the editor and so create a new file.
FORMAT	Sets up a new disc for use with the file management system so creating an empty directory and also checks each sector on the disc is OK and gives no CRCC error.

After formatting, a disc has available to the user 628 sectors each of 128 bytes i.e. 80,384 bytes in all. Because of this apparent limitation it is good practice to maintain separate discs for data and program storage. SYSTEM ONE is very helpful to the user in keeping to such a practice, providing all the "housekeeping" features necessary.

Support Software

The evaluation SYSTEM ONE was supplied with various support software packages including the SE/A-1 Super Editor/Assembler and two different BASIC packages — one an interpreter — the other a compiler! S.E.E.D. will also have available, in the very near future, a FORTRAN compiler which will further widen the scope of SYSTEM ONE into scientific applications. Also for release in the near future is another compiler known as STRUBAL (an acronym for STRUctured BAsic Language) which is in fact a hybrid language containing elements of PL/M, assembler and BASIC.

The Super Editor is very well conceived, being very simple to use and easy to learn to operate. Noteworthy is the fact that the editing capabilities are not just restricted to the preparation of program source, but are equally useful for simple word processing.

During the edit process each line of text is numbered by the editor, so keeping track of position within the text is easy at all times. Using this editor cuts down the risk of making the type of mistakes that can cost hours to correct.

Facilities include adjustable tab settings, multiple string replacement, block text transfer, insertion and deletion of text, an overlaying facility as well as many other functions; some not to be found on other editors. The documentation provided is simple to understand taking the first time user through the features in a simple and informative manner.

The Assembler too is simple to use, yet it offers a selection of features including multi-level conditional assembly, as well as being fast due to a hashing technique used for symbol table access. Error diagnosis is easy, there is an error summary at the end of the assembly. Used together the

The Soroc terminal used with the MSI 6800.

Super Editor/Assembler package earns its name.

BASIC is still the most popular high level language available for microcomputers. RANDOM BASIC is an interpreter designed in close conformity to the ANSI standard. RANDOM refers to the ability to access records in a disc file by key as well as just sequentially. This makes RANDOM BASIC ideal for applications such as stock control where a sequential search through a disc file could be too slow; and being written to the ANSI standard most BASIC programs can be run using RANDOM BASIC, on the SYSTEM ONE, to great advantage.

RANDOM BASIC has amongst its repertoire the usual mathematical functions, array and string handling, disc control and housekeeping functions, program loop, branch and subroutine handling, terminal accept and display, memory peek and poke and program chaining. From the console, commands provide for the saving and retrieval of BASIC programs and file management. Program debugging is greatly facilitated by a trace feature that displays the line number of each instruction of the BASIC program as it is executed. When a syntax error is encountered the erring instruction line is displayed with the error clearly marked.

The second BASIC package, SDBASIC, being a compiler is very unusual. Programs written with SDBASIC are prepared in a similar way to assembly language programs using the Super Editor. The program is then compiled using system program, SDBAS. As a result an intermediate object file is created. This is translated into an executable file by a further system program SDASM. This file is run by loading it using system program SDRUN each time the program is required.

The big advantage of this is that the object program runs several times faster than it would if it were interpreted. SDBASIC is fairly close to the ANSI standard and has a repertoire similar, but not identical, to RANDOM BASIC.

Application Packages

SYSTEM ONE is not directly associated with specific application packages, but this does not detract from what this microcomputer offers. There is a vast amount of applications written in BASIC that are ideal for SYSTEM ONE.

Prices

The SYSTEM ONE is available from Strumech Engineering Electronics Division (S.E.E.D.). The following prices serve as examples of various builds of SYSTEM ONE but the price of a 16K single disc system complete with ACT-1 terminal and Sanyo 9" monitor is only £1674.

System One Hardware

DISCS	32K RAM
SINGLE	£1510
DUAL	£1790

Pack 10 mini discs £35

Suitable Terminals

ACT-1A Keyboard + Video monitor	£385
SOROC IQ120 Visual Display Unit	£699
ELBIT DS1920 Visual Display Unit	£850


An early start in business, with the MSI 6800, for the author's son.

Software for System One

RBASIC Random Access BASIC Interpreter	£65
6800BC BASIC Compiler (SDBASIC)	£185
SE/A-1 Super Editor/Assembler	£36
STRUBAL Structured BASIC Language	£170
FORTRAN Fortran Compiler	£T.B.A.
TP-1 Text processor	£30
TD-1 Trace Disassembler	£14.25
SG-1 Source Generator	£17.70

All prices are exclusive of delivery and V.A.T.

Maintenance

In keeping with the professional approach they have taken to the production of SYSTEM ONE, S.E.E.D. have organised servicing of this product through one of the national maintenance organisations and the rates for this service, although not known at the time of writing, are likely to be competitive.

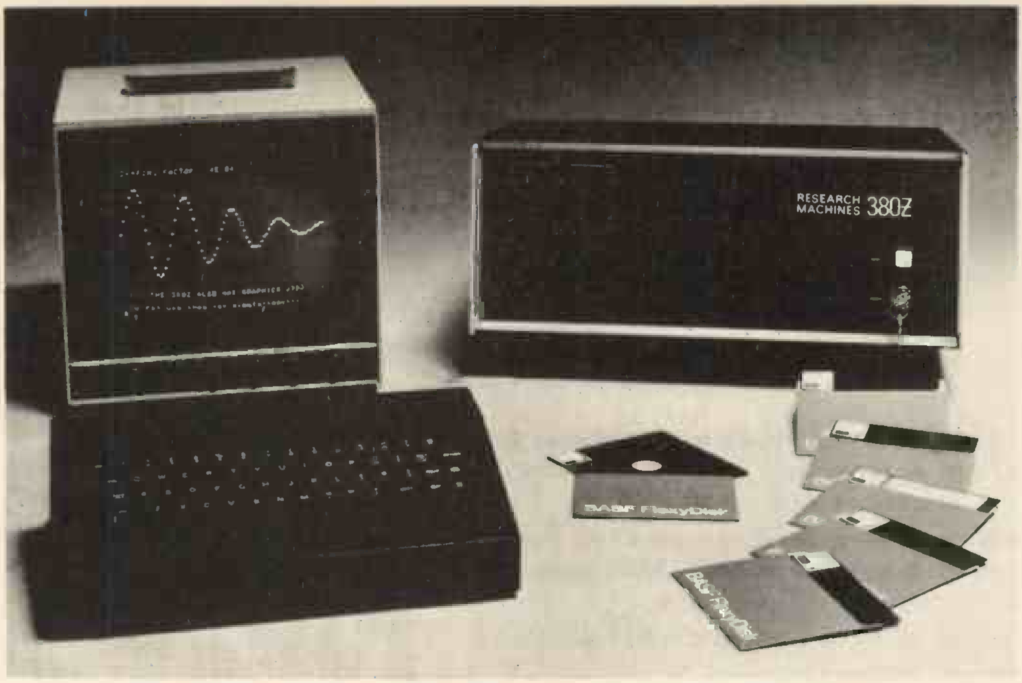
Conclusions

SYSTEM ONE is a well composed piece of hardware. It is well constructed, generously laid out and should give the potential owner reliable service. It is supported by good software that can be used to develop programs for all sorts of applications be they educational, scientific, business or pleasure. Furthermore SYSTEM ONE is an ideal computer to run many of the wealth of BASIC programs that have been developed over the past few years. The BASIC compiler overcomes many of the weaknesses of some BASIC interpreters and should make some BASIC programs very competitive.

SYSTEM ONE was not without some quirks. Hi-Fi lovers who are also home computing enthusiastics may find it disturbing that during disc accesses some breakthrough on VHF radio was detectable on stereo signals. The cooling fan was quite noisy and could be annoying in an office environment; however, as the cooling is more than adequate, a slower speed fan would probably suffice; the boards were cool even after several hours running. The system will lock up when trying to access a disc that is not there — SYSTEM ONE is not alone in this situation.

Nevertheless SYSTEM ONE is a very fine microcomputer which, with prices starting at £1674 for a computer with a mini-floppy disc, keyboard and monitor, must be amongst the most competitive on the British market today.

THE RESEARCH MACHINES 380Z COMPUTER SYSTEM



THE RESEARCH MACHINES 380Z
A UNIQUE TOOL FOR RESEARCH AND EDUCATION

Microcomputers are extremely good value. The outright purchase price of a 380Z installation with dual mini floppy disk drives, digital I/O and a real-time clock, is about the same as the annual maintenance cost of a typical laboratory minicomputer. It is worth thinking about!

The **RESEARCH MACHINES 380Z** is an excellent microcomputer for on-line data logging and control. In university departments in general, it is also a very attractive alternative to a central mainframe. Having your own 380Z means an end to fighting the central operating system, immediate feedback of program bugs, no more queuing and a virtually unlimited computing budget. You can program in interactive BASIC or, using our unique Text Editor, run very large programs with a 380Z FORTRAN Compiler. If you already have a mini-computer, you can use your 380Z with a floppy disk system for data capture.

What about Schools and Colleges? You can purchase a 380Z for your Computer Science or Computer Studies department at about the same cost as a terminal. A 380Z has a performance equal to many minicomputers and is ideal for teaching BASIC and Cesil. For A Level machine language instruction, the 380Z has the best software front panel of any computer. This enables a teacher to single-step through programs and observe the effects on registers and memory, using a single keystroke.

WHAT OTHER FEATURES SET THE 380Z APART?

The 380Z with its professional keyboard is a robust, hardwearing piece of equipment that will endure continual handling for years. It has an integral VDU interface – you only have to plug a black and white television into the system in order to provide a display unit – you do not need to buy a separate terminal. The integral VDU interface gives you upper and *lower* case characters and low resolution graphics. Text and graphics can be mixed *anywhere* on the screen. The 380Z has an integral cassette interface, software and hardware, which uses *named* cassette files for both program and data storage. This means that it is easy to store more than one program per cassette.

Owners of a 380Z microcomputer can upgrade their system to include floppy (standard or mini) disk storage and take full advantage of a unique occurence in the history of computing – the CP/M™* industry standard disk operating system. The 380Z uses an 8080 family microprocessor – the Z80 – and this has enabled us to use CP/M. This means that the 380Z user has access to a growing body of CP/M based software, supplied from many independent sources.

380Z mini floppy disk systems are available with the drives mounted in the computer case itself, presenting a compact and tidy installation. The FDS-2 standard floppy disk system uses double-sided disk drives, providing 1 Megabyte of on-line storage.

**Trademark, Digital Research.*

Versions of BASIC are available with the 380Z which automatically provide controlled cassette data files, allow programs to be loaded from paper tape, mark sense card readers or from a mainframe. A disk BASIC is also available with serial and random access to disk files. Most BASICs are available in erasable ROM which will allow for periodic updating.

If you already have a teletype, the 380Z can use this for hard copy or for paper tape input. Alternatively, you can purchase a low cost 380Z compatible printer for under £300, or choose from a range of higher performance printers.

380Z/32K complete with SINGLE MINI FLOPPY DISK SYSTEM MDS-1

£1787.00

380Z/16K System with Keyboard

£965.00

RESEARCH MACHINES Computer Systems are distributed by RESEARCH MACHINES LTD., P.O. Box 75, Chapel Street, Oxford. Telephone: OXFORD (0865) 49792. Please send for the 380Z Information Leaflet. Prices do not include Carriage or VAT @ 8%.

Jeff Orr and Graham Knott are now trading as:

STACK COMPUTER SERVICES

Dual Drive Floppy Disk

£840 +VAT

* Diskman operating system
* Random & sequential access
* 100k on each drive (200 k on double density)
* ex-stock

Expandamem Memory Expansion

24k – £320 +VAT

16k and 32k also available
All ex-stock

Undirectional interface I-EEE to RS232 £89 + VAT
Bi-directional interfaces, 1 or 2 port, and A/D convertors also available.

We also supply:

Commodore Pet complete range. Peripherals software and manuals. Petsoft, CBM and STACK software.

Apple II 16k, 32k or 48k
mini-disk drives, interface cards and software.

ITT 2020 16k, 32k or 48k
plus all peripherals

Printers Centronics 779, Perkins Elmer Pussycat Teletype 43 etc., including interfaces and stationery.

625 Video Adaptors
for Pet £25 +VAT
Joystick £25 +VAT
Page Printer Interface
(20m A loop) £25 +VAT
ex-stock

PHONE US ON

051-924 1125

for details of our catalogue or write to
Stack Computer Services, 1 Westward View, Waterloo, Liverpool

Stack Computer Services is the trading name of Mardonstar Ltd.
All prices are +VAT at 8% and include carriage. Please make cheques payable to Mardonstar Ltd.

Small Computers for Small Organisations part 2

Mervyn J. Axson

PCW *The principles in this article and the previous one are applicable to other computers, and for this reason alone the articles are well worth reading carefully* PCW.

We now come to the "nitty-gritty". How do we make PET operate our paper system? This is done firstly by entering our membership data onto a cassette tape. A program is entered into PET's internal memory and then 'Run' so writing the data tape. A program consists of a series of numbered Statements, each of which tells the computer to perform a simple operation. It does not matter in what order they are entered for the computer will always go through them in numerical sequence; e.g. if they are entered in the order 3, 2, 1 they will still be executed in sequence 1, 2, 3. If they are labelled 10, 20, 30 the same procedure will result, but with the advantage that if you later want to add something between the first and second statements i.e. 10 and 20, then a statement labelled 15 will do this. This fact is very useful in our system.

Now for the first program, which we will call "Write Membership".

0010	OPEN 1,1,1	Opens a file called 1, on a device 1 (the cassette recorder) to write.	
0021	PRINT #1, MR	Will write MR as the first data item in file 1.	
0022	PRINT #1, D J	2nd data item	Initials
0023	PRINT #1, ALLEN	3rd data item	Forename
0024	PRINT #1, THE NOOK	4th data item	Address
0025	PRINT #1, MAPLE DRIVE	5th data item	"
0026	PRINT #1, ANYTOWN	6th data item	"
0027	PRINT #1, SOME COUNTY	7th data item	"
0028	PRINT #1, ANY 1234	8th data item	'phone number
0029	PRINT #1, 1	9th data item	Category
0030	PRINT #1, 1	10th data item	Relationship

This completes the data for the first member, so we go on to the second member.

```
0101    PRINT #1, MRS
```

and so on, until the data for all the members has been entered.

This may seem to be a laborious task and it is! But it only has to be done *once* and it can be tackled a bit at a time and the program so far saved on a cassette. When convenient this can be loaded into PET and the next batch of data entered. This can be repeated as often as required until "Write membership" is completed. Moreover, a similar process of entering the initial data has to be performed whatever the system, whether it be a large computer or a simple register, so it is not a defect caused by using PET.

The rather strange method of line numbering shown is to help in verifying the data. The first item of data for each member is entered in a line ending in 1, the second in line 2, etc. If you find that you are entering the last data item in line 9 then obviously one data item has been missed out. Leading zeros are given to the line numbers so as to ensure that each type of input appears exactly under the previous one no matter whether the line number is 20 or 9000. This helps to pinpoint missing characters e.g.

```
0021  PRINT #1, MR
0022  PRINT #1, D J
0023  PRNT #1, ALLEN
```

shows up an error in line 0023. The large gaps in the sequence of line numbers between members allows for later insertion of data for new members in correct alphabetical sequence.

"Write membership" can now be run to create the data file on the cassette tape. All fairly simple, but there is one awful snag, which at first sight would appear to invalidate the whole procedure. To test both this and the various operating programs I had set up a dummy membership file of thirty members which was sufficient for this purpose. All went well and odd snags were easily ironed out, but deep down there was a nagging thought that something had been overlooked. But what? Suddenly the thought emerged. Everytime PET is switched on it displays:—

```
***COMMODORE BASIC***
7167 BYTES FREE
READY
```

Simply, a "byte" is a storage space in the computer's internal memory, and the horrible thought was "How many bytes are needed to write the data for each member?" A few rapid tests and calculations showed that it was 160 give or take a few, and that 7167 divided by 160 is approximately 44. This would mean that the longest "write membership" program could only deal with 44 members, and if spaces were to be left to insert future members, perhaps 20 to 30 would be better. This would result in having something like twenty separate data tapes which would have to be put into the cassette player one after the other for the computer to read, so defeating the object of the exercise, automatic operation.

At first this seemed a major stumbling block, but a little thought showed a possible solution. We know that we can produce the program for 25 members, so let us do so for the first 25. Then first, save the program on a cassette tape and then run the program to record file 1 on another cassette. Don't rewind either cassette. Clear PET's internal memory and then repeat the operation to create file 2 which will hold the data for the next 25 members. The only change in the "Write" program will be that it now refers to file 2 e.g.

```
0010  OPEN 2,1,1
0021  PRINT #2, MR
                          and finally
9999  CLOSE 2
```

When the new write program has been run file 2 will be stored on the data tape immediately following file 1 and 2. Write mem will follow 1 Write mem on its own tape. Repeat for file 3 and so on until the data for the 430th member has been dealt with. This is only a form of batch entry, and is no more or less tedious than the original method.

Obviously, we now have files 1 to 18 successively recorded on the data tape. All that we now have to do is to ensure that the operating programs tell PET to process the data for file 1 and then proceed to file 2 and so on until the processing of file 18 has been completed. This is very simple, as you will see.

Before we finish with the creation of the data tape, there is one other operational point to be mentioned. PET needs to know how many members' data is recorded in each file, otherwise it will try to read beyond the end of the file. All this requires is an additional data item after the file is opened and before the data for the first member i.e. in the Write program:—

```
0015  PRINT #1,25
```

Note that if you later add or delete a member or members, this initial data value must be amended. Obviously, since 17 files will cover 425 members, this value will be 5 in file 18.

None of this may be very elegant, but it works, and we now have the membership data stored on a cassette tape, so how do we use it?

We now come to the operating programs, and these are very simple. All that is required is to read the data for a member and then write out those parts that give the required answer. Then go back and read the data for the next member and so on. This is the working part of the program and it must be preceded and followed by some general instructions to PET, such as which file to read. These instructions will be the same for every operating program and we will look at them first.

```
10 LET R = 1         i.e. file no. is 1
20 OPEN R,1,0        Opens file R on device
                     1 to read
30 INPUT R, N        Reads no. of members
                     in file R
40 FOR Z = 1 TO N    Tells PET to go
                     through the working
                     part of the program
                     the same number of
                     times as N
50 INPUT #R,A$,B$,C$,D$,E$,F$,G$,
      H$,I,J         Reads ten data items
                     from file R.
```

That concludes the opening to each program. Note the use of A$, etc. for the first eight data items which are either letters or letters & numbers, whereas just I & J are used for the last two since they are simple numbers. This gives a valuable check as to whether or not a whole line of data including the line number has been missed on the data writing, since PET expects the data types to match. If they don't the program will stop and give an error message. Obviously a test for this should be made at the stage of writing the data tape as each file is completed.

We now leave a number of free lines for the working part of the programs, and then give the closing instructions.

```
500 NEXT Z           goes back to line 50
                     and repeats program
                     for next member
510 CLOSE R          Reached when all
                     members in file have
                     been dealt with
520 LET R = R+ 1     Gives number of next
                     file
530 IF R = 19        All files have been
  GOTO 550           processed
540 GOTO 20          Starts on next file
550 PRINT
  "FINISHED"         End of program.
```

These opening and closing lines can be entered into PET and then recorded on a cassette tape. Then whenever an operating program has to be written, they can be loaded into PET in a few seconds, and lines 51 to 499 are available for the entry of the working part of the program.

Consider the list for the steward. We want to print name (A$,B$,C$,) for each member followed by 'phone number (H$). We must then also print JUNIOR in the appropriate cases.

The program will be:—

```
60 IF I = 13 OR I = 14 GOTO 90 (Sorts
                           out Juniors)
70 PRINT A$; B$; C$; TAB (20); H$
           (Prints data for other categories)
80 GOTO 500          (Transfers to NEXT Z
                      (next member in file))
90 PRINT A$; B$; C$; TAB(20); H$;
    TAB(30); "JUNIOR"   (Prints data for
                 Juniors. Will then goto next
                              line i.e. 500.)
```

Typical output from this program would be:—

```
MR M J AXSON      2916
MR D A J AXSON    2916      JUNIOR
MR J E BROWN      NO PHONE
```

Address labels for playing members would be catered for by:—

```
 60  IF I = 5 OR I = 6 GOTO 500
 70  IF I = 11 OR I = 12 GOTO 500
 80  PRINT A$; B$; C$
 90  PRINT D$
100  PRINT E$
110  PRINT F$
120  PRINT G$
```

Lines 60 and 70 omit social members and lines 80 to 120 print names and addresses of playing members. Even simpler for the list of lady members. Since their categories are 7 to 12, only one line is needed to select them, followed by another line to print their names.

```
60  IF I<7 OR I>12 GOTO 500
70  PRINT A$; B$ ; C$
```

The "horror" task even has a very simple program. It requires more lines but the operations are routine. This task could have been tackled in many ways, but the Hon. Sec. has decided that an individual letter should be produced for each member, which suitably folded may be put in a 'window' envelope, so obviating the need to address envelopes and ensure that each has the correct letter inserted. The program could be:—

```
 60  PRINT A$; B$; C$
 70  PRINT D$
 80  PRINT E$
 90  PRINT F$
100  PRINT G$
101  PRINT
102  PRINT
110  PRINT "DEAR"; A$; C$
111  PRINT
112  PRINT
120  PRINT "YOUR SUBSCRIPTION IS
     NOW DUE FOR NEXT YEAR"
121  PRINT 'PLEASE SEND IT TO THE
     HON. TREASURER AS SOON"
122  PRINT "AS POSSIBLE"
130  IF I = 1 GOTO 150
131  IF I = 2 GOTO 152
132  IF I = 3 GOTO 154
133  IF I = 4 GOTO 156
134  IF I = 5 GOTO 158
..............................
142  IF I = 13 GOTO 174
144  IF I = 14 GOTO 176
150  PRINT "THE SUM DUE IS £51.80"
151  GOTO 180
152  PRINT "THE SUM DUE IS £50.00"
```

```
.................................................
176 PRINT "THE SUM DUE IS £15.00"
180 PRINT
181 PRINT
182 PRINT "YOURS SINCERELY"
183 PRINT "JOE BLOGGS"
184 PRINT "HON. SECRETARY"
```

The Hon. Sec. will of course record the program and the annual chore will be virtually eliminated.

Finally, to show the flexibility of the system let us see how data item J can be used to produce address labels for families.

J has the following values:—

No other family members	1
Wife only	2
Wife & child/children	3
Child/children only	4
Husband is a member	5
Parent is a member	6

The simple address label program is:

```
200 PRINT A$; B$; C$
201 PRINT D$
202 PRINT E$
203 PRINT F$
204 PRINT G$
```

This is what is required if no others in family are members i.e. J = 1 so:

```
60 IF J = 1 GOTO 200
```

Now if J = 2, wife a member, the first line on the label must be altered.

```
70 IF J = 2 GOTO 110
110 PRINT A$; "& MRS"; B$; C$
111 GOTO 201
```

Note that the use of A$; "& MRS" rather than the apparently simpler "MR & MRS" ensures that courtesy titles such as DR and REV are preserved, so frustrating the pedants.

Similarly:

```
80 IF J = 3 GOTO 120
90 IF J = 4 GOTO 130
120 PRINT A$; "& MRS"; B$; "AND
    FAMILY"
121 GOTO 201
130 PRINT A$; B$; C$; "AND
    FAMILY"
131 GOTO 201
```

Now we must omit the family members, those for whom J = 5 or 6.

```
100 IF J = 5 OR J = 6 GOTO 500
```

The objective has been achieved. Many of the routine tasks associated with Membership have been transferred to the small computer. No experts have been required. No great knowledge of programming has been needed. The system is simple and it works. It is flexible and can easily be made to give other outputs if wanted. It may be fairly slow, but that does not matter in this context. Depending on the speed of the printer used, an hour or less will see many of these programs completed.

Let us conclude with an example of the sort of thing that happens in real life. The Hon. Sec. has spent many hours recently carrying out his duties, including preparing the list of lady members. It is a lovely evening and he decides that he will relax and actually play some golf himself. He arrives at the club, only to be greeted by the youngest and most attractive member of the ladies' committee — they always choose her for this type of task — who thanks him for the list and then goes on to say "but you haven't given their 'phone numbers" — they hadn't asked for them — "so could you please add them to the list, and if it is not too much trouble leave it with the steward before you go home to-night?".

In pre-PET days he could have reacted in several ways e.g. resigned on the spot, committed murder, or even, possibly, have given up the idea of playing golf and settled down to some more tedious work. Now, he merely smiles and says "No trouble at all, my dear", goes into the office, loads "PRINT LADIES", changes line 70 to PRINT A$; B$; TAB(20); H$ and sets PET to run. Sometime later, he comes in after an enjoyable game and picks up the revised list from the office; on his way to the mens' bar.

PCW Book Review

Michael James

A FORTRAN COLORING BOOK

By Roger Emanuel Kaufman
The MIT Press
Cambridge, Massachusetts
and
London, England

Here is a book that makes programming sound like fun. It even makes programming in Fortran sound like funl It more than lives up to its motto on the cover: "Learn COMPUTER PROGRAMMING the Painfully Funny way!".

Fortran is not presented in terms of magical formulae; the author puts forward a clear computational model of the Language. Mommy's bureau drawers may seem a frivolous way to describe a computer's memory, but Kaufman's model is consistent and effective.

The topics covered in this book include:
Names and Values (Integers and Reals)
Assignment (Arithmetic Expressions and Built-in Functions)
Branching
Input and Output (Formats)
Loops
Arrays (and implied Do Loops)
Data Types (Explicit, Logical and Complex)
End statement
Subprograms (Functions and Subroutines)
Dummy Arrays
Function Statement
Storage Organisation (Common and Equivalence)
External statement
Data statement

The above list of topics reflects the order of presentation in the book. The text includes as well a complete set of sample problems and exercises. The emphasis is on numerical methods and the reader is encouraged to use these critically.

Kaufman has a very humorous style of presentation. Feminists will be delighted by his offer of an alternative page if they find his outright chauvinism offensive. (See pages 111; there are two!) The handwritten script and illustrations in the text deserve a mention; only a movie of the book could be more entertaining.

On the whole I liked the way Kaufman related the "piddling puny petty paltry particulars" of Fortran. Even his awful puns are endearing. This is an ideal book for people who want to teach themselves computer programming and who have to learn Fortran. It is well worth $ 6.75 (approx. £4) for the paperback edition.

Cornelia Boldyreff
Edinburgh Regional Computing Centre

THE CHEAP VIDEO COOKBOOK

By Don Lancaster,
Howard W. Sams and Co., Inc., $ 5.95

Another masterpiece in clear and interesting electronics from the Lancaster stable, this sequel to the "TV Typewriter Cookbook" should be of interest to any computer hobbyist whether or not they have any intention of building a video display. In contrast to the designs in the previous book, the techniques used are predominantly software orientated, with the resulting reduction in the complexity which gives rise to very low hardware costs. This does of course mean that the software is much more complicated since it carries the burden of much of the timing. For users of the 6502 and the KIM 1 in particular, complete debugged software is provided, and this is said to be easily translatable for the 6800.

As with all books of this type of American origin, it is necessary to remember the differences between the British and American TV standards. However, despite the difference between the 50Hz and 60Hz field rates, the line rates is quite small so only small changes will be needed to the software.

This book sets out by setting a problem: to build a TV display device with minimum cost, and solves it in a way which most hobbyists will be able to understand. It also raises some very interesting possibilities, for instance the possibility of building an intelligent (ie. with onboard dedicated microprocessor) visual display unit at a cost apparently below that of a unit using the so-called super-chip controller.

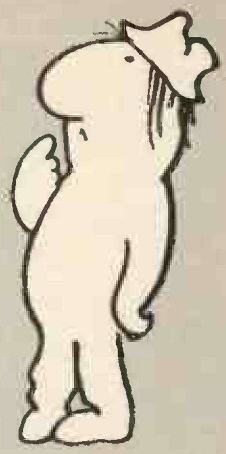
Despite the relatively high cost, which hobbyists must be getting used to, this book is recommended for the libraries of all enthusiasts with interests in hardware.

Paul M. Jessop

Error Message: "Evolutionary Programming" (PCW April 79) In the list of steps on page 46 an extra step should be inserted between steps (iv) and (v). This step reads: "This input will generate a single digit output — for the sake of the example suppose this is 9."

PCW *Next issue: Mike Lord writes about the Z8000* PCW

PCW OPEN PAGE



THE AMATEUR VIEW

Mike Lord

NEWS FROM THE CLUBS

Students at the University College of Swansea are forming a local amateur computer club for the *Swansea/South-West Wales* area; Peter Skan of 6D7 Vivian House, Roman Bridge Close, Blackpill, Swansea, would like to hear from anyone interested in joining.

The recently formed *Grampian Amateur Computer Society* meets on the second Monday of each month; details of their meetings may be obtained from their president Michael Brown at 282 Queens Rd., Aberdeen AB1 8DR. Having recruited several new members as a result of publicity in the local press and radio, they suggest that other groups try the local media; which will usually cost the club nothing.

The *Amateur Computer Club* is now in its seventh year, making it the oldest existing amateur computing organisation in the world and — with around 2000 members — one of the largest. For those who are not aware of it, the ACC is essentially a national club, supported by a bi-monthly newsletter, whose main function is to help enthusiasts get in touch with one another, and to pass ideas, programs and hardware tips between members. Several libraries and user groups exist within the Club, usually based around particular processors. Although the membership fee has risen this year it is still a bargain and can be re-couped many times by taking advantage of discounts available to ACC members. For more details and a (computerised!) membership application form, send a sae to Mr. D.J. Ellis 82 St. Albans Rd., Kingston, Surrey.

A computer Club catering for both amateurs and professionals has been started at *Southgate Technical College*, High St., London N14. Anyone interested in computers (especially microcomputers) is welcome and can ring Paul Woolley at the college (01-886 6521) for further information.

David James of 5 Ox Lane, *Harpenden, Herts* asks if anyone is interested in forming a group in his area. If you are, then don't hesitate to write to him or telephone him at Harpenden 5366.

The *Bristol Computing Club* is now well under way, and holds regular meetings on the third Wednesday of each month. Details of future meetings may be obtained by sending a sae to the Chairman: Mr. L. Wallace at 6 Kilbernie Rd., Bridge Farm Estate, Bristol BS14 0HY.

AND FROM GLOUCESTER

Heathkit's new 'Computer Systems' catalogue makes fascinating reading for DIY enthusiasts, for as well as the H8 and H11 computers, Heath are now producing a useful range of peripherals, including the H10A paper tape reader/punch; which can read 8 hole tape at 50 characters per second and punch at 10 cps; and the H9 ASCII VDU terminal. But perhaps the most interesting item is the H14 line printer. At £386 for the kit, or £554 assembled, this is a dot matrix printer with a maximum instantaneous speed of 135 upper/lower case characters per second. It uses sprocket feed edge punched paper up to 9½" wide and interfaces via a RS232C or 20mA current loop serial interface at 110 to 4800 baud. Given Heathkit's reputation for a quality of design and documentation, this printer must be attractive to any amateur looking for a decent hard-copy peripheral.

POP SHOP

The **PCW Open Page** *Service for amateurs. Buy, sell, exchange —*
One-off advertisements only accepted. Not more than 50 words.

The South Yorkshire Personal Computing Group meets on the second Wednesday of every month, and details can be had from Tony Rycroft, SYPCG Secretary, 88 Spinneyfield, Moorgate, Rotherham, S. Yorks. Tel: Rotherham 74889 (Evenings).

The Merseyside Nascom Users Group meets on the first Wednesday of every month at 7.30 pm at the Manx Suite, The Mona Hotel, James Street, Liverpool 2. Contact: Graham Myers at 051-677 9340 (after 7pm).

FOR SALE
Large SWTPC/MSI installation (24k), twin discs, VDU, IBM printer, word processor and program development software for sale. Many extras **£2000** for lot or will split. Phone or SAE for details.
Rowat, Braeside, Clarach Road, Borth, Dyfed. (097-081-267)

FOR SALE
MK14, Built, Fully Working with: Additional Onboard Ram, All I.C.s socketed, Ram I/O Chip, Replacement Calculator Keyboard, Full Address Decoding, Cassette Interface + 2k 21L02 Ram, Manual, Etc. Worth **£80+, only £65 ono**
Mr. S. Smith, 36 Woodvale Avenue, London, SE25 4AE.

FOR SALE
PET 2001 + some software **£500 ono** Must sell for home purchase. **Phone: 01-341 2385** (after 6.30 pm).

Exidy **SORCERER** (32K). Why pay £850 plus VAT? Mine has had 3 weeks use, is as new and boxed complete with manuals and plug in ROM PAC. Has modulator for use with domestic TV. A powerful machine for **£795.00**
Tel 01-959 6184 Mark Shavick.

Science of Cambridge MK14, sockets throughout, extra RAM10, additional improved keyboard, 2a 5v power supply, cassette interface, "Super Scamps" VDU & AZTEC Modulator, Manual. Wired & working **£60** (will split)
Tel: Chris Swift 0761 34104 (Near Bath)

FOR SALE
KIM I with power supply. Almost new. Connections to power supply and cassette made. Ready to run. Including three manuals, 'First Book of Kim', Microchess and 'Please' games packages on cassettes, with instructions.
£130.00
R. Faulder, 57 Eskdale Avenue, Carlisle, Cumbria, CA2 5QY. Tel. Carlisle 37568.

FOR SALE
T158 Calculator – perfect (bought January); complete with manuals, coding forms, case, charger and other accessories. Genuine reason for sale – cost **£70**: will sell for **£50 (+p/p)**. **Tel 041-778 2419** (after 7pm).

FOR SALE – 8K PET with cover, plus 5 programmed game cassettes (Startrek, Wartrek etc.), 13 blank cassettes, games book, basic programming book + selection of computer mags. Less than 2 months old. **£600 ono** J. Penton, 7 Sutton Road, New Ranges, Shoeburyness, Essex. SS3 9SS.

FOR SALE Two 4K Memory Boards – SWTP S50 bus – assembled and fully tested, **£37 each**. Phone Mr. Johnson-Davies, **0223-63781** (Cambridge).

WANTED Second Hand Mini-Computer and or accessories. Any type or make considered. Will collect if required.
Telephone: Eastbourne (0323) 639043

Become a *Sorcerer's Apprentice*. **Mr. M.S.L. Patrick** casts spells with the aid of an Exidy Sorcerer and is interested in starting an active users' group and a newsletter. Phone him at (01) 642 8362, evenings.

Is the Sorcerer magic? **S. Gimblett,** F & CPO Mess, HMS Neptune, Farlane, Helensburgh, Dunbartonshire, GB4 8HL, thinks so and wants to start a magicians' circle (users' group). Get in touch with him if you're interested.

MK14. Cased. 4A PU. New K/B. Large 7 Seg. Flag Indicators. Audio Interface. Extra RAM. RAM I/O. Spare SC/MP. 4 Manuals. Buyer Collect. **£65.** 22 x 2112 **£25**
Ring Cambridge 67395 after 7pm

Michael Brown, president of *The Grampian Amateur Computer Society* invites canny computer loving Scots to contact him at 282 Queen's Road, Aberdeen, AB1 8DR. Meetings usually on 2nd Monday of each month, locations vary according to event. SAE for details, please.

Ian Preece and friends are endeavouring to begin a Personal Computer Club in the Bournemouth/Poole Area. Interested? Contact him at 246 Stewart Road, Charminster, Bournemouth.

MICROCOMPUTER WANTED
To swap for my 5 octave, touch sensitive electronic piano. Write to exchange spec's:
R.I. GILCHRIST
62 Shelley Road, East Grinstead, West Sussex.

MK14 Professionally built with improved Metal Dome keyboard, perspex case, and power supply IC's socketed throughout **£50.00**
Mr. S. Grosvenor
11 Bernard Road, Oldbury, Warley. W. Mids
021-422 3298

FOR SALE Sinclair Cambridge Programmable Calculator Programme Library and Mains Adapter £15.
Mr. A.D. Cann, 16 Pendeen Road, Threemilestone
Truro Cornwall.

DATA RECORDER MDS1105 For Sale
With Tapes. Must Dispose due to Space Problems, Offers.
Tel: **Wivenhoe (020 622) 4484** After 6pm

HONEYWELL KEYTAPE
7 channel magtape drive. 556 CPI, 24 in/sec complete with manual keyboard, 2K core buffer. Make me an offer – cash or swap
G Coates; Ipswich 59690

FOR SALE
16K LEVEL II TRS-80, and Several Games Programs £640.
Phone – 061 320 8512 Daytime.

FOR SALE
Heath H9 video terminal **£320**
Heath H10 paper tape reader/punch **£225**
Both carefully constructed and maintained, offered at well below the kit prices.
David Fox, 17 Mornington Avenue, Wokingham, Berkshire, RG11 4UE.
Tel: Eversley (0734) 734343

OHIO SUPERBOARD II
Ready-built Microcomputer with power supply **£260 ono.** Tel: Manchester 273 1308 (Ext. 7)

FOR SALE
Motorola D2 Kit (Built) + Manuals **£110**
6800 Assembly Programming **£5**
Scelbi 6800 Cookbook **£6**
Texas T158 Programmable Calculator **£45**
Phone: **Kevin O'Brien,** Birmingham 426 3015 (Evenings/Weekends)

FOR SALE
SINCLAIR PROGRAMMABLE CALCULATOR, PROGRAM LIBRARY AND MAINS ADAPTOR.
A. Cann, 16 Pendeen Road, Threemilestone, Truro, Cornwall

SPECtacular! That's the Sorcerer Program Exchange Club. In business now. Send SAE for bulletin to Messrs. G.F. Counsell & M.P. Hannaby at 65 Trafalgar Road, Birkdale, Southport, Merseyside.

FOR SALE
NASCOM 1 Ready built with power supply and 2K Basic, Games Tapes, Nascom Literature and Programming Manual perfect working order **£190**
Mr. G.R. Lindfield, Telephone: Coventry (0203) 462697

NEW WORKBOOKS FOR SALE
Sorcerer Handbooks (Cost £15) – **£10.00**
Pet TIS Workbooks 1-5) (Cost £15.95) – **£10.00**
Unread. Bought Nascom, so not needed.
Mr. L. Stroud, Hall North, Botton, Danby, Whitby, Yorks
Castleton 281

A computer club, for amateurs and professionals, has started at **Southgate Technical College.** The groups welcome anyone interested in computers, especially micro-computers. Please ring **Paul Woolley on 01-886 6521** for further information.

ASSEMBLY POINT

J. R. Keneally

Many small system users have the problem of balancing hardware and software costs within a fixed budget. Every penny saved on software means better hardware facilities. The D.I.Y. enthusiast is at a big advantage in this situation, since not only are his software costs minimised, but with a little extra effort the software can be designed to use the available hardware to the best advantage. With these thoughts in mind, the author has written a Z80 Editor/Assembler to run on a NASCOM 1 system. The design objectives and general principles should be of interest to anyone developing systems software for a small system. (The Assembler itself should eventually be available through the INMC.

Design Objectives

1. *Target Hardware*

This was the NASCOM 1 with 4K of expansion RAM, and dual cassettes. (A version of the Assembler has in fact run on the basic NASCOM, although this version had numbered instead of named labels). For those unfamiliar with the NASCOM, its basic form has 1K of RAM dedicated to the VDU, plus 1K of user RAM. However, the VDU RAM can be used for programs if desired.

2. *Memory Required*

An important objective was to leave as much RAM as possible for user programs, the aim being 5K out the total of 6K including VDU RAM. This was achieved by using separate Editor, Assembler, and Linker programs.

Development of a user program proceeds as follows. The source code is first saved on cassette using the Editor. This cassette is then input to the Assembler, which outputs an object code cassette containing unassembled labels, but having all code translations completed. Finally, the executable program is linked direct into RAM by feeding the object code into the Linker program. Some other advantages of this arrangement are the potential for creating libraries of object code routines which can be linked with other programs, and the ability to relocate programs easily.

3. *Flexibility*

The Assembler was written with later expansion in mind, in particular the inclusion of instructions which decode to more than one machine code instruction, and of improved assembler directives. These objectives were met by using a single data table to define both instruction syntax and the actions required to generate the machine code or handle directives. This approach is efficient in memory requirements, easily debugged, and recommended to anyone thinking of writing an Assembler. Further details are discussed below.

Syntax Decoding

All syntax decoding problems are similar in the sense that they specify legal combinations of some basic alphabet or character set, and the context within which these combinations are valid. The legal combinations can themselves be combined into sequences, thus making 'sentences' from the 'words'. Repeating the process generates 'paragraphs' from the 'sentences' etc.

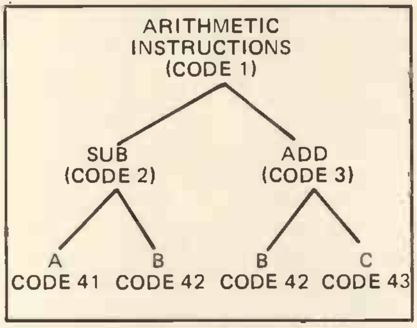


Figure 1. Syntax tree

As a simple example, consider the problem of specifying the syntax for an assembler having only four possible instructions, ADD B, ADD C; SUB A, SUB B. (See Fig. 1). At the highest level of combination, all these instructions are arithmetic types. At the next level down, level 2, the SUB instruction can be followed by A or B, while the ADD instruction can be followed by B or C. The tree diagram of Fig. 1 therefore defines the complete syntax for this assembler.

In a practical assembler, the full tree diagram would be subdivided into a number of sub-trees, each containing only a few levels.

All that remains is to *translate tree diagrams into software*. Let us attach a unique number, the syntax code, to each syntax *element* in the tree, where a syntax element can be any single character, group of characters or word, or any higher level of combination.

In our example, the highest level is all arithmetic instructions, and this is allocated code 1 in Fig. 1.

The problem is now solved, bar the hard work!

Characters from the actual instruction are first converted into the equivalent syntax codes.

A *preset* table (SYNTAB) is created containing the syntax codes defined in the tree diagram, and an indication of the sequence in which these must occur.

The instruction codes are then checked against those defined in SYNTAB.

If we now take a stroll through the process of verifying the syntax of an instruction, the structure required in the SYNTAB table will become apparent. We enter the table at the highest level, which contains code 1. On its own, this number is not very informative, since we do not know whether it represents a single character, or higher grouping. Consequently, a control bit is needed in each entry to indicate this. In addition, an address word, or means of indicating the start of the next level, is required. Armed with this data, we can proceed to the next level down.

At this point, it is instructive to view the above process in a different way. Let us think of the code (1) at the first level, and its associated control bit, as a 'call' to a 'subchain' defined elsewhere in the table, i.e. at the address defined by the address word mentioned above. Instead of level 2, therefore, we shall change the nomenclature to sub-chain 1. The parallel between subchains and subroutines in a program is obvious, and for most people it is easier to think in terms of subchains rather than syntax levels.

Having reached subchain 1, we see that this can be either of two elements, code 2 or code 3, and that both of these are themselves subchains. (See Fig. 1). Subchain 2 is accessed first, and this is seen to have only one possible sequence of letters (SUB) and no subchain calls, and can therefore be checked against the syntax codes from the actual instruction. If the codes match, then we may return to the call to subchain 2 with success.

However, this creates a problem, since where do we go next? If the match had failed, we would have checked subchain 3, and if that had failed to match also, we would exit with a syntax error. When a match is obtained, however, another address word is required to indicate where to find the next SYNTAB entry, assuming that the syntax is not complete. Every SYNTAB entry is therefore like a conditional jump, since a failed match always goes to the adjacent table entry, while a successful match branches to a non-consecutive entry.

Most of the SYNTAB entry structure is now apparent, and may be summarised as follows:—

1. Define syntax code values for all syntax elements.
2. Equate all code values which are not single characters with the corresponding sub-chain.
3. Use a syntax table with the following minimum format for entries:
   a) Control word bits defining subchain calls, end of syntax, and end of a set of consecutive alternative codes.
   b) Syntax code value, and address of subchain if needed.
   c) Address of next entry if this code was found successfully.

Figure 2 shows a syntax table for the above example, constructed along the lines indicated. The first word of each entry is a control word (CW) with the following bit allocations:

a) Bit 0. End of syntax, or of subchain.
b) Bit 1. End of set of alternative codes.
c) Bit 2. Sub-chain call indicator.
d) Bit 3. Save sub-chain code if set.

Two more control bits have been used in addition to those mentioned earlier. These control the storage of syntax codes in an output buffer.

LABEL	CW	SYN CODE	CHAIN ADDR	NEXT ADDR
L0:	06	01	L1	X
L1:	04	02	L2	N1
	06	03	L3	N2
N1:	11	41	X	X
	13	42	X	X
N2:	11	42	X	X
	13	43	X	X
L2:	02	53	X	N3
N3:	02	55	X	N4
N4:	08	42	X	X
L3:	02	41	X	N5
N5:	02	4E	X	N6
N6:	0D	44	X	X

Figure 2. The SYNTAB Table
All numbers are in HEX.
X means unused address

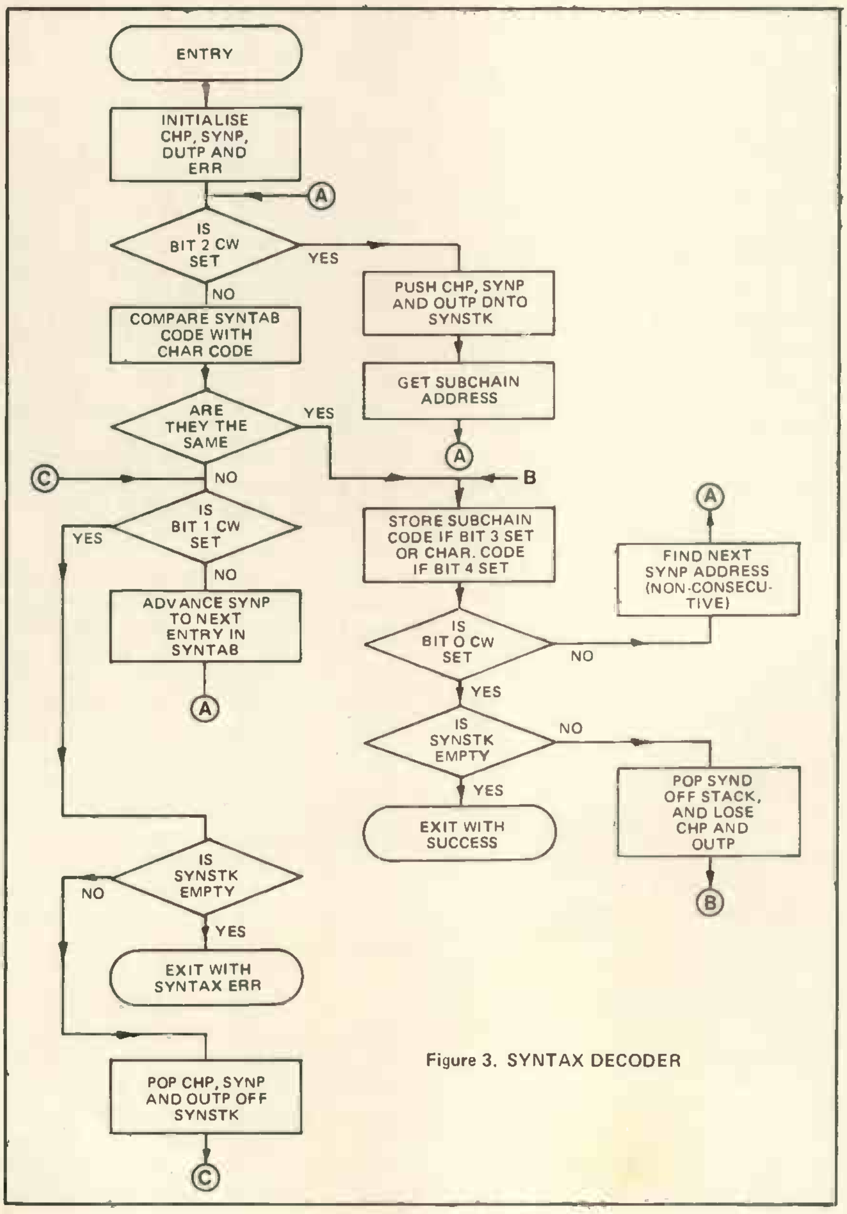


Figure 3. SYNTAX DECODER

Also note that the syntax codes for single characters in Figure 2 are the same as the ASCII code without parity.

Figure 3 is a flow diagram for a routine which can be used as a syntax decoder in conjunction with the table. The input to this routine is the current instruction, and the output is a list of checked syntax codes plus an indicator for legal syntax. Note the use of a special software controlled stack (SYNSTK) which is used in calling subchains. The following is a brief description of the main variables used:

1. CHP. This is used to point to the current position in the input list.
2. OUTP. This points to the next free entry in output list of syntax codes.
3. SYNP. Points to the current SYNTAB entry.
4. ERR. This is cleared to indicate a syntax error.
5. SYNSTK. This is a FIFO stack, used to save and restore the first three variables above on entry to or exit from a subchain.

So far, no mention has been made of how the actual machine instructions are generated. This is done simply by defining a set of routines which, in conjunction with code definition tables, are capable of generating the required machine code. These routines constitute the code generator, and may be identified by numbers which can be inserted at appropriate points in the syntax table, together with any required parameters. The appropriate routine can then be executed when the associated syntax code has been successfully matched in SYNTAB. A simple modification of the flow diagram of Figure 3 will allow this to be done.

Typical routines for a Z80 code generator might be:

1. Get instruction from code defining list.
2. Modify instruction (e.g. to insert register reference).
3. Insert byte to instruction (e.g. for IX and IY references).
4. Write instruction to output device.

This is not intended to be an exhaustive list, but gives an idea of the routines needed. Assembler directives can be handled by a second set of routines.

In conclusion, Figure 4 illustrates all of the main elements of this type of Assembler. Once an Assembler has been written within this general framework, it is relatively easy to generate assemblers with different syntax, or cross assemblers, or even simple high level languages such as BASIC. It is hoped that this article will encourage budding Assembler writers to take the Bit firmly between the teeth!

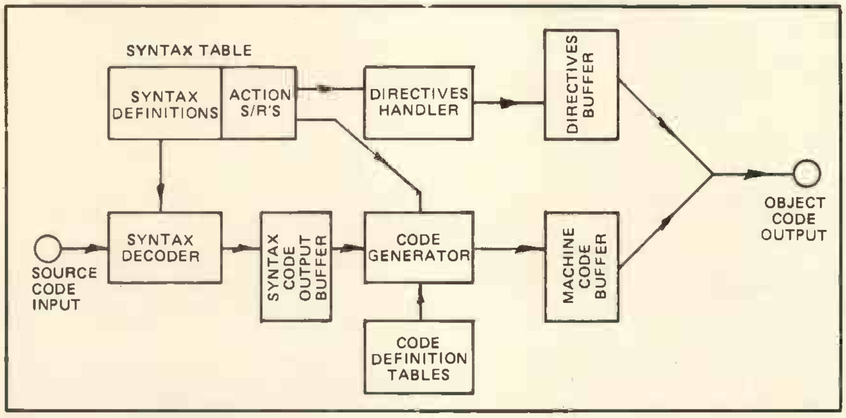


Figure 4. Assembler Framework

PARKINSON'S REVAS

Continued from Vol. 2 No. 1

Note *The author has forwarded some corrections which will be published in Vol. 2 No. 3.*

D. W. Parkinson

```
0000      ; A REVERSE ASSEMBLER FOR 8080/Z80 CODES
0000      ;
0000      ;  WRITTEN BY DAVID PARKINSON
0000      ;        12-08-78
0000      ;        VERSION 2.1
0000      ;
0000      ; THE OUTPUT IS IN STANDARD Z80 MNEMONICS (ALMOST)
0000      ;
0000      ; THE PROGRAM IS IN THE FORM OF A SUBROUTINE
0000      ; IT IS CALLED BY A CONTROL PROGRAM WHICH
0000      ; COULD HAVE READ OBJECT CODE FROM CASSETTE
0000      ; TAPE,DISC ETC, OR BE JUST REQUESTING
0000      ; THE TRANSLATION OF AN EXISTING PROGRAM IN MEMORY
0000      ; (IN WHICH CASE IT MUST NOT OVERLAY
0000      ;  THE REVERSE ASSEMBLER!).
0000      ;
0000      ; THE CONTROL PROGRAM MUST HAVE PLACED THE OBJECT
0000      ; CODE SOMEWHERE IN MEMORY. REVAS IS THEN ENTERED
0000      ;  WITH THE FOLLOWING SEQUENCE....
0000      ;
0000      ; LD BC WITH THE "PROGRAM" ADDRESS
0000      ;     (THE STARTING ADDRESS USED IN THE LISTING)
0000      ; LD DE WITH THE START MEMORY ADDRESS
0000      ;     (ACTUAL LOCATION OF CODE TO BE REVERSE ASSEMBLED)
0000      ; LD HL WITH THE END ADDRESS
0000      ;     (REVAS RETURNS WHEN THIS ADDRESS IS PASSED)
0000      ; CALL REVAS
0000      ;
0000      ;   VARIABLE STORAGE
0000      ;
0000              ORG    $1000       ;WORKSPACE
1000 0003         OUTPUT: DS   3           ;RAM PATCH TO ROUTINE
1003 0002         ADDR:   DS   2           ;PROGRAM COUNTER
1005 0002         MADDR:  DS   2           ;CURRENT MEMORY ADDRESS
1007 0002         EADDR:  DS   2           ;WHERE TO STOP
1009 0002         OPCADR: DS   2           ;POINTER IN BUFFR
100B 0001         HXYFLG: DS   1           ;HL,IX,IY FLAG
100C 0002         LINEA:  DS   2           ;ADDR. LINE REF. LABEL
100E 0002         LABELA: DS   2           ;LABEL VALUE
1010 0002         LABELP: DS   2           ;POINTER TO POS. IN BUFFER
1012 0034         BUFFER: DS   52          ;OUTPUT BUFFER
1046              ;
1046              ; REVERSE ASSEMBLER ENTRY POINT
1046              ;
1046                      ORG  $F800
F800 ED 43 03 10  REVAS:  LD   (ADDR),BC ;SAVE DATA
F804 ED 53 05 10          LD   (MADDR),DE
F808 22 07 10             LD   (EADDR),HL
F80B              ; NEXT LINE OF OUTPUT
F80B 2A 07 10     NEXTL:  LD   HL,(EADDR);CHECK FOR END
F80E ED 5B 05 10          LD   DE,(MADDR)
F812 AF                   XOR  A           ;CLEAR CARRY
F813 ED 52                SBC  HL,DE
F815 D8                   RET  C           ;RETURN IF PAST
F816 CD 27 F8             CALL INITB       ;INITIALISE OUTPUT BUFFER
F819              ; LOOP POINT AFTER IX/IY (DD/FD)
F819 CD DB F8     MORE:   CALL BYTE        ;GET NEXT BYTE
F81C CD 5A F8             CALL DECODE      ;DECODE IT
F81F 21 12 10             LD   HL,BUFFER ;SET ADDRESS
F822 CD 00 10             CALL OUTPUT      ;PRINT OR WHATEVER
F825 18 E4                JR   NEXTL       ;LOOP
F827              ;
F827              ; INITIALISE THE OUTPUT BUFFER (AND CERTAIN VARIABLES)
F827              ;
F827 11 12 10     INITB:  LD   DE,BUFFER ;GET START ADDRESS
F82A D5                   PUSH DE          ;GET TO IX
F82B DD E1                POP  IX
F82D 21 00 00             LD   HL,0        ;CLEAR LABEL POINTER
F830 22 10 10             LD   (LABELP),HL
F833 32 0B 10             LD   (HXYFLG),A;CLEAR FLAG
F836 2A 03 10             LD   HL,(ADDR) ;WRITE ADDRESS
F839 22 0C 10             LD   (LINEA),HL;KEEP ADDRESS
F83C CD D0 F8             CALL HEX4
F83F 62                   LD   H,D         ;NOW CLEAR REST OF BUFFER
F840 6B                   LD   L,E
F841 13                   INC  DE
F842 ED 53 09 10          LD   (OPCADR),DE;SET POINTER FOR OPCODE
F846 36 20                LD   (HL),$20  ;SET FIRST SPACE
F848 01 30 00             LD   BC,48       ;ONE MORE THAN NECESSARY
F84B ED B0                LDIR             ;CLEAR
F84D DD 36 2F 3B          LD   (IX+47),' ;';COMMENT DELIM.
F851 36 0D                LD   (HL),$D   ;SET CR AT END
F853 23                   INC  HL          ;FOLLOWED BY...
F854 36 00                LD   (HL),0    ;... A NULL
F856 11 2D 10             LD   DE,BUFFER+27;SET ADDRESS FOR MNEMONIC
F859 C9                   RET              ;DONE
F85A              ;
F85A              ;DECODE BYTE AND EXECUTE
F85A              ;
F85A F5           DECODE: PUSH AF          ;SAVE
F85B E6 C0                AND  $C0         ;ISOLATE MSBS
F85D FE 40                CP   $40         ;01XXXXXX?
F85F CA AD F9             JP   Z,LOAD8     ;YES,8-BIT LOAD
F862 FE 80                CP   $80         ;10XXXXXX?
```

```
F864  CA 1C FA     0096         JP    Z,ARITH8    ;YES,8-BIT ARITHMETIC
F867  F1           0097         POP   AF          ;GET BYTE BACK
F868  F5           0098         PUSH  AF          ;PUT ON STACK FOR LATER
F869  E6 8F        0099         AND   $8F         ;ISOLATE 7XXX3210
F86B  07           0100         RLCA              ;FORM AS  ..32107.
F86C  07           0101         RLCA
F86D  21 78 F8     0102         LD    HL,TABLE    ;LOAD BASE ADDRESS
F870  CD 2A F9     0103         CALL  FTADR       ;FORM ADDRESS
F873  4E           0104         LD    C,(HL)      ;GET ADDRESS AT THAT ADDRESS
F874  23           0105         INC   HL
F875  46           0106         LD    B,(HL)
F876  C5           0107         PUSH  BC          ;GO TO IT VIA STACK
F877  C9           0108         RET
F878               0109 ;
F878               0110 ;JUMP TABLE FOR DECODE
F878               0111 ;32 ENTRIES
F878               0112 ;
F878  1F FB        0113 TABLE:  DW    NOPETC
F87A  7A FA        0114         DW    CJR
F87C  4D F9        0115         DW    LD16
F87E  4C FA        0116         DW    POP
F880  06 FC        0117         DW    STIND
F882  7A FA        0118         DW    CJR
F884  2F F9        0119         DW    INC
F886  67 FB        0120         DW    JPETC
F888  2F F9        0121         DW    INC
F88A  7A FA        0122         DW    CJR
F88C  34 F9        0123         DW    DEC
F88E  51 FA        0124         DW    PUSH
F890  AD F9        0125         DW    LOAD8
F892  1C FA        0126         DW    ARITH8
F894  C8 FA        0127         DW    ROTMIS
F896  F5 FA        0128         DW    RST
F898  49 FB        0129         DW    EXAETC
F89A  7A FA        0130         DW    CJR
F89C  6D F9        0131         DW    ADDHL
F89E  D6 FB        0132         DW    RETETC
F8A0  1B FC        0133         DW    LDIND
F8A2  7A FA        0134         DW    CJR
F8A4  34 F9        0135         DW    DEC
F8A6  BB FB        0136         DW    CBETC
F8A8  2F F9        0137         DW    INC
F8AA  7A FA        0138         DW    CJR
F8AC  34 F9        0139         DW    DEC
F8AE  63 FA        0140         DW    CALETC
F8B0  AD F9        0141         DW    LOAD8
F8B2  1C FA        0142         DW    ARITH8
F8B4  C8 FA        0143         DW    ROTMIS
F8B6  F5 FA        0144         DW    RST
F8B8               0145 ;
F8B8               0146 ;UTILITY SUBROUTINES
F8B8               0147 ;
F8B8               0148 ;TRANSFER A AS TWO HEX CHARACTERS TO
F8B8               0149 ;OUTPUT BUFFER (DE POINTS TO CURRENT POSITION
F8B8               0150 ;IN THE BUFFER).
F8B8               0151 ;
F8B8  F5           0152 HEX2:   PUSH  AF          ;SAVE A
F8B9  0F           0153         RRCA              ;SHIFT HI' NIBBLE DOWN.
F8BA  0F           0154         RRCA
F8BB  0F           0155         RRCA
F8BC  0F           0156         RRCA
F8BD  CD C1 F8     0157         CALL  HEX1        ;PRINT IT
F8C0  F1           0158         POP   AF          ;RECOVER A
F8C1  F5           0159 HEX1:   PUSH  AF          ;SAVE A
F8C2  E6 0F        0160         AND   $0F         ;ISOLATE LO' NIBBLE
F8C4  F6 30        0161         OR    $30         ;MAKE ASCII
F8C6  FE 3A        0162         CP    $3A         ;NUMERIC?
F8C8  38 02        0163         JR    C,NUMB      ;YES,SKIP
F8CA  C6 07        0164         ADD   7           ;NO,MAKE ALPHA
F8CC  12           0165 NUMB:   LD    (DE),A      ;PUT IN BUFFER
F8CD  13           0166         INC   DE          ;BUMP ADDRESS
F8CE  F1           0167         POP   AF          ;RECOVER A
F8CF  C9           0168         RET
F8D0               0169 ;
F8D0               0170 ;HL TO BUFFER AS FOUR HEX CHARACTERS
F8D0               0171 ;
F8D0  F5           0172 HEX4:   PUSH  AF          ;SAVE A
F8D1  7C           0173         LD    A,H         ;GET HI BYTE
F8D2  CD B8 F8     0174         CALL  HEX2        ;PRINT IT
F8D5  7D           0175         LD    A,L         ;GET LO BYTE
F8D6  CD B8 F8     0176         CALL  HEX2        ;PRINT IT
F8D9  F1           0177         POP   AF          ;RECOVER A
F8DA  C9           0178         RET
F8DB               0179 ;
F8DB               0180 ;GET NEXT BYTE OF DATA. PUT IN OUTPUT
F8DB               0181 ; STRING ALONG WITH ANY ASCII
F8DB               0182 ; EQUIVALENT
F8DB               0183 ;
F8DB  DD 23        0184 BYTE:   INC   IX          ;BUMP POINTER
F8DD  2A 03 10     0185         LD    HL,(ADDR)   ;UPDATE PC
F8E0  23           0186         INC   HL
F8E1  22 03 10     0187         LD    (ADDR),HL
F8E4  2A 05 10     0188         LD    HL,(MADDR);LOAD BYTE ADDRESS
F8E7  7E           0189         LD    A,(HL)      ;LOAD BYTE
F8E8  23           0190         INC   HL          ;UPDATE ADDRESS
F8E9  22 05 10     0191         LD    (MADDR),HL
F8EC  2A 09 10     0192         LD    HL,(OPCADR);NOW PUT IN OUTPUT...
F8EF  23           0193         INC   HL          ;SPACE BETWEEN OPCODES
F8F0  EB           0194         EX    DE,HL       ;..STRING IN CORRECT..
F8F1  CD B8 F8     0195         CALL  HEX2        ;..PLACE.
F8F4  EB           0196         EX    DE,HL       ;RESTORE POINTER
F8F5  22 09 10     0197         LD    (OPCADR),HL;SAVE IN CASE NEEDED AGAIN
F8F8  F5           0198         PUSH  AF          ;SAVE
F8F9  3C           0199         INC   A           ;ROLL "RUB OUT" ROUND
F8FA  E6 7F        0200         AND   $7F         ;STRIP OFF PARITY
F8FC  FE 22        0201         CP    $22         ;PRINTABLE?
F8FE  38 04        0202         JR    C,BYOUT     ;NO,EXIT
F900  3D           0203         DEC   A           ;CORRECT A
F901  DD 77 2F     0204         LD    (IX+47),A ;APPEND TO BUFFER
F904  F1           0205 BYOUT:  POP   AF          ;RECOVER A
F905  C9           0206         RET
F906               0207 ;
F906               0208 ;WRITE "LD"
F906               0209 ;
F906  EB           0210 WRLD:   EX    DE,HL
F907  36 4C        0211         LD    (HL),'L'
F909  23           0212         INC   HL
F90A  36 44        0213         LD    (HL),'D'
F90C  21 32 10     0214         LD    HL,BUFFER+32;SET OPERAND ADDRESS
F90F  EB           0215         EX    DE,HL
F910  C9           0216         RET
F911               0217 ;
F911               0218 ; WRITE ','
F911               0219 ;
F911  EB           0220 COMMA:  EX    DE,HL
F912  36 2C        0221         LD    (HL),','
F914  23           0222         INC   HL
F915  EB           0223         EX    DE,HL
F916  C9           0224         RET
F917               0225 ;
F917               0226 ;WRITE "$"
F917               0227 ;
F917  EB           0228 DOLLAR: EX    DE,HL
F918  36 24        0229         LD    (HL),'$'
F91A  23           0230         INC   HL
F91B  EB           0231         EX    DE,HL
F91C  C9           0232         RET
F91D               0233 ;
F91D               0234 ;COPY OVER CHARACTERS
F91D               0235 ;
F91D  ED A0        0236 COPY6:  LDI
```

To be continued

OHIO SCIENTIFIC

Microcomputers from the world's largest full-line manufacturer

The C2-4P Mini Floppy

20K RAM
Basic + Assembler
Personal, Games, Small Business & Educational Disks
90K Mini Floppy Storage
Printer Interface
OS 65D V·30 Operating System
Only £1595.00 Complete + VAT.

Economic expandable systems with good disk based software, available now.
See your nearest dealer for full price list and catalogue.

Other systems available include the C3 OEM with 32K RAM, 512K of disk storage and BASIC as standard, £2950.00 + VAT. (FORTRAN and COBOL available as extras.) All dealer enquiries direct to Abacus Computers Limited.

Abacus Computers Limited
62 New Cavendish Street
London W1 Tel: 01-580 8841

Mutek
Quarry Hill, Box Corsham
Wiltshire SN14 9HT
Tel: 0225-743289

Thames Personal Computers
13 Wilmot Way Camberley
Surrey Tel: 0276-27860

Linn Products
235 Drakemire Drive
Castlemilk Glasgow
G45 9SZ Scotland
Tel: 041-634 3860

U Microcomputers
PO Box 24 Northwich
Cheshire CW8 1RS
Tel: 0606-75627

SORCERER™

Exidy inc.

Just look at these features..

For the technical. . . . . .

The Sorcerer Computer is a completely assembled and tested computer system ready to plug in and use. The standard configuration includes 63 key typewriter-style keyboard and 16 key numeric pad, dual cassette I/O, with remote computer control at 300 and 1200 baud data rates, RS232 serial I/O for communication, parallel port for direct Centronics printer attachment, Z80 processor, 4K ROM operating system, 8K Microsoft BASIC in separate plug-in Rom Pac™ cartridge, composite video of 64 chars 30 lines, 128 upper/lower case ASCII character set and a 128 user-defined graphic symbols, up to 32K on-board RAM memory, operators manual, BASIC programming manual and cassette/video cables, connection for S100 bus expansion unit giving access to the spectrum of exciting and useful peripheral devices, such as Floppy disk drives, voice recognition/synthesis battery back-up board in case of power failure, additional memory boards, E-PROM cards give you the facility to program and re-program your own ROM memories etc. etc. This is the most useable and flexible system that's now available to the home and business user at such a low price.

* WORD PROCESSING
* PLUG IN PROGRAMME CARTRIDGES
* WORKS WITH NORMAL TV
* EXPANDS TO 48 K
* 1,920 CHARACTER DISPLAY
* OUTPUTS TO PRINTER
* MANY EXPANSION FACILITIES

PRICES		£		£
8K	Sorcerer	650	Line Printer	850.00
16K	Sorcerer	760	Video Display	240.00
32K	Sorcerer	859	Development Pack	70.00
630K	Dual Disc Drive	1,200	Word Processing Pack	70.00
143K	Single Disc Drive	500	Technical Manual	8.95
S100	Expansion Unit	210	Memory Expansion 16K	110.00

NOW CONTACT YOUR LOCAL DEALER . . .

LANCASHIRE
051-236 0707 MICRODIGITAL 25 Brunswick St., Liverpool L2 BJ

WEST YORKSHIRE
053-65094 BASK COMPUTING Oakville, Oakworth Rd., Keighly

SHEFFIELD
0742-668767 E.S. MICROCOMPUTERS 7 Berkley Precint, Ecclesall Rd., S11 8PN

NORTH WALES
0248-52042 TRYFAN A/V SERVICES 3 Swifts Bldgs., High St., Bangor, Gwynedd

AVON
0272-292375 ELECTROPRINT 5 Kingsdown Parade, Bristol BS6 5UD

NORTH HANTS
0536-83922 H.B. COMPUTERS LTD. 22 Newland St., Kettering

LONDON & Counties
BERKSHIRE
0635-30505 NEWBEAR COMPUTING STORE 40 Bartholomew St., Newbury RG14 5LL

KENT
01-300 0380 INFORMEX 61 Harland Avenue, Sidcup, DA15 7NY

SURREY
0276-34044 MICROBITS 34b London Rd., Blackwater, Camberley

0276-62506 T. & V.J. MICROCOMPUTERS 165 London Rd., Camberley

. . OR SEND COUPON FOR FURTHER INFORMATION

To: GEOFF WILKINSON
FACTOR ONE *Computers Ltd.,*
11-17 Market Place, Penzance, Cornwall.
Tel: 0736-66336/66565

Please send me details of the SORCERER COMPUTER

Name ______________________

Address ______________________

Tel. No. ______________________

All Systems Are Not Created Equal

Your computer application is unique. It differs from all others. It is because not all applications are equal that MSI has developed a variety of computer systems.

At the heart of every MSI System is the powerful MSI 6800 Computer, one of the fastest and most versatile available. Depending on the System you select, the MSI 6800 has from 16K to 56K of RAM. Mass memory storage in MSI Systems range from 315K bytes in the System 1 to over 10 megabytes in our most powerful System 12.

In addition to the computer and memory subsystem, MSI Systems include a CRT terminal and high speed character printer. The System 12 is housed in a compact desk unit.

As with hardware, computer software is not always created equal. Since there are a myriad of programs available, MSI offers a choice of Operating Systems for use with your MSI Computer System. Of course, our favourite is MSIDOS, but we offer the powerful SDOS operating system as well. All MSI Systems will support the other software products associated with each operating system.

MSI also has a variety of software programs including a Multi-User Basic program capable of supporting up to four users.

MSI Systems are currently being used in a broad spectrum of personal, scientific, educational, professional, and business situations. In addition to our Systems, we can supply you with individual components for personal and OEM use. All MSI System components are available, some in kit form.

Write or call us for more information about MSI Systems and products and the name of your nearest MSI dealer.

seed

Strumech Engineering
Electronic Developments
Portland House,
Coppice Side, Brownhills,
BRO 4321 Telex: 335243

PUZZLE DAZZLE No.1 – THE UNDAZZLES

Neil Harrison

Some months ago now, in the November 78 issue of PCW, there was a letter from **David Broughton** posing an interesting little programming problem for the 8080 which went something like this:

"Imagine you possess an 8080 with 64K of read/write memory which you want to clear. Write a program that sets all 65536 bytes to zero."

At David's suggestion this turned into a competition the object of which was to write the above program in either the fewest number of bytes or the shortest execution time. Spurred on by a two pound (increased to five pound) prize solutions soon started pouring into the PCW offices. Here at last are the results of the first Puzzle Dazzle competition.

Many people assumed that the program had to start at location 0000, presumably since that is where the 8080 starts execution after a reset. Not so! Since the original problem did not specify whether or not the system in question has a monitor program or not I am prepared to assume it has. After all, how would you set the program in there in the first place? Given a monitor program, execution can usually start at any location. However, just for the purists who think that we **should** start at 0000, here is the best solution. It was sent in by Martin **Sach** and is only 8 bytes long.

Address	Code	Mnemonic	number of clock cycles
0000	21 06 00	LXI H,0006	10
0003	54	MOV D,H	5
0004	5C	MOV E,H	5
0005	F9	SPHL	5
0006	D5	PUSH D	11
0007	E9	PCHL	5

The first instruction sets the HL registers to 0006 (00 in H, 06 in L) and the next two set the D and E registers to zero using the data (00) in H. The stack pointer SP is then loaded with 0006 from HL. 'Pushing' the DE registers in the next instruction writes zeros into locations 0004 and 0005 and the stack pointer is reduced by two to 0004. The next instruction, PCHL, means 'jump to the address stored in HL' which, since HL contains 0006 means that execution continues with the PUSH D instruction. Successive Pushes write pairs of zero bytes *down* through memory, past location 0000 to FFFF and on;until,after 32767 pushes,the only two bytes in memory which are not zero are the PUSH D and PCHL instructions themselves. The 32768th PUSH overwrites these last two and the whole of memory is set to zero. The whole process takes 10+5+5+5+32767*(11+5)+11= 524308 cycles or 0.262154 seconds with a 2MHz clock.

The shortest solution, or I should say solutions, since there are four, turned out to be six bytes long. The first, and by far the most popular entry consists of two three byte instructions which took longer to run than the previous 8 byte solution.

Address	Code	Mnemonic	number of clock cycles
FFFA	31 FC FF	LXI SP,FFFC	10
FFFD	CD FD FF	CALL FFFD	17

The first instruction sets the stack pointer to FFFC, the byte before the next instruction. The CALL instruction which follows calls itself at location FFFD pushing the return address, which happens to be 0000, onto the stack. With successive CALLs the stack pointer works downwards through memory writing pairs of zero bytes. After the 32767th CALL only two bytes, FFFD and FFFE, remain unaffected so that the last CALL, which overwrites these last two bytes, jumps to location 00FD. Total execution time is 10+32768x17=557066 cycles or 0.278533 seconds with a 2Mhz clock.

The second solution is very similar to the earlier 8 byte program and works in exactly the same way. The extra two bytes are saved by using the HL registers in the PUSH instruction, eliminating the need to load D and E.

Address	Code	Mnemonic	number of clock cycles
FFFC	AF	XRA A	4
FFFD	67	MOV H,A	5
FFFE	6F	MOV L,A	5
FFFF	F9	SPHL	5
0000	E5	PUSH H	11
0001	E9	PCHL	5

Execution time to clear memory is 4+5+5+5+32768x11+ 32767x5=524302 cycles or 0.262151 seconds at 2MHz.

The two remaining six byte solutions are equally 'correct' since they both take the same time to execute.

Address	Code	Mnemonic	number of clock cycles
FFFC	21 00 00	LXI HL,0000	10
FFFF	F9	SPHL	5
0000	E5	PUSH H	11
0001	E9	PCHL	5

This program is very similar to the previous solution and works in exactly the same way. Four clock cycles (2 micro-seconds) are saved by loading HL direct rather than using the accumulator. Execution time to clear memory is 10+5+32768 x11+32767x5=524298 cycles or 0.262149 seconds at 2MHz.

Address	Code	Mnemonic	number of clock cycles
FFFB	21 FF FF	LXI H,FFFF	10
FFFE	F9	SPHL	5
FFFF	C7	RST 0	11
0000	E9	PCHL	5

The first instruction loads HL with address FFFF which is then put into the stack pointer by the second instruction. 'Restart 0' is a single byte subroutine call to location 0000, which since it is being executed at location FFFF pushes a return address 0000 on the stack. The PCHL jumps back to FFFF until after the 32768th Restart instruction all memory is set to zero. Execution time is 0.262149 seconds.

Only two readers sent in both correct solutions and as there seems no fair way of discriminating between them I declare **Eric Baddiley** of Congleton, Cheshire and **David Parkinson** of Ipswich joint winners of PUZZLE DAZZLE No. 1.

Bridge Hands Competition

Sheridan Williams

The PCW competitions seem to have gathered momentum now. The number of entries is staggering. This competition attracted over 200 entries. The age ranges and areas were diverse, but yet I received only one female entry.

This competition was to write a BASIC program to print four Bridge hands. There were two categories for entries. These were: (i) write the Fastest program, and (ii) write the program with the fewest number of statements. Each run should produce different hands of the form North, East, South, and West followed by the cards. I deliberately left the layout undefined. Each entry required the inclusion of a stamped addressed envelope; amazing how many people ignored this request. I always reply personally to each entry, and hence the need for an SAE.

The winner of category (i) was J. Yale of Wimborne, Dorset; who provided a program that ran 20 times in 21½ seconds. This program was a clear winner over the next entries. Creditable entries were received from Chris Why, Paul Shirley, John Steggals, P. J. Andrews, and D. C. Broughton.

The winner of category (ii) was J. T. Steggals, whose program was only 8 statements long. Creditable entries that were only nine statements long were received from A. H. MacDonald-Smith, H. O. Roberts, J. Yale, G. M. Tennant, B. Bloomfield, J. A. L. Clark, and H. N. Dobbs; so as you can see, the winner only won by one statement, and even then he only won because I did not specify the layout.

Here are the two winning programs:-

FASTEST

```
10   DIM P(51)
20   P$=", AC, 2C, 3C, 4C, 5C, 6C, 7C, 8C, 9C, 10C, JC, QC,
     KC"
```

```
25   P$=P$+", AD, 2D, 3D, 4D, 5D, 6D, 7D, 8D, 9D, 10D,
     JD, QD, KD"
30   P$=P$+", AH, 2H, 3H, 4H, 5H, 6H, 7H, 8H, 9H, 10H,
     JH, QH, KH"
35   P$=P$+", AS, 2S, 3S, 4S, 5S, 6S, 7S, 8S, 9S, 10S, JS,
     QS, KS"
40   H$="NORTH      SOUTH      EAST      WEST"
50   P(1)=1
60   FOR C=2 TO 51
70   R=INT (C*RND(1))
80   P(C)=P(R)
90   P(R)=C
100  NEXT C
110  FOR I=0 TO 39 STEP 13
120  PRINT MID$ (H$, I+1,6);
130  FOR C=I TO I+12
140  PRINT MID$ (P$,4*P(C)+1,4);
150  NEXT C
160  PRINT
170  NEXT I
180  END
```

FEWEST STATEMENTS

```
10   P$= "**AC2C3C4C5C6C7C8C9CTCJCQCKCAD2D3D4
     D5D6D7D8D9DTDJDQDKD"
20   P$=P$+ "AH2H3H4H5H6H7H8H9HTHJHQHKHAS2S
     3S4S5S6S7S8S9STSJSQSKS"
25   PRINT "NORTH", "EAST", "SOUTH", "WEST"
30   FOR M=52 TO 1 STEP -1
40   G=INT(M*RND(1)+2)
50   PRINT TAB(40-10*(M-4*INT(M/4)));MID$,2*G-1,2)
60   P$=LEFT$ (P$,2*G-2)+MID$ (P$,2*G+1)
70   NEXT M
80   END
```

This competition has been very difficult to judge because of the number of entries; as I have said, over 200 entries. It was necessary to run *all* of the entries. Some people submitted two or three programs making the task even longer. Judging this competition was aided by the fact that Research Machines supplied me with their "real-time clock board" for the 380Z. I modified the BASIC interpreter so that it could access the clock, and also allowing BASIC to scan the keyboard in a similar way to PET's GET command. Because of Research Machines' kindness I was able to time the programs much more accurately than I could otherwise have done. As users of the PET will know, there are a great many applications and games that require a clock and to users of the 380Z I can recommend its purchase. Advice on the above interpreter and on applications for the board can be obtained by including an SAE to my address at the end of this article.

While on the subject of the 380Z and also considering that I wrote an article on how to make BASIC programs run faster, users of the 380Z might like to note that 8K/9K BASIC allows you to finish a loop with NEXT with no parameter, this results in a 25% increase in speed. Also NEXT X,Y works.

Back to the subject of judging the competition. I had a great many people who asked, "How are you going to time category (i)." Some people even suggested that I would not be able to judge the competition fairly because of the difference in times between a teleprinter and a VDU, or even a lineprinter. Well, the scheme devised, hopefully, overcomes the problem, and should have occurred to those who posed the question. Each program was run as submitted to see if it worked. If it worked it went through to the next round. I only received three that didn't work, surprising when you consider how many entrants admitted to having no computer on which to test their programs. Round two consisted of dividing the programs into the two categories, category two being put aside for judging later. Category one (the fastest program) proved to be the most time consuming because each of the programs had to be modified in the following ways: ▶

1. A loop of 20 runs was put in to make the timing more accurate, and also the clock functions had to be put in.
2. Each PRINT statement was replaced by a 'buffer' statement. The purpose of this 'buffer' statement was to remove the differences in times caused by differing PRINT formats. For example here are three PRINT statements and their replaced statements:

```
PRINT "NORTH SOUTH EAST WEST"
A$ =  "NORTH SOUTH EAST WEST"

PRINT MID$ (P$,4*P(C)+1,4)
A$ =  MID$ (P$,4*P(C)+1,4)

PRINT  TAB(40-10(M-4*INT(M/4)));MID$(P$,2*G-1,2)
     A = INT(40-10(M-4*INT(M/4)))
    A$ = MID$(P$,2*G-1,2)
```

There is no need, in fact it is wrong, to print the buffer lines. There was a great range of times for the programs; from the winning one which ran 20 times in 21½ seconds to one which took over 200 seconds for 20 runs. The average was around 30 seconds. Some of the output layouts were poor, but as the output format was undefined this made no difference to their final positions.

As an example of programming techniques, and the ability to make a program faster I would like to show a program submitted by Mr. R. Ross which took 50 seconds for 20 runs. After applying some of the techniques outlined in my previous article on faster BASIC programs, it took only 24 seconds, over 100% improvement.

```
10 DIM A$(1,53)
30 FOR X=1 TO 53
32      READ A$(1,X)
34 NEXT X
40 FOR Z=1 TO 25
50      LET X=INT(51.9999*RND+1)
60      LET Y=INT(51.9999*RND+1)
70      LET A$(1,53)=A$(1,X)
80      LET A$(1,X)=A$(1,Y)
90      LET A$(1,Y)=A$(1,53)
95 NEXT Z
```

I have no idea at all why a 2-dimensional array is used, but it certainly takes longer to reference an element in a 2-D array than it does for a 1-D array, so these were all changed. A$(1,53) is only used to hold a temporary quantity during the swapping process, and could easily be allocated a non-array variable, for example T$, with a consequent time saving. The RND function returns a value x within the range $0 < x < 1$ and hence there is no need for the number 51.9999 to occur, 52 will do. As 52 is an integer quantity the expression will be evaluated faster than using a non-integer. The following program incorporates all my suggested improvements and results in a program which runs *twice* as fast. I am most grateful to Mr. R. Ross for allowing me to tear his program to pieces in rather a cruel way.

```
10 DIM A$(52)
30 FOR X=1 TO 52
32      READ A$(X)
34 NEXT X
40 FOR Z=1 TO 25
50      LET X=INT(52*RND+1)
60      LET Y=INT(52*RND+1)
70      LET T$=A$(X)
80      LET A$(X)=A$(Y)
90      LET A$(Y)=T$
95 NEXT Z
```

Note also that some interpreters work faster with each numeric constant held as a variable. Consult John Coll's benchmark tests to see whether your machine would benefit from that alteration. (The benchmark tests were given in Vol. 1 No. 1, and the timings again in the November 1978 issue).

In order to restrict the entries to future competitions, I shall be setting harder problems to solve.

Competition – Magic Squares

We have all met the magic square

2 7 6
9 5 1
4 3 8

where each row, column, and diagonal all add up to 15. There are magic squares of other orders than 3 x 3; but I will concentrate this competition on 3 x 3 squares only. Look at the following magic square

68 76 57
56 67 78
77 58 66

the row, column, and diagonal total (which I shall call the 'magic constant') is 201. There is something even more 'magic' about this square — when the digits are reversed the square is still magic —

86 67 75
65 76 87
77 85 66

MC = 228. Unfortunately four of the original numbers (66, 67, 76, 77) occur again in the reversed square.

The competition is to find a magic square that satisfies the following conditions:-

1. The square is comprised of 2 digit numbers (zeroes not allowed)
2. It is a 3 x 3 square
3. When the digits are reversed, another magic square is produced with none of the original numbers reappearing.
4. The sum of the magic constants is less than 200.

Solution is to be by BASIC program. A £10 prize will be awarded to the *first* correct entry supported by the program.

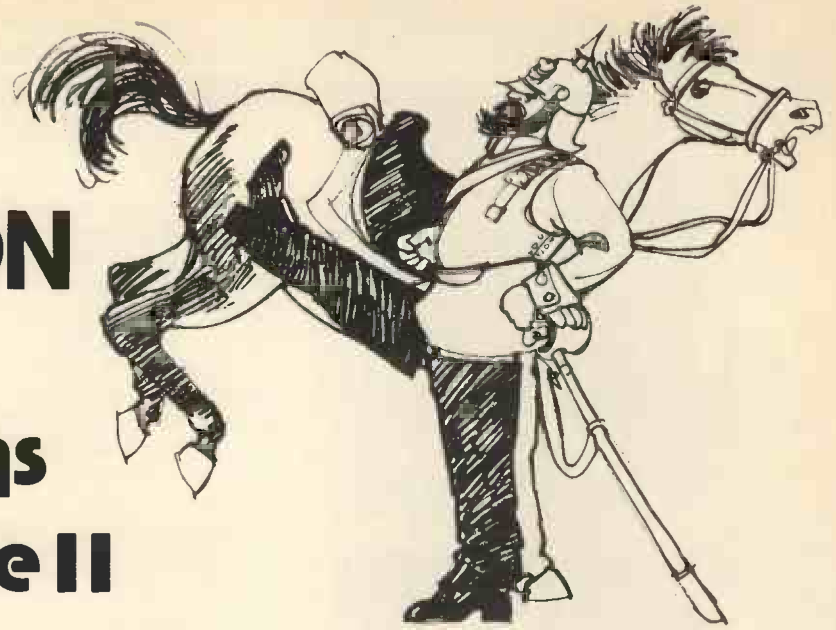
Entries to Sheridan Williams, 114, Beech Road, St. Albans, Herts AL3 5AU. Enclose an SAE if you require the program returned.

David Levy is setting up a company to develop intelligent software for sophisticated microprocessor applications. Any readers interested in writing assembler programs on a contract basis please write to David Levy, Box 123, Personal Computer World, 62a Westbourne Grove, London W2. Anyone actively interested in speech recognition, music composition or robotics is also invited to apply.

★★★★★★★★★★★★★★★★

TRUMPET, VOLUNTARY
Responding to TIDBITS information or an advertisement?
Say you saw it in PCW
Blow your own trumpet.
Mention your magazine.

★★★★★★★★★★★★★★★★



APPLEVISION

Demonstrations using the Apple II

G. F. FILBEY
School of Physics, Polytechnic of the South Bank, London

Micro computers can be used in education in various ways, most usually by means of an interactive program. A particular skill is taught by question and answer. However, another use to which the computer can be put is to simulate processes, not for the purpose of solving problems but for a visual impact which cannot really be achieved in any other way. Cine film is useful but it always presents a mere picture whilst the computer can present a close analogue of the physical process.

For this purpose random processes are the ideal choice. A random number generator is used here; a sequence is obtained by adding together seven hexadecimal bytes to get the next byte. In the machine code of an eight-bit processor such as the 6502 in the Apple II which I used for these programs, this is easy to do modulo 256. However, the cycle length in this modulus is too short so we convert to modulo 255. This is done by an easy trick. You just keep adding the high and low bytes together until the high byte is zero. The result is a number mod. 255 — which means the remainder obtained by dividing the original number by 255.

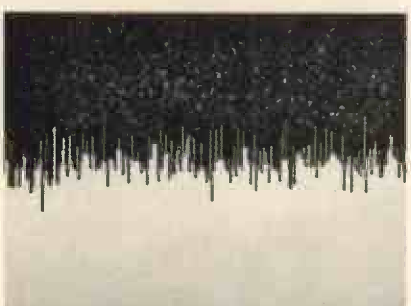


Figure 1

To express, for example, a decimal number as a number mod. 9 you add the digits.

196 →1 + 9 + 6 = 16 →1 + 6 = 7.

∴ 196 mod. 9 = 7.

You can check this process by using the mod. command in integer BASIC.

The programs are written in machine code (they take all night in BASIC!) and the random number generator resides in locations 11ED to 122A. You can experiment with your own subroutine. If you have an Apple, or access to one, and haven't tried machine programming, then the resident mini assembler is the thing to start on. You'll learn by talking to the machine with it just as I did. But mind, it spits back at you. A tip about the mini. You are bound to discover you've missed out some instruction and there doesn't seem to be a way of getting it back in. However, you'll find a simple routine for shifting the whole block of instructions from the missing one downward out of the way; inserting the instruction, and then putting the block back. The routine is in the red book. You've probably got a bit of tidying up to do with branch instructions but it is better than retyping the whole thing. And, by the way, get a book on touch typing and practise right from the start. Nothing looks more ridiculous than a computer wizard sitting at a sophisticated machine with fists partially clenched and a pair of index fingers poking away!

The programs listed here make use of the machine subroutines of the HIRES demo tape produced by Apple. You need to load this tape, list it, and alter the instruction in location ØCØ8 to LDA CØ52 to give a full screen of graphics.

The Demonstrations

Now for the demonstrations. The first is illustrated in fig. 1. Bins from zero to 254 are loaded with random numbers as they come up. Some 50,000 numbers are involved. The X axis represents the random number X and the Y axis the incidence of that number. You can see the sort of scatter obtained in 50000 trials. Repetitions, by hitting key R, produce different pictures showing that the generator is not in a short loop.

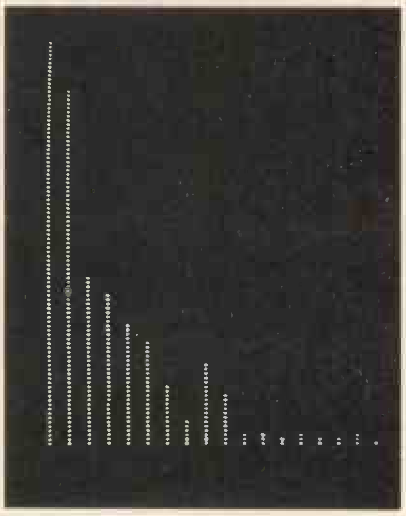


Figure 2

The second demonstration shows the Boltzmann distribution (figs. 2 and 3). This distribution illustrates energy distributions of molecules and is, in fact, the best-guess distribution

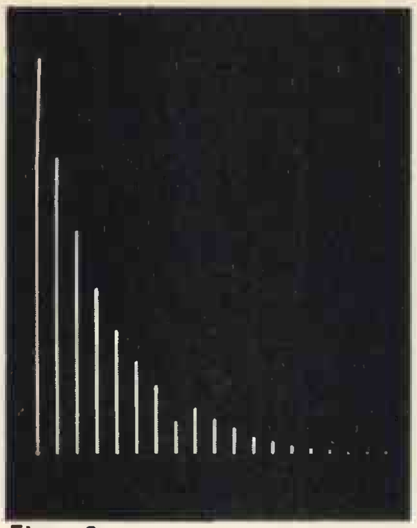


Figure 3

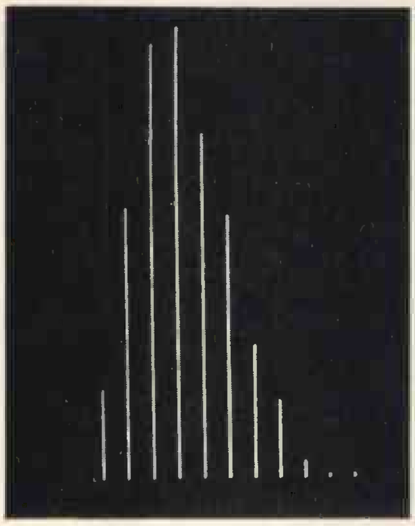


Figure 4

of a system when the average value of a parameter is known.

As a simple example suppose you throw a die repeatedly. What is the probability that you get a six on, say, the 10th throw and not before? You assume two things.

1) The six must come up eventually i.e. the probability that it comes up in an infinite number of trials is unity.

2) The expectation value for the number of times it comes up is n/6 where n is the number of trials (the fair die assumption).

To show the difference in quality of the distribution function when a small or large number of trials is used the demonstration first produces fig. 2. The top of the picture (Y max) represents 95 i.e. a success was scored on the first throw in 95 of the trials. (We are here using a die with three faces!).

Fig. 3 shows the improvement when the top of the screen represents 50,000. Fig. 3 also reveals a fault in the random-number generator. Bin 7 is low. (Counting the first bin as bin zero). It is consistently low. This is due to the fact that seven numbers are added to produce the random number. It is as if you were using a die for which it was unlikely to score a six on the seventh throw after a six last came up! However for most applications this error is unimportant and its consistency permits it to be allowed for. A fourth-order additive system would here put this error in an awkward place.

The next demo is a Poisson distribution. This occurs in the case of a very large number of trials and a very low probability of success. When all the probable causes of human death, including old age, have finally been conquered you are pretty sure to die of something unlikely! The classic poisson distribution concerns the incidence of death among officers of the Prussian army by being kicked by their horses, for which actual records were kept (I am told the converse was a common occurrence).

The Rapid Fall-Off

In this demo 765 trials are made of the occurrence of a random number which has a probability of 1/255 of coming up. This gives an expectation value of three successes. The first bin represents the number of times that no success occurred in a succession of groups of 765 trials. Nearly 3/4 of a million random numbers are used here.

A characteristic of this distribution is the extremely rapid fall-off. With an expectation value of 3 successes the probability of 10 successes is negligibly small. The formula for this distribution is

$$f_n = \frac{\lambda^n \exp(-\lambda)}{n!}$$

Where f_n is the probability of n successes and λ is the expectation number. You can test the distribution

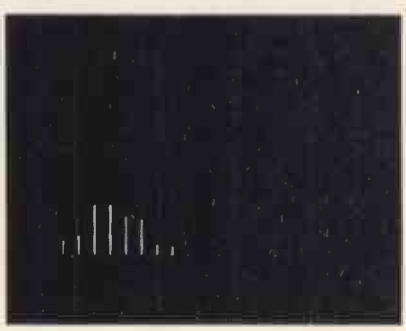


Figure 5

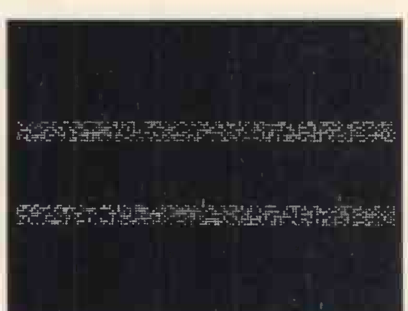


Figure 6

shown against this formula. Notice it should give equal answers for 2 and 3 successes. They are nearly equal on the histogram shown. The ratio of two consecutive values of f_n comes out as 3/(n + 1) which checks up reasonably with the values in fig. 4.

Fig. 5 shows the histogram in its early stages. Better results still can be obtained by, say, sets of 3000 trials with a 1/1000 probability of success. This would take at least 15 minutes to display.

Finally fig. 6 shows a simulation with random numbers. A close-spaced set of dots in parallel lines represents the valence band of a semiconductor (this is for physics students).

A random number is used to punch a hole in the valence band (i.e. to put a black dot amongst the white) and simultaneously to put an electron (a white dot) at a fixed height above it. This represents absorption of a photon. A yes-no random number decides whether this or the reverse will happen. The reverse represents stimulated photon emission.

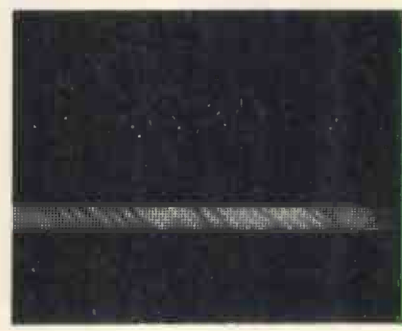


Figure 7

However, the reverse process does nothing at all unless there is an excited state (white up — black down) to de-excite. If this was the only process (excitation and equally probable de-excitation) then the population of white dots up and white dots down would quickly equalise. We would say that the electron temperature was infinite. This is shown in fig. 6. However, in practice the electron temperature is kept down, unless the photon irradiation is very intense, by being cooled by the atoms of the substance. This is called photon induced recombination and a variable recombination rate (again on a random basis) is fed in during the program. Fig. 7 shows the case of a very high recombination rate.

In concluding this article I should like to thank the "Lion House" of Tottenham Court Road, London, who kindly, on a couple of occasions gave me a print-out of the programs, without which editing would have been difficult. The final print-out which you see in this article was provided by my friends at Personal Computers Ltd., and was done on an Axiom printer. I must have one of these. It is the best thing I have seen for the home user.

Appledem

```
  1 NUMBERS=4186:POISSON=4096:PHOT=
    4395
  2 NEGEX=4226
  3 FOR I=1 TO 7
  4 POKE I,I
  5 NEXT I
  9 POKE 106,0
 10 CALL -936
 20 VTAB 10
 30 PRINT "HISTOGRAM OF RANDOM NUMBE
    RS."
 40 PRINT : PRINT
 50 PRINT "HIT ANY KEY TO START."

 60 X= PEEK (-16384)
 70 IF X<127 THEN 60
 80 POKE -16368,0
 90 CALL NUMBERS
 95 FOR I=1 TO 1000: NEXT I
100 CALL -936
110 POKE -16303,0
120 VTAB 10
130 PRINT "HIT  R  TO REPEAT OR ANY
    OTHER KEY"
140 PRINT "TO CONTINUE."
150 X= PEEK (-16384)
160 IF X<127 THEN 150
170 IF X=210 THEN 80
180 POKE -16368,0
182 CALL -936
184 VTAB 10
185 TAB 8
186 PRINT "BOLTZMANN DISTRIBUTION."

188 PRINT : PRINT "IF A SUCCESS NEED
    S ON THE AVERAGE 'A'"
190 PRINT "TRIES,WHAT IS THE CHANCE
    THAT IT WILL"
192 PRINT "OCCUR ON EXACTLY THE 'N'T
    H TRY?"
194 PRINT "THE NO.OF TRIALS FOR SUCC
    ESS IS PLOTTED HORIZONTALLY AND
    THE SUCCESS FREQUENCY  VERTICALL
    Y."
200 PRINT
202 PRINT "THE RESULT OBTAINED WHEN
    THE NUMBER OF  TRIALS IS SMALL I
    S SHOWN 5 TIMES IN"
203 PRINT "SUCCESSION, THEN THE IMPR
    OVEMENT WHICH  RESULTS FROM 500
    TIMES THAT NUMBER IS"
204 PRINT "DISPLAYED": PRINT
205 GOSUB 1000
206 VTAB 10: PRINT "HIT R TO REPEAT
    OR ANY OTHER TO CONTINUE"
207 X= PEEK (-16384)
208 IF X<127 THEN 207
209 IF X=210 THEN 205
210 POKE -16368,0: CALL -936: POKE
    -16303,0
220 PRINT "POISSON DISTRIBUTION...ME
    AN VALUE=3."
230 PRINT : PRINT
240 PRINT "EACH POINT PLOTTED REQUIR
    ES 750 TRIALS"
250 PRINT "AND EACH TRIAL HAS 1/250
    CHANCE"
270 PRINT "OF SUCCESS."
280 PRINT : PRINT
290 PRINT "NOTE THE IMPROVEMENT IN S
    HAPE AS THE"
295 PRINT "NUMBERS INCREASE."
296 PRINT : PRINT
300 PRINT "HIT ANY KEY TO CONTINUE"

310 X= PEEK (-16384)
340 IF X<127 THEN 310
360 POKE -16368,0
370 CALL POISSON
375 FOR I=1 TO 1000: NEXT I
380 CALL -936
390 POKE -16303,0
400 PRINT "NOW FOLLOWS A SIMULATION
    OF THE PROCESS"
410 PRINT "OF PHOTON ABSORPTION IN A
     SEMICONDUCTOR."
420 PRINT "THE LOWER BAND OF CLOSELY
    -SPACED DOTS"
440 PRINT "REPRESENTS THE VALENCE BA
    ND."
450 PRINT : PRINT "ELECTRONS ARE EXC
    ITED TO THE CONDUCTION"
460 PRINT "BAND,AND EQUALLY PROBABLY
     DE-EXCITED"
470 PRINT "BACK TO THE VALENCE BAND.
    "
480 PRINT "THIS IS CALLED 'STIMULATE
    D EMISSION'."
490 PRINT : PRINT "THERE IS ALSO AN
    ADDED RATE OF"
500 PRINT "SPONTANEOUS EMISSION WHIC
    H IS VARIED"
510 PRINT "THROUGH THE PROGRAMME."

515 PRINT "IN THE THIRD STAGE THIS R
    ATE IS ZERO"
520 PRINT "SO THE POPULATIONS EQUALI
    SE."
530 PRINT : PRINT "ALL THESE PROCESS
    ES ARE CONTROLLED BY"
540 PRINT "INDEPENDENT SETS OF RANDO
    M NUMBERS."
560 PRINT : PRINT "HIT ANY KEY TO CO
    NTINUE."
570 X= PEEK (-16384)
580 IF X<127 THEN 570
585 POKE -16368,0
590 POKE 80,254
592 POKE 129,255: POKE 130,85: POKE
    131,3: POKE 132,85
```

```
 600 CALL PHOT
 605 CALL -936
 607 POKE -16303,0
 670 TAB 15: VTAB 10
 680 PRINT "THE END"
 685 FOR I=1 TO 2000: NEXT I
 688 CALL -936
 690 END
1000 PRINT "HIT ANY KEY TO CONTINUE."

1005 X= PEEK (-16384)
1010 IF X<127 THEN 1005
1020 POKE -16368,0: CALL -936
1030 POKE 4262,234: POKE 4263,234
     : POKE 4264,234
1040 CALL NEGEX
1050 FOR I=1 TO 2000: NEXT I
1070 P=P+1
1080 IF P<5 THEN 1040
1085 P=0
1090 POKE 4262,208: POKE 4263,224
     : POKE 4264,232
1100 CALL NEGEX
1110 FOR I=1 TO 1000: NEXT I
1120 POKE -16303,0
1130 RETURN
1000-   20 00 0C    JSR   $0C00
1003-   20 46 10    JSR   $1046
1006-   A9 FF       LDA   #$FF
1008-   8D 2C 03    STA   $032C
100B-   EA          NOP
100C-   A9 00       LDA   #$00
100E-   A8          TAY
100F-   85 50       STA   $50
1011-   85 51       STA   $51
1013-   85 69       STA   $69
1015-   EA          NOP
1016-   8D 12 03    STA   $0312
1019-   20 ED 11    JSR   $11ED
101C-   C9 25       CMP   #$25
101E-   D0 03       BNE   $1023
1020-   20 DB 10    JSR   $10DB
1023-   18          CLC
1024-   A9 01       LDA   #$01
1026-   65 51       ADC   $51
1028-   85 51       STA   $51
102A-   A9 00       LDA   #$00
102C-   65 50       ADC   $50
102E-   85 50       STA   $50
1030-   C9 03       CMP   #$03
1032-   D0 E5       BNE   $1019
1034-   A6 69       LDX   $69
1036-   EA          NOP
1037-   DE 00 0B    DEC   $0B00,X
103A-   BD 00 0B    LDA   $0B00,X
103D-   AC 12 03    LDY   $0312
1040-   20 7C 0C    JSR   $0C7C
1043-   4C 51 10    JMP   $1051
1046-   A9 C0       LDA   #$C0
1048-   A2 00       LDX   #$00
104A-   9D 00 0B    STA   $0B00,X
```

To be Continued

WHEELER DEALER

ROULETTE IN KEN'S CASINO

Ken Wheeler

This is a simplified form of the now well-known Roulette casino game, written in Processor Technology's Cassette Extended BASIC, tested, debugged and run on a SOL-20.

The program is, however, straight-forward and will be easy to adapt to another BASIC dialect. A full listing is given.

Lines 140, 1170, 1280 and 1520 contain a PRINT "&K" statement. This is a control statement for the SOL-20 which simply clears the screen and can easily be substituted or even left out if necessary. The DIM statement in line 150 is peculiar to PTC BASIC and specifies the number of digits which may be contained in the quoted string. In most BASICs such a statement would be used for subscripted string variables but that facility is not available with PTC's BASIC. There are several PAUSE statements included in the listing and, if your BASIC does not have this facility, it can very simply be replaced by a FOR . . NEXT loop.

There should be nothing else in the program to cause any problems and you should be on your way to losing all your hard-earned cash within seconds of typing the program into your machine.

```
10 REM***                  ROULETTE                     ***
20 REM***
30 REM***           WRITTEN BY KEN WHEELER              ***
40 REM***          BEWDLEY, WORCESTERSHIRE.             ***
50 REM***
60 REM***                  3 FEB 79                     ***
70 REM***
80 REM***          THIS VERSION  CONFIGURED             ***
90 REM***          FOR SOL-20 AND PROCESSOR             ***
100 REM***         TECHNOLOGY CASSETTE  EX-             ***
110 REM***              TENDED BASIC.                   ***
120 REM***
130 REM***
140 PRINT "&K"
150 DIM B1$(45),B2$(10),B3$(10)
160 PRINT "                    WELCOME TO KEN'S CASINO"
170 PRINT
180 PRINT "TODAY'S GAME IS ROULETTE - WOULD YOU LIKE INSTRUCTIONS?"
190 PRINT "TYPE YES OR NO AND PRESS THE RETURN BUTTON.   ";
200 INPUT I$
210 IF I$="YES" THEN 250
220 IF I$="NO" THEN 490
230 PRINT "ANSWER YES OR NO PLEASE!"
240 PRINT : GOTO 200
250 PRINT
260 PRINT "IN THIS VERSION OF ROULETTE YOU ARE ALLOWED TO"
270 PRINT "BET ON INDIVIDUAL NUMBERS, NUMBER SETS, ODD NUMBERS"
280 PRINT "OR EVEN NUMBERS."
290 PRINT "THE NUMBERS RUN FROM 00 TO 36 AND YOU PLACE YOUR BET"
300 PRINT "BY ENTERING A NUMBER BETWEEN 0 AND 44. THE"
310 PRINT "NUMBER REPRESENTATIONS ARE SHOWN BELOW."
320 PRINT "00 IS ALWAYS A WIN FOR THE HOUSE.  IF 00 IS ENTERED"
330 PRINT "AS YOUR NUMBER THIS WILL BE INTERPRETED AS A DECISION"
340 PRINT "TO END THE GAME."
350 PRINT : PRINT "PRESS THE RETURN BUTTON TO CONTINUE.  ";
360 INPUT X$
370 PRINT
380 PRINT" 0 TO 36 REPRESENT THEMSELVES."
390 PRINT "38 REPRESENTS THE NUMBER GROUP 1 TO 18."
400 PRINT "39   ''              ''         19 TO 36."
410 PRINT "40 REPRESENTS EVENS."
420 PRINT "41 REPRESENTS ODDS."
430 PRINT "42 REPRESENTS THE NUMBER GROUP 1 TO 12."
440 PRINT "43   ''              ''         13 TO 24."
450 PRINT "44   ''              ''         25 TO 36."
460 PRINT
470 PRINT "PRESS RETURN TO CONTINUE. ";
480 INPUT X$
490 PRINT
500 LET S=0
510 PRINT "THE BETTING ODDS ARE:-"
520 PRINT "     SINGLE NUMBER BETS :           36 TO 1"
530 PRINT "  GROUPS OF 12 NUMBERS :             2 TO 1"
540 PRINT "  GROUPS OF 18 NUMBERS :         EVEN MONEY"
550 PRINT "          ODDS OR EVENS :         EVEN MONEY"
560 PRINT
570 PRINT "IF YOU WOULD LIKE A PRINT-OUT OF THE TABLE THEN TYPE THE"
580 PRINT "LETTER Y (OTHERWISE TYPE N) AND PRESS "
590 PRINT "THE RETURN BUTTON.  ";
600 INPUT T$
610 IF T$="Y" THEN 640
620 IF T$<>"N" THEN  PRINT "ANSWER NOT UNDERSTOOD - TRY AGAIN!": GOTO
                                                                   600
630 GOTO 840
640 PRINT
650 PRINT
660 LET B1$="    *************************************"
670 LET B2$="    *    "
680 PRINT B1$
690 PRINT "    *    0          *          00    *"
700 LET B3$="    *   "
710 PRINT B1$
720 LET Q=-2
730 FOR I=1 TO 3
740   LET Q=Q+3
750   PRINT B2$;Q;B2$;Q+1;B2$;Q+2;B2$
760   PRINT B1$
770 NEXT I
780 FOR I=1 TO 9
790   LET Q=Q+3
800   PRINT B3$;Q;B3$;Q+1;B3$;Q+2;B3$
810   PRINT B1$
820 NEXT I
830 PRINT
840 PRINT
850 PRINT "HOW MANY POUNDS ARE YOU PREPARED TO GAMBLE TODAY  ";
860 INPUT P
870 LET S=S+1
880 IF S>5 THEN 910
890 PRINT "PLACE YOUR BET (NUMBER 0 TO 44)  ";
900 GOTO 920
910 PRINT "WHAT IS YOUR NUMBER  ";
920 INPUT B
930 IF B=00 THEN  END
940 IF B>37 AND B<42 THEN  LET D=C
950 IF B>41 AND B<45 THEN  LET D=C*2
960 IF B>44 THEN 870
970 IF B<0 THEN 870
980 IF S>5 THEN 1010
990 PRINT "HOW MUCH IS YOUR BET  ";
1000 GOTO 1020
1010 PRINT "AMOUNT    ";
1020 INPUT C
1030 PRINT
1040 IF P=C THEN 1090
1050 IF P>C THEN 1090
1060 PRINT "YOU DON'T HAVE THAT MUCH MONEY!  "
1070 PRINT "THE MOST YOU CAN BET IS ";P;" POUNDS."
1080 GOTO 990
1090 LET R=INT(RND(0)*100)
1100 IF R=37 THEN  LET R=00: GOTO 1140
1110 IF R>37 THEN 1090
1120 IF B>37 THEN 1370
1130 IF B=R THEN 1260
1140 LET P=P-C
1150 IF P>0 THEN 1170
1160 GOTO 1510
1170 PRINT "&K"
1180 FOR J=1 TO 5: PRINT "CLICK! ";: PAUSE 5: NEXT J
1190 PRINT : PRINT
1200 PRINT "THE WINNING NUMBER IS ";R
1210 PAUSE 10
1220 PRINT
1230 PRINT : PRINT "YOU LOST BUT YOU STILL HAVE ";P;" POUNDS."
1240 PRINT
1250 GOTO 870
1260 LET D=36*C
1270 LET P=P+D
1280 PRINT "&K"
1290 FOR J=1 TO 5: PRINT "CLICK! ";: PAUSE 5: NEXT J: PRINT
1300 PRINT
1310 PRINT "THE WINNING NUMBER IS ";R
1320 PRINT
1330 PAUSE 10
1340 PRINT "  **  YOU WON  **  !!  ": PRINT
1350 PRINT "NOW YOU HAVE ";P;" POUNDS.": PRINT
1360 GOTO 870
1370 IF B>39THEN 1640
1380 IF B=38 THEN 1480
1390 IF R>18 THEN 1410
1400 GOTO 1420
1410 IF R<37 THEN 1460
1420 LET P=P-C
1430 IF P>0 THEN 1450
1440 GOTO 1510
1450 GOTO 1170
1460 LET P=P+D
1470 GOTO 1280
1480 IF R>18 THEN 1420
1490 IF R>36 THEN 1420
1500 GOTO 1460
1510 PRINT
1520 PRINT "&K"
1530 FOR J=1 TO 5: PRINT"CLICK! ";: PAUSE 5: NEXT J: PRINT
1540 PRINT "NUMBER ";R;" WON": PRINT
1550 PAUSE 10
1560 PRINT "YOU JUST LOST YOUR LAST POUND . . . . !"
1570 PRINT
1580 PRINT
1590 PRINT "WOULD YOU LIKE TO START AGAIN (Y OR N)   ";
1600 INPUT A$
1610 IF A$="Y" THEN 490
1620 IF A$<>"N" THEN  PRINT "ANSWER NOT UNDERSTOOD - TRY AGAIN.":
                                                           GOTO 1600
1630 GOTO 1880
1640 IF B>41 THEN 1730
1650 LET Y=(R/2)-INT(R/2)
1660 IF B=41 THEN 1690
1670 IF Y>.01 THEN 1420
1680 GOTO 1460
1690 IF R=37 THEN 1420
1700 IF R=0 THEN 1420
1710 IF Y>.01 THEN 1460
1720 GOTO 1420
1730 IF B=42 THEN 1830
1740 IF B=43 THEN 1790
1750 IF R>24 THEN 1770
1760 GOTO 1420
1770 IF R=37 THEN 1420
1780 GOTO 1860
1790 IF R>12 THEN 1810
1800 GOTO 1420
1810 IF R>24 THEN 1420
1820 GOTO 1860
1830 IF R>0 THEN 1850
1840 GOTO 1420
1850 IF R>12 THEN 1420
1860 LET P=C*2+P
1870 GOTO 1280
1880 END
```

STATPACK

Continued

Colin Chatfield

```
LIST  STAT8

0005 REM STAT8
0080 LINE= 80
0090 GOSUB 9380
0100 ? "THIS PROGRAM IS SPARE FOR THE USER TO ENTER HIS PROGRAM"
0110 ? :?TAB(20);"STATPACK"
0120 GOTO 1200
1000 GOSUB 9600
1010 ? "YOUR ARRAY IS ";A;"X ";B;:?CHR$(8);:?". ";A*B;"ITEMS."
1110 INPUT "     CARRIAGE RETURN WHEN READY",A$
1120 GOSUB 5000
1200 GOSUB 9360
1210 IF LEFT$(A$,1)="N"THEN1240
1220 IF LEFT$(A$,1)<>"Y"THEN1140
1230 INPUT "     IF LINEAR REGRESSION INPUT `Y' ",A$
1232 IF A$="Y"THEN1120
1234 CHAIN STAT1
1240 ? TAB(20);"STATPACK END": END
5000 REM THIS IS A SPARE PROGRAM WHICH CAN BE USED BY THE USER
5010 ? "THIS PROGRAM IS SPARE AND HAS NOT BEEN WRITTEN ":RETURN
9000 REM SUB PROGRAMS
9300 FOR K=1TO5:?#(Z9):NEXTK:RETURN
9360 INPUT "     ENTER `Y' FOR MORE, `N' FOR NONE ",A$
9370 RETURN
9380 ? CHR$(25);:?CHR$(25);:?CHR$(22);
9385 ? CHR$(12);:RETURN
9400 B2=0:IFB=1THENB2=1:GOTO9430
9410 ? :INPUT"     COLUMN # STATISTICS REQUIRED FOR ",B2
9420 IF B2>B THEN?"TOO HIGH";:GOTO9410
9430 ? #(Z9):RETURN
9600 OPEN #10, STATFL1 FOR INPUT
9610 OPEN #20,STATFL2 FOR INPUT
9620 FIELD #10,F=6
9630 FIELD #20,A=6,B=6
9640 SET #10=1:SET#20=1:GET#20
9650 DIM C(A,B),B3(A)
9660 FOR I=1TOA:FORJ=1TOB
9670 GET #10:C(I,J)=F
9680 NEXT J:NEXTI
9683 INPUT "     ENTER `Y' FOR VISUAL OF DATA",A$:IFA$<>"Y"THEN9690
9685 ? :FORI=1TOA:FORJ=1TOB:?C(I,J);:NEXTJ:?:NEXTI:?
9690 CLOSE #10:CLOSE#20:RETURN

THE GHOST
```

WITBIT

David Parkinson & Graham Trott

The assembly language problem forum

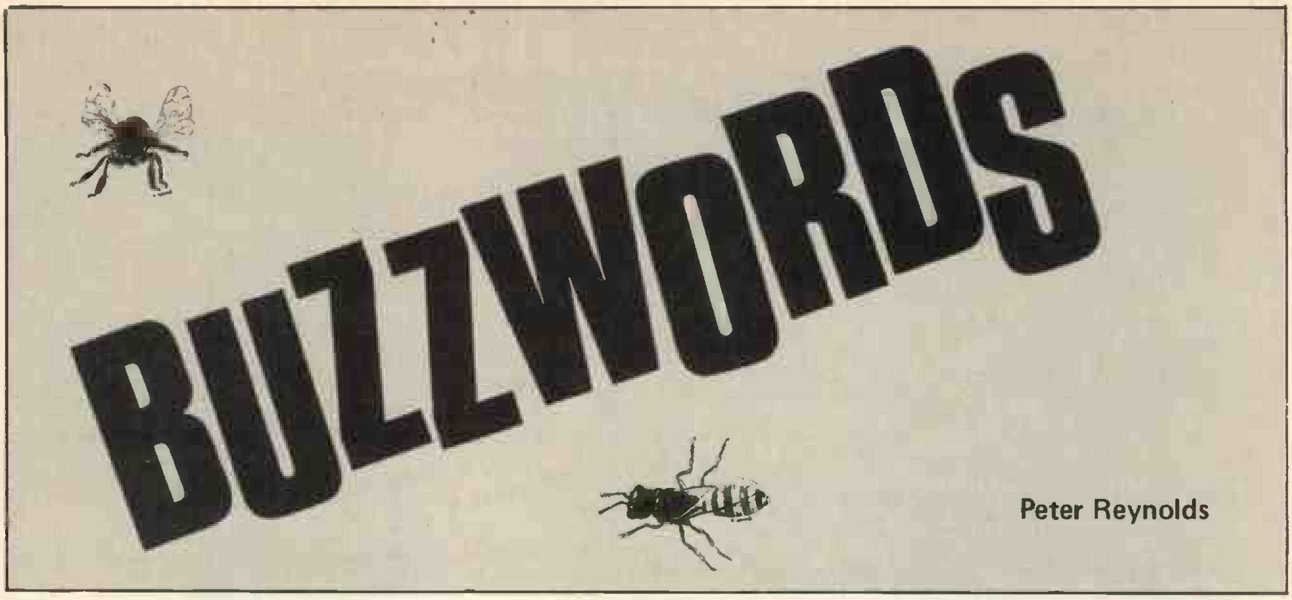
The aim of this column is to set small problems, not necessarily complex or obscure, which will provide those of you who program in assembly language with an opportunity to exercise and develop your skill. The published solutions will illustrate one or more approaches to the problem and we hope that in time the pages will form a useful reference library of small subroutines.

Solutions should be in Z80 or 6800 code, and prizes of £10 and £5 will be awarded in each section. Entries for this month's competition should be in by the end of June. Results in August issue.

Problem: To write a short subroutine for an editor to execute a "Find string" command.

Background: The variable SKEY points to a delimiter at the start of an ASCII character string. The string is terminated by an identical delimiter. The variable CURR points to the first character of the current line of an editor source file. The file is stored as:- ascii line (CR) ascii line (CR) . . . ascii line (CR) (NULL)

Required: To search the file from CURR in an attempt to match the string pointed to by SKEY. On finding it CURR is set to point to the start of the line containing the string, otherwise CURR points to the end-of-file marker. Any "@" in the search key represents a "don't care" character.



BUZZWORDS

Peter Reynolds

Ga As. Gallium Arsenide — descriptive element of some *light emitting diodes* and *field effect transistors.*

Gain. Measure of amplification in an electronic circuit, usually expressed either as a direct multiplier (e.g. 1000 times) or in decibels.

Game Theory. Arithmetical process of selecting the best strategy against an opponent who has a strategy of his own. It is an inherent characteristic of game theory that decisions of one contestant alter the opportunities of others.

Ganged (controls). Physically connected so that they move together, though not necessarily in the same direction. A pair of ganged switches might close one circuit at the same instant as another circuit is opened.

Gap (1). The gap in a read/write head for some magnetic storage peripheral (for example, tape, or disc) is an unchanging physical fact built in by the manufacturer which will not concern a computer user so long as he keeps the heads clean. It is virtually the same as in a domestic sound recorder.

(2). The space between a read/write head and the magnetic surface it serves. Where that surface is constantly in motion, for example, on disc, physical contact would cause undue wear and the head is, therefore, kept slightly clear. The gap should be kept small and constant.

(3). Interval between successive blocks of data stored, for example, on magnetic tape, where it provides opportunity for acceleration and deceleration of the transport mechanism.

Garbage. A meaningless jumble of characters sometimes found (after a mishap) in a computer's data files or output.

Gate. An electronic switch, typically found in an integrated circuit and arranged to respond logically to a particular pattern of signals presented simultaneously to a number of input connections.

Gating. Electronic switching of a circuit, typically under the control either of a clock (opening and closing the gate periodically) or of logical circuits responding to specific patterns of input signals.

Generalised Routine. A computer program written to a specified structure (for example, to extract information from a magnetic tape file of customers' balances) but which has been designed so that the parameters can be varied very easily. A general program for sales ledger analysis, for example, might on one occasion be set up to note only slow moving accounts and on another occasion, by minor modification to the initial instructions, the same program could be used to extract only details of balances which had moved within the last few days. There is sometimes very little distinction between a general routine and a general sub-routine which has been designed to be used in a considerable variety of programs.

Generate. To compose a computer file (for data or program) semi-automatically, given a number of initial parameters.

Generated Address. An address composed out of quantities which arise in the operation of a program. A *modified address or relative address* can be a form of generated address, but a generated address may also be one determined without any reference to a given base or absolute address.

Giga. Prefix signifying one thousand million 10^9.

GHz. Abbreviation for GigaHertz.

GigaHertz. A frequency of 10^9 cycles per second.

Gigo. Garbage In Garbage Out (acronym). A reminder that computers do not think and can produce nonsense on a massive scale if fed with inappropriate data.

Golf Ball. A particular design of printing head for an electric typewriter or sequential printer, so called because the type face is embossed on the surface of a metal sphere about the size of a golf ball.

Gosub. A program instruction (e.g. in BASIC) which transfers control to a sub-routine, written elsewhere in the program, and afterwards (by a RETURN instruction) allows the program to be resumed from the point of diversion. In this way a single subroutine may be invoked many times by different Gosub (Go to SUBroutine) instructions.

Goto. A program instruction (e.g. in BASIC) which transfers control to another part of the program.

Graceful Degradation. Capability in a computer on breakdown, such as power failure, to close its activities in an orderly way so that minimum damage is done either to the machine or to the work being processed. Synonymous with *fail soft.*

Grandfather-Father-Son. An extension of the *Father-Son* technique of preserving a copy of the earlier record whenever a new version is prepared. As the title suggests, each tape or other record is kept for an extra generation (or updating cycle) before its data is destroyed; in the Father-Son payroll example data for week 39 would be retained until records for both weeks 40 and 41 had been completed and found free from error.

Graphics. The use, particularly on a computer's display screen, of graphs, bar charts and other designs in place of or in addition to the usual alphanumeric character set.

Graphics Mode. The setting of an adjustable device, such as a daisy wheel type of sequential printer, so that graphs may be drawn on the paper where characters would normally be printed. 'Graphics mode' implies that both print head and paper will be moved by a very small distance (and in either direction) between each imprint of a dot or other chosen plotting character.

Graph Plotter. A computer output device which can draw lines (sometimes in several different colours) on paper under computer control.

Gray Code. A binary code drawn up in such a way that in going from one decimal value to the next only one binary digit changes its value, for example:

Decimal	*Binary*	*Gray*
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110

Synonymous with a cyclic code.

Ground. An Americanism for *earth* (as an electrical connection).

Group Mark. A code used to identify the limits of data treated as a single group.

Guard Band. An area capable of carrying information but deliberately left unused in order to prevent accidental interaction between signals carried in neighbouring bands.

Guard Signal. A signal provided in a central processor to prevent the reading of any area of store while it is in the process of change.

Guillotine. A device for cutting paper accurately, normally many sheets at a time; sometimes (instead of a *burster*) to separate forms printed on continuous stationery into individual documents.

NASCOM
FLOATING POINT

BASIC LEVEL C Runs on an 8K system
- Full floating point arithmetic
- Software compatible with Levels A and B
- Contains all features found on A and B plus more
- Line editor, 52 variables
- Commands include:— FOR; NEXT; READ; DATA; PEEK; POKE; IN; OUT; USR; INKEY

BASIC LEVEL A Runs on the minimum Nascom
- No extra memory needed
- Fitted in 2 minutes in place of monitor
- Integer arithmetic < , > , < > , < =, > =, =
- Fast execution time
- Keyboard pause and interrupt
- Abbreviated commands allow economical use of your memory, and consist of:— LET; PRINT; IF; GOSUB; RETURN; INPUT; REM; STOP; GOTO; LIST; NEW; RUN; SAVE; ABS; RND

BASIC LEVEL B As previously advertised

LEVEL A or B in 2x2708 PROMS	£21.50
LEVEL B Cassette	£ 7.50
LEVEL C in 4x2708 PROMS	£42.00
LEVEL C Cassette	£12.50
Partial disassembler/debug	£ 5.00
Superstartrek for Level B or C	£ 4.50
Relocator package	£ 4.00

All products fully documented

C C SOFT (Southfields)
83 Longfield St. London SW18
Tel: 01-870 4891 (Anytime)

Callers by appointment please

LET'S GET DOWN TO BUSINESS!

ACCOUNTS RECEIVABLE/ ACCOUNTS PAYABLE, INVENTORY, LEDGER, WORD PROCESSING, INFORMATION HANDLING, WHATEVER.

Ohio Scientific's full lineup of small business computer systems can handle any part of your business. Or all of it. Quickly, accurately, and economically. Each model costs less at the start. Nice for small budgets. And using a low-cost building-block approach, Ohio Scientific's computers can grow with your needs. Simply, you get the most for your money, right from the start. There's a full library of easy-to-use business programs, too.

See this **OHIO SCIENTIFIC** dealer today!

MILLBANK COMPUTING
2, EAST LANE, KINGSTON UPON THAMES, SURREY
KT1 2NN ENGLAND TELEPHONE.01 549 7262 TELEX.8951525

XITAN SYSTEMS

Southern England stockists for **CROMEMCO**

Selected nationwide by Government departments, Research Establishments, Universities, Professional Consultants, Assurance Companies, Industry and the serious hobbyist. Features include a range of software equalled by no other manufacturer for price and performance. Fortran, Basic, Cobol, Z-80 Macro Assembler, Trace, and Word Processing, all at only £85 each. Xitan Systems can supply CP/M for use on Cromemco equipment with 8 in. floppy discs.

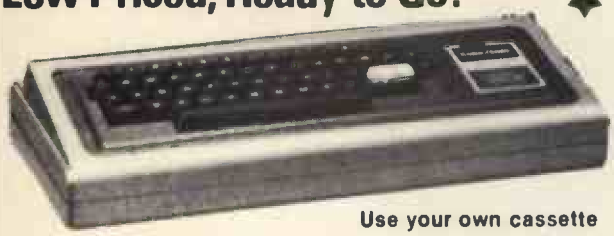
THE machine for professional use

XITAN SYSTEMS,. 23 CUMBERLAND PLACE, SOUTHAMPTON SO1 2BB. TEL: (0703) 38740

Also suppliers of: PET • HORIZON • PT SOL • DYNABYTE.MEMORY • BOOKS • VDUs • PRINTERS

JUST COMPARE OUR CASH AND CARRY PRICES!

THE TRS-80 *(SPECIAL SCOOP)*
Low Priced, Ready to Go!



Use your own cassette

Level-II with 4K RAM
Improved graphics, print formatting, and a faster cassette transfer rate are features of Level-II BASIC.

£399 + VAT

PLUGS INTO YOUR OWN TV

Level-II with 16K RAM
A combination of 16K RAM and the powerful Level-II BASIC produces a system capable of handling most demands.

£499 + VAT

FREE 100 PROGRAMS

KEY BOARD ONLY	COMPLETE WITH UHF MODULATOR

UK POWER SUPPLY — £9.90 + VAT

ATTENTION! SAVE £90
TRS 80 & APPLE II USERS

Simple to fit - only a screw-driver is required

16K UP GRADE KIT — HALF NORMAL PRICE £99 + VAT — LIFETIME GUARANTEE

NEW Standard QWERTY Keyboard with 16K Memory

only £625 + VAT

The No. 1 in the U.K. Affordable

8K PET still at unbeatable price of £499 + VAT

NEW! THE ITT APPLE (2020)

EX STOCK 4K

LOW PRICE MEMORY UPGRADE PURCHASED AT TIME OF ORDER

16K £49 + VAT

* Full colour · UHF output * Audio cassette tape interface * Up to 48K RAM on board * BASIC in ROM (graphics commands include COLOUR = VLIN, HLIN, PLOT and SCRN) * Built in Loudspeaker *Buckets of software already available *disk system (110 K byte per drive — includes controller) only £425 + VAT

THE EXIDY SORCERER.

SAVE

SORCERER COMPUTER SYSTEM

CPM NOW

The Sorcerer Computer is a completely assembled and tested computer system. Standard configuration includes 63-key typewriter-style keyboard and 16-key numeric pad, Z80 processor, dual cassette I/O with remote computer control at 300 and 1200 baud data rates, RS232 serial I/O for communications, parallel port for direct Centronics printer attachment, 4K ROM operating system, 8K ROM Microsoft BASIC in Rom Pac™, cartridge, composite video of 64 char/line 30 line/screen, 128 upper/lower case ASCII set and 128 user-defined graphic symbols, operation manual, BASIC programming manual and cassette/video cables, connection for S-100 bus expansion.

LOOK!
*32K RAM on board
*RS232 interface *8K BASIC ROM
*CUTS interface *4K MONITOR
*KANSAS CITY interface *S100 BUS
*User defined graphic symbols *Z80 cpu

EXIDY SORCERER 32 RAM COMPUTER **£790** + VAT

BOOK SHELF

LOW PRICES FAST SERVICE!

Vol 0 The Beginner's Book	£5.40
Vol 1 Basic Concepts +	£5.90
Vol 2 Some Real Products (June 1977)	£9.90
Microprocessor series, by Rodney Zaks C201 From Chips to Systems	£7.50
C207 Interfacing Techniques	£7.50
Microprocessor Systems Design by Edwin Klingman ISBN 0-135-81413-8 (1977) 480 pages hard cover	16.40
CMOS Cookbook by Don Lancaster	£6.90
6800 Assembly Language Programming	£6.50
6800 Software Guide & Cookbook from Scelbi	£7.90
8080 Software Guide & Cookbook from Scelbi	£7.90
8080 A/8085 Assembly Language Programming	£6.50
8080A Bugbook Interfacing & Programming by Rony, Larsen & Titus (1977) 416 pages	£6.90
Z80 Programming for Logic Design by Adam Osborne	£5.90
Z80 Microcomputer Handbook by William Barden	£7.20
Structured Programming & Problem Solving with Pascal ISBN 0-138-54869-2 (1978) 365 pages by Kieburtz "When will your microcomputer speak Pascal"	£9.50
BASIC Computer Games — Microcomputer edition	£5.40
Best of BYTE Vol 1 (1977) 376 pages	£8.90
Best of CREATIVE COMPUTING Vol 1 (1977) 326 pages	£6.90
Vol 2 (1977) 323 pages	£6.90
MOI OTHELLO Game for 2 players on NASCOM 1 cassette	£2.00
MOI MINO PILOT powerful text editor and interpreter in ½KB!	£5.00

COMP COMPUTER COMPONENTS
(PART OF THE COMPSHOP LTD. GROUP)

For technical and sales literature send SAE to: COMP COMPUTER COMPONENTS (Exidy UK Sales), Freepost, 14 Station Road, New Barnet, Herts. Tel: 01-441 2922 (Sales) 01-449 6596 Telex: 298755

TAKE ADVANTAGE OF OUR SHORT DELIVERY TIME.

All prices exclusive of VAT

SEND FOR OUR SPRING '79 CATALOGUE - 0·60p