PERSONAL COMPUTER WORLD

50p U.S.$2.00

—PUNCH LINES

—PATTERN RECOGNITION

—JOHANN SEBASTIAN BYTE

—I'M A MICROCOMPUTER WIDOW

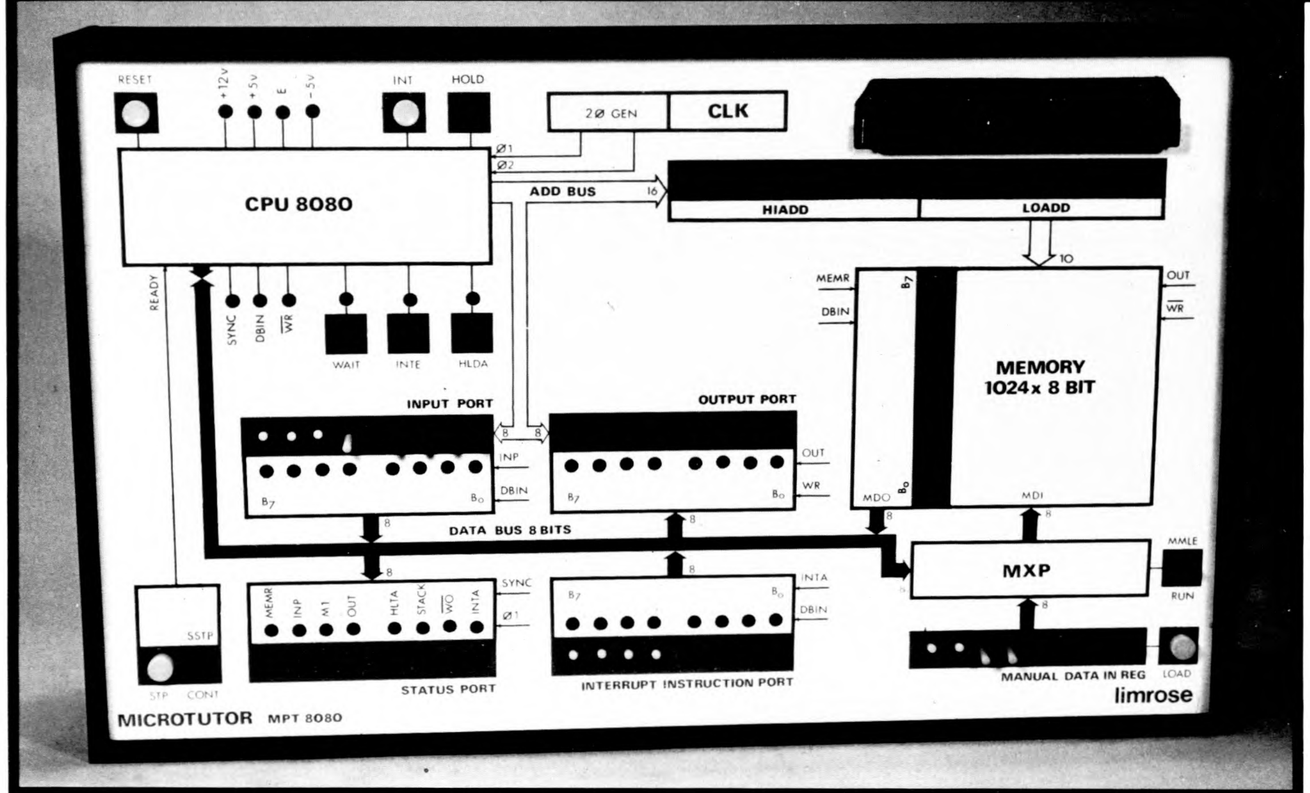
PLUS FASTER PROGRAMMING

JULY 1978

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

THE LIMROSE MICROTUTOR 8080

New, Fast and low cost method for learning the "how" of microprocessors



* 8-bit Microcomputer with 1K RAM
* 8-bit Input Port
* 8-bit Output Port
* 8-bit Status Port
* Manual or Teletype Operation.
* Full vectored- Interrupts
* Buffered Data and Address Buses
* Advanced "Real-time" monitor
* Expandable with plug-in cards
* Comprehensive Instruction Book

limrose electronics limited

241-243 Manchester Road, Northwich, Ches., CW9 7NE. Tel. 0606 41696/7

Now.. there is a new, fast, simple and inexpensive MICROTUTOR to give you truly "hands on" experience to master and apply microprocessors.

Limrose's MICROTUTOR MPT 8080 has been chosen by the British Post Office, and many other large companies in U.K. and overseas, to train their engineers in this new and exacting technology.

The MICROTUTOR MPT 8080 comes ready to use. Nothing else to buy or debug. It's front panel has all you need to understand how the industry-standard 8080 microprocessor works. And... you can expand it later to use as a 'Development System'.

It's inexpensive, it's versatile and comprehensive.

It has a full range of expansion modules, such as expansion motherboard, RAM, ROM, Teletype and VDU interfaces, Cassette Interface, EPROM and PROM Programmers, and even an inexpensive EPROM erasure.

And, of course, the MICROTUTOR MPT 8080 is supplied with a comprehensive Instruction Book. By following the Instruction Book, most persons with limited technical knowledge can rapidly learn how microprocessors work.

AND.... IT IS BRITISH - MADE !!

PERSONAL COMPUTER WORLD

UK 50p
US $2.00

Vol 1, No 3
July 1978

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

CONTENTS

PUBLISHER'S LETTER 2

EDITORIAL PAGE 3

LETTERS 4

TIDBITS
Products...company news 8

FASTER, MORE EFFICIENT PROGRAMS *Sheridan Williams*
Save time, memory, effort 11

A DAY IN THE LIFE OF... *Tim Moore & Andrew Stephenson*
The wacky world of selling personal computers in 1981 14

THE GATES OF REASON *Patrick Sutton*
Continued from Vol. 1 No. 1 16

A MIGHTY MICROMITE *K. R. James*
Ingenuity and the COSMAC 1802 22

IN PRAISE OF THE PDP11 *Mike Lord*
A classic small (though not personal) computer 26

PROGRAMMING: A PERSONAL APPROACH *Stephen Holden*
A small software consultancy's approach to programming 29

DO IT YOURSELF MICROS FOR COMPANIES *Richard R. Waller*
Consultant of a powerful software house outlines its approach to small business computing 32

JOHANN SEBASTIAN BYTE *Michael Inggs*
No, *don't* say it: this Byte isn't better than Bach 34

OWNER'S REPORT *Ken Wheeler*
The Sol also rises 38

I'M A MICROCOMPUTER WIDOW *Linda A. Grand*
A story to touch you to the tip of your funny bone 42

PCW **OPEN PAGE** *Mike Lord*
The ACC; and other items 44

PSEUDO RANDOM NUMBER GENERATOR *Cliff Clark*
First of *PCW* calculator nuggets 47

A GUIDED TOUR OF THE Z80 *Neil Harrison*
Another *PCW* reference article 48

8080 DEBUG ROUTINE *Margaret Berg*
A tidy little routine to treat bug bytes 52

PUTTING BITS IN THEIR PLACE *David Goadby*
In response to a reader's suggestion, an article on building your own EPROM programmer 56

PUNCH LINES *Mark Colton*
If you have a 6800 system and want to marry it to a Westrex punch, this is the first part of the ceremony for it 59

PATTERN RECOGNITION *David Hand*
If robots ever come near to emulating man, pattern recognition will be a fundamental part of their processes 62

MAKING A SPLASH *F. Heathman*
If you win the pools using these hints nobody will be more surprised than us. But as an exercise in thinking it's a winner 66

BUZZWORDS *Barry M. Woollard*
A first A to Zzz . . . of computer terms 68

HOW TO FIND OUT THROUGH LIBRARIES *Mark O'Connor*
Man is a knowledge gatherer. In the beginning was the Word. Now there's computer storage 70

COMPUTING AND DEAFNESS *T. D. Allen*
An article to emphasise that applications are for people 74

PCW **BOOK REVIEW** 76

ERROR MESSAGES 76

DOTPIKS © K. HAMME
CARTOONS © GIGO & SAM
COVER: PAINTING by SAUVEUR LAURENT SANT.
PHOTOGRAPHED by PETER McGEE

Editorial and Advertising Office:
62A Westbourne Grove, London W2
Phone: 01-229 5599 (Publisher)
01-727 8758 (Editorial and Advertising)
Publisher:
A. Zgorelec
Editor:
Meyer N. Solomon
Policy Advisor:
Peter Crofton-Sleigh, FRAS

Consultants:
John Coll, Mike Dennis, Neil Harrison, Charles Sweeten, Patrick Sutton, Michael James, R. W. Davy
Art:
Sauveur Laurent Sant, Kathryn Hamme
Secretarial:
Vanessa Blackburn Kiddle
Layout Consultants:
D. Norris, T. Gabos
PCW Photography: Yoshi Imamura, Peter McGee

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 with the 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 The Yale Press Ltd, Delga House, Carmichael Road, South Norwood, London SE25 5LY. Sole UK Distributors: Seymour Press Ltd, 334 Brixton Road, London SW9, England. Distribution to specialist shops by Intra Press.

Publisher's Letter

Dear Reader,

We receive a few letters every day from readers telling us why they like *PCW.* One of our readers said that the best thing about us is that we do not have too many advertisements.

I hope he is not going to be shocked too much when he learns that it is not a deliberate policy on our part but more the result of difficulties any new magazine not linked to a big publisher experiences at its initial stages. A credibility gap has to be bridged, and this we are well on the way to doing: convincing shops, manufacturers, dealers and agencies that their money is well spent.

Our advertisers are happy with *PCW* as a medium; readers can help to further convince them by mentioning *PCW* when writing or phoning about products or services.

Our own exhibition! Elsewhere in this issue are some more details. Ours is being billed as the exhibition with the difference, and with good reason. We have found an ideal associate to help us organise the exhibition — **Interbuild Exhibitions Ltd,** an exhibition organiser which agrees entirely with our aims. Since 1895, they have staged trade and similar exhibitions all over the world. Interbuild Exhibitions Ltd., is part of the Andry Montgomery Group, the most experienced exhibition organisers in the U.K. The combination of *PCW* reader enthusiasm and their expertise promises an exhibition to remember.

Now to an apology. The last issue was late, but as this issue shows we are back on schedule, having organised ourselves to switch over to being monthly.

(SP) SIRTON PRODUCTS

We specialise in the S.100 Bus System with 8080 or Z.80 CPU's.

Board Kits

* 8080 CPU Board with jump on reset complete with sockets etc.	£106.75
* 8K RAM Board low power 450 n Sec. (21L02-1)	£118.50
Cassette Interface Board 187 to 540 Bytes/Second — built & tested	£121.55
* PROM Board for 8K EPROM 2708 with Integral Programmer	£118.50
* S.100 Motherboard 10/11 slot with Active Terminations and 4 edge connectors	£49.95

Keyboard — now in stock

Keyboard ASR33 style, in kit form, having ASCII 7 Bit output, with case	£38.50
(Available ready built to special order only)	

Coming Soon

* I/O Board Four 8 Bit Parallel and One Serial Port — due end June	T.B.A.
* Video Interface Board with 1K RAM	T.B.A.

Hardware

S.100 edge connectors gold plated solder tail	£5.45
Transformer Pri 110/240 V; sec. 8 V @ 10 Amp & 25 V CT @ 2 Amp	£12.25
Bridge Rectifier with integral bracket rating 18 Amp	£3.95

Integrated Circuits

Due to recent price reductions from the semiconductor manufacturers we can now offer the following prices:

8080A CPU Chip 2 μ Sec		£9.75
21LO2 RAM 1Kx1 Bit Low Power 450 n Sec		£1.50
21LO2 RAM 1Kx1 Bit Low Power 250 n Sec		£1.95
2708 EPROM 1Kx8 Bit 450 n Sec		£12.85
1702 EPROM 256 x 8 Bit	Few Only	£6.50
8212 I/O PORT 8 Bit		£3.10

Ask also for our Lists of 74, 74LS, TTL and CMOS IC's.

* *Items thus marked also available ready built and tested at £20 extra.*

Please add 8% VAT.

NEW PRODUCTS BEING ADDED TO OUR RANGE EACH MONTH — WRITE OR 'PHONE FOR LATEST LISTS, PRICES AND DISCOUNTS.

SIRTON PRODUCTS
13 Warwick Road, Coulsdon, Surrey CR3 2EF.
Tel: 01-660 5617 (day and evening).

Post & Packing: Kits: 60p per kit. Transformer: £1.00. Hardware and IC's: 30p per order.

XITAN SYSTEMS

A **comart** DISTRIBUTER

Z80 SYSTEMS
OUR SPECIALITY
From £1539 ex vat.

CROMEMCO
NORTH STAR
LEAR SIEGLER
INFO 2000
DYNABYTE
AXIOM EX800
MODEL 43 TELETYPE

31 ELPHINSTONE ROAD, HIGHCLIFFE, DORSET, BH23 5LL.
Phone (04252) 77126

Editorial Page

A Letter to the editor
Any tendency to complacency and self congratulation I might have had after the success of *PCW* was quickly righted when I received the following letter from a reader, **Derek J. Chown.** It has been edited. He writes, "I wonder if any of your readers feel the same as I do about the term 'personal computer'? At first I couldn't think why the term should offend me . . . it seems to preclude the use of the machine by anyone but its owner . . . a 'personal computer' permanently displayed for all to see. Ah, that's it, ". . . for all to see". It's the same old human weakness being exploited by the manufacturers just as it is with cars and T.V. sets and kitchen appliances and hi-fi. With computers, which are already established outside our homes, it is necessary to emphasise 'personal' in order to keep ahead of the Joneses. With a mere 'home' computer, it is doubtful whether this could be achieved. 'Personal' helps us overlook the fact that it is a 'little brother', or rather 'microbrother' to scientific and commercial machines. Or am I just too British?"

Of course I do not agree with this viewpoint, but then I do have a vested interest: our magazine. All the same, Derek Chown has said something worth thinking about.

Mr. Huge advises me
I was having a quiet conversation with Mr Small when Mr Huge strode into the office — impeccably dressed, as affable as a nightclub commissionaire. His eyes twinkled with good humour until he sat down, opened his attaché case, and talked business. The gist of this one-way conversation was that our readers were sheep; we weren't doing enough to clip the wool off their backs. Then he turned nautical and said that if we didn't trim our sails others with greater resources would come sailing in and blow us out of the water; if we didn't want to sink, he should take the helm.

I noticed that Mr. Huge, imposing though he was, avoided looking Mr. Small straight in the eye.

Finally, taking my attention for agreement, a look of patronising boredom spread over his face. He was certain he had won his point, so I was no longer of interest. He immediately fell asleep and the room was filled with his self-important snores.

Mr Small now spoke for the first time. He said just one word, "Poppycock!"

Competition Result
The NASCOM 1 has been won by Kenneth F. Horton of 50, Lymefield Drive, Worsley, Manchester. The subject of his article was, "Frequency Measurement with the 6800." A Sinclair Programmable has been won by David Francis of 11 Croft Crescent, Yarpole, nr. Leominster, Herefordshire, with his entry, "Getting into Neighbourhood Consultancy". The other Sinclair Programmable was won by Mr. W. McIvor of 2 George St., Accrington, Lancs. His entry was, "A Byte Orientated Hex Keyboard". The names and addresses of the winners of the September 1977 isue of *Scientific American* will be published next month.

New Contributors
Andrew M. Stephenson is a full time writer specialising in science fiction of a technological nature. His latest novel, *Nightwatch*, is published by Futura as an Orbit book . . . Once David Goadby gets on the telephone to his friends they get chained to the line. He's a practical wizard . . . Mark O'Connor works at Chelsea College library, plays guitar (claims he's improving) and says he's paranoid about computers . . . Cliff Clark is a Member of the Institute of Science Technology, but he says, "don't go in for judging by letters behind a name . . . go more for, if you switch it on does it work?"

Bob Waller is an expert's expert, a consultant at CAP Microsoft. He is very tall, with a sense of humour to match . . . Mark Colton is just seventeen, a pupil at Oundle . . . Stephen Holden has his own small software house, Small Machine Software. He writes engaging letters and has an empathy with the beginner . . . Margaret Berg is a software programming consultant with Wilcox Computers of Wrexham. Her 8080 Debug routine is a subset of one that has been in use for over a year.

Linda Grand tells us a little something about herself in her article . . . F. Heathman is a computer engineer working for a big computer manufacturer. His interests are in minimal hardware circuitry, and robot hardware . . . Ken Wheeler at first wasn't sure if he would stay in England or go back to California; he's stayed, and we get his owner's report . . . Dr David Hand's article on pattern recognition is written by a man of wide interests.

Finally, to Sheridan Williams. Our cover is an imaginative illustration for his article on faster Basic programming. It is entirely apt that he is a keen amateur astronomer and has built his own 8½" Newtonian reflecting telescope. Twenty nine years old, he is a lecturer at Barnet College, London.

Advertisers

It pays to advertise *in* **PCW**.
★ It's the magazine people want to keep.
★ A highly intelligent and articulate readership with purchasing power to match.
★ Unique in Europe, from the quality of its content to its widespread distribution: on sale to the public at all good newsagents.
★ On sale at specialist shops.
★ Available at over thirty prestigious hotels.
★ Estimated sales of No. 1 — 25,000.
★ Over 3,000 subscriptions.
It pays to advertise in PCW.
Contact the Advertising Department at 62A Westbourne Grove, London W2.
OR 'phone 01-727 8758.

Subscriptions

Having difficulty in ensuring your copy of *PCW*?
You can make sure of getting your copy by taking out a subscription. Rates (for twelve issues):
UK — £8
USA — $20
The Continent and elsewhere — £9.80
Subscriptions payable to Intra Press, 62A Westbourne Grove, London, W2.
Back numbers: Available to callers at the above address (newsagents, ground floor) or by post for 65p. Also available to personal callers: *Byte* magazine (£2.00).

Letters

Dear Sir,

I am 15 years old and am very interested in the subject of computer programming. We have a terminal in school which we can log onto either the H.P.2000F or the Systime 5000. I am not doing computer studies but I use it as much as I can (during lunch and after school).

I would like to make programming my career. If you have any advice on how I could go about this I would be very grateful.

Among the subjects I am taking for 'O' levels are: Mathematics, Physics, Chemistry, English.

I have enjoyed reading your first edition of Personal Computer World. Carry on the good work.

Jeffry Kallenbach
78 Carlton Mansions,
Randolph Avenue,
London W9 INS

PCW *We invite the submission of articles which can comprehensively, in not more than 1,000 words, answer Jeffrey's enquiry.* **PCW**

Interaction with a big I.

You asked for interaction, well here it is, but please bear in mind that I am a computer support Engineer so I obviously know more about hard and software than most. (Letter refers to Vol. 1, No. 1).

1. The Searcher/Minmon — good with tapes, either cassette or paper, for the monitor.
2. Editorial — very good.
3. Who's Who — good idea but the info should be with the article.
4. 77 — 68 — good but at this level very limited.
5. How not to crumb up your breadboard — Bad, Bad, Bad, Ordinary Solder must NEVER be used on electronic circuits, to do so is asking for trouble. Standard TTL only requires + 5V so why a double P.S.U. Remaking bad solder joints the way described works if it was due to a cold iron but it will not help dirty joint at all (of course you cleaned and tinned your components before using them — didn't you?) Integrated circuits are not necessarily highly sensitive devices; military spec devices and radiation-hardened types can work in environments that will kill men, what are sensitive are M.O.S. I.C's, since most M.P.U.'s (but not all) are M.O.S. Care should be taken, circuits like PIOs and RAMs also tend to be M.O.S. so beware.
6. NASCOM — Good; I may well buy one.
7 A Computer That Means Business — this seems to me to be pure advertising — what's it doing in Personal Computer World?
8. The Gingerbread Man's Computer — If you must have business machines in, this approach is much better.
9. Flow charts and Pontoon — good but basic.
10. Missionary job — great but where can I get a cheap Selectric?
11 It's a pity the chart in John Coll's article was split up.
12. Modelling — looks good, but I've not read it properly yet.

In conclusion very well put together with few glaring errors, well balanced but aimed a little bit low for me; but worth having if only to keep in touch with amateur computing.

A.J. Dickinson
295 Stretford Road,
Urmnston,
Manchester.
M31 3AG

CASBA

You ask for interaction from readers and I make the following comments: The spread of articles appears good — catering for the novice through, apparently, the real expert. However, in order that novices such as me can begin to understand articles aimed at the expert, could you not give a mini review at the beginning of each article indicating what experience, knowledge etc. is required to appreciate the article.

The idea of CASBA is excellent — please note me as a person interested. I am a Chartered Accountant with a particular interest in business systems and becoming increasingly involved with computer manufacturers/systems houses etc trying to sell small systems. My experience indicates that there is a need for a CASBA type organisation. Particular areas where it could be of assistance is in making computers understandable to the small business community and perhaps encouraging suppliers to bring prices down to those acceptable to the small business community.

N. Horder,
15 Stokes Road,
Winchester, Hants.

Our readers have a proprietary interest in their magazine:

Congratulations (belated) on your very informative *first* issue. Writing as a senior programmer with no electronics knowledge, I found the level about right. There is a natural tendency for technicians to assume technical knowledge and software people to assume programming experience in writing articles. You seem to have successfully stamped on this.

Suggestions for future issues — an article on power supplies (why, what and how) and a bit on the history and development of BASIC.

Now to a niggle. I assume the inset to "Past Procession" (Vol. 1, No. 1) was written without reference to Randell's collection of essays recommended by Kewney. From this book, I noted that the first mechanical calculator was produced by Hr. Schickard of Tübingen, that Hollerith produced a punched card system for the Baltimore Mortality Statistics in 1887, Zuse and Schreyer predated IBM and Harvard by 3 years (and the code-breakers at Bletchley may have been earlier than that) and that Williams and Kilburn in Manchester produced a stored-program computer in 1948. Torres' electro-mechanical calculator of 1920 (with input/output by modified typewriter!) is also worth a mention. Perhaps not all these dates are significant in the development of computers, but then Pascal's calculator is as irrelevant as Schickard's in comparison with Thomas' Arithmometer of 1820, which gained wide acceptance. I do not have the book in front of me, but I believe my notes are accurate.

Some significant developments you missed were LEO in 1951 (first business computer) and Atlas (first with a lot of things).

May I also plead for a future panel on software developments — not so dramatic, but nonetheless important.

As I say, it was only a niggle about what was just an appendix, in effect, and I think the magazine as a whole is marvellous.

Frank Little
47 Christopher Road
Ynysforgan
SWANSEA
SA6 6QR

PCW *Niggle away. It's good for us all.* **PCW**

Phenomenal Computing

I would like to appeal, through your pages, to any reader who might be interested in a special project, but first a few words to put the context. Some of your readers may know of my small journal, FORTEAN TIMES, and the recent book I co-authored with John Michell, PHENOMENA. The subjects of both are the many accounts of varieties of strange phenomena, from UFOs to falls of frogs, from haunted houses to vanishing people, from enigmatic archaeological objects to series of mystery fires, etc (full prospectus on request). Despite the 'cranky' nature of our material, our interest is genuinely scientific, and our biggest problem is in handling the sheer volume of data. The late Charles Fort, from whom we take our name and direction, spent most of his life in libraries extracting material from scientific journals and newspapers from all over the world, and within his set limits of 1800-1931 AD came up with about 40,000 separate datums. From my own researches I know there are many he missed, and our research programme involved scanning all the likely sources before and since Fort's period. Meanwhile FT's small network of readers scour contemporary papers for reports of these types of events for the pages of FT. All of which makes for an incredible amount of data to correlate. The real figure for separate datums may be several millions, and added to daily!

At the present the few serious researchers engaged in Fortean studies are firmly convinced from years of combined experience that one of the most fruitful lines of inquiry is the intergration of many different kinds of information. A very limited pilot analysis of Fort's data, some years back, revealed that in selected categories there were identifiable periodicities. Similar work (primarily for stock exchange speculators) by the American Institute for Cycle Research, has established, eg., a 9.6 year periodicity that links, say, lynx abundance in Canada, European wheat prices and the U-magnetic value of our palnet's magnetic envelope. For our part we wish to feed in correlates and data from as many sources as possible, and retrieve case details in statistical, serial and similarity combinations.

Perhaps, among your doubtlessly fast growing readership, there are a few closet Forteans who would like to combine their interest in the Unusual with their knowledge and experience of home computing. In terms of the applications we have in mind, it's obvious that someone who has read Fort, or know of our modern continuation of his pioneering work, will have less of a communication gap and have more motivation than anyone starting from scratch. Essentially I'd like to hear from anyone who would be interested in advising us. We have very limited resources, but lots of data, inventiveness and enthusiasm. For my part I'm prepared (and interested) in learning enough about software and hardware to play a constructive role in what is, for us at least, an exciting and inevitable step — our own computer and database.

Best wishes for your future success.

Robert J M Rickard
Editor
FORTEAN TIMES
c/o DTWAGE.
9-12 St Annes Court,
London W1.

PROFESSIONAL ADVICE

I have just bought a copy of "Personal Computer World", as we intend to buy a computer quite soon for business use.

As we are a firm of commercial and industrial photographers, I noted immediately the low standard of photographs used in the magazine.

Good photography is not cheap but, in my opinion, the interesting articles are being let down by the photographs accompanying them. Bad photography can do nothing but harm the sales of your new magazine whereas good photography can do nothing but improve them.

J R Goodman
Partner
Sales Photographic
40 Preston Rd,
Brighton BN1 4QF.

HELP!

I am at present compiling (with friends) discographies (bibliographies of gramophone records) of various types of music, and the obvious problem is one of the sheer volume of paperwork, index cards etc., and the chaos caused when a draft is subsequently to be updated by more research. The flash answer to our problems would be a ready made word processing from one of the big boys . . . the problem is one of finance. I would be grateful for any advice which would point me in the right direction. Is it possible to assemble a storage system and the necessary hardware/software to be able to compile and edit directories of reference information from the components available to the amateur?

Mike Black
26 Alma Road
Carshalton, Surrey.

I have been in computing for more years than I care to remember and have seen a lot of changes of attitude, from single user machines to enormous centralised batch and multi-user machines, and now the expanding trend back to single-user minis and micros. Back to hands-on use again, but the most important change has been the continual reduction in cost, to a point where small businesses and individuals can begin to afford them.

But despite the low cost of hardware there is still a big gap in the area of applications software and present hardware for micros still leaves a lot to be desired for applications oriented computing. At present micro-processors are most suited to control functions which don't involve much numerical computation. It is certain that this will change and new generations of microprocessors will provide arithmetic functions beyond addition and subtraction.

In the meantime we can prepare for this by trying to find useful applications for micro-computers beyond games and basic software tools. Readers like the one who wants to know how to use small computers for calculating the daily feed rations for a herd of 120 cows, should be *encouraged to describe their present manual processes,* so that those of us who are good at programming can have a go at estimating what is necessary and then writing some software.

As an owner of a Texas SR52 programmable calculator which offers a great deal more computing functions than the simple micro, I have written more than 80 application programs which are stored on magnetic cards. I have been surprised at what is possible with such a physically small device, and even programs that I had previously written for full-scale computers of 8K or more bytes have proved to be simpler to write and squeezable onto a few cards.

With best wishes for the success of PCW.

J.H. Sexton
46 Riverdale,
Wrecclesham,
Farnham,
Surrey,
GU10 4PJ.

I read your first issue with much interest, most of the articles being very well informed. The article "Basic Pontoon", however, contained an algorithm that could be of interest, only to someone who wishes to burn as much time as possible on his processor. I refer to the shuffle. I, in fact, instrumented this algorithm and came up with the following average figures.

Over 1,000 shuffles *each shuffle* generated approximately 120 random numbers, and iterated the inner loop (where cards were counted) no less than 3,700 times!

To anyone considering mounting this game I would like to offer the following code:

```
31 FOR I = 1 to 52
33      LET P(I) = INT(I/4)
35 NEXT I
36 REM THE ABOVE SETS UP THE PACK AS
37 REM 1, 1, 1, 1, 2, 2, 2, 2,. . . .ETC
38 REM AND IS ONLY EXECUTED ONCE!

60 FOR I = 1 to 52
70      LET R = INT(RND*52)+1
```

```
 80    LET S1 = P(I)
 90    LET P(I) = P(R)
100    LET P(R) = S1
110 NEXT I
120 REM THE ABOVE RE-ARRANGES THE PACK
130 REM INTO RANDOM ORDER
140 REM AND IS THE ACTUAL SHUFFLE
150 REM
```

These statements can be plugged into the program as published.

G.E. Greenwood
12 Ryeland,
Stony Stratford,
Milton Keynes,
Bucks.

Reader's Assessment

Welcome to a new computing magazine. As you say in your editorial you would like to interact with your readers, I thought that I should respond.

The overall standard of the first issue is very high, my only grumble being that there should be more pages and it should be a monthly!

As regards the articles themselves, I would rate them as follows (points out of 10):—
The Searcher 8, Personal Power 7, Past Procession 4, Mighty Micromite 8, How not to . . . 5, Yours to Command 9, The Little Symposium . . . 7, The Elegant Minmon 10, A Computer that . . . 3, Gates of Reason 8, Gingerbread Man's . . . 6, Flowchart . . . 7, Basic Pontoon 10, Missionary Job 5, Open Page 6, Do We Want . . . 2, Direct Addressing . . . 8, Cutting the World . . . 1.

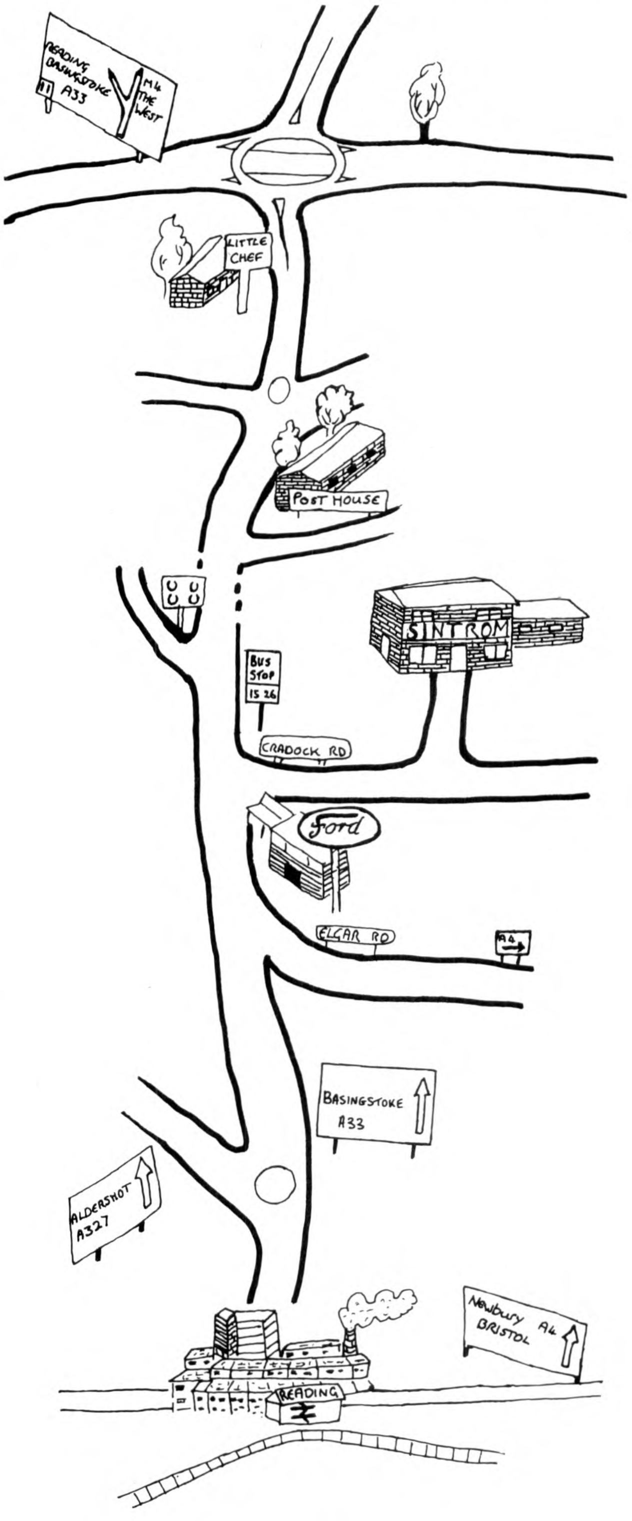
Some of the above may surprise you, but I've tried to give a fair rating from my point of view. The highest ratings have gone to those articles which convey the most information or which will be useful in one's library. The lowest rated ones don't really deserve the space devoted to them! The one about converting an IBM 73 had so much promise but failed completely — anyone with the knowledge to understand it probably woudn't need the article.

I myself am a relative beginner having gained an interest in home computing through taking courses with the Open University. I'm sure that there are many others in my position who wish to start home computing and are eager to read a magazine such as PCW in order to gain information. Some of the American magazines fall into the trap of trying to cater for both the hobbyist and the professional user, whether he be a small businessman or a teacher, and I feel that those magazines are not as good as those which are primarily for the hobbyist. Although of course such a publication will contain articles of great interest to the professional.

I would like to see PCW developing in interaction with the hobbyist, possibly by printing programs like BASIC in an educational format and allowing for as much feedback as possible from readers by way of comments, improvements, criticisms etc. Also constructional articles on retail kits or home-brew designs which could extend the capability of reader's systems. How to take cheap surplus equipment and connect it to one's system — written for the beginner so that he learns as he goes along. Possibly a series of articles on fault finding — I know from experience that it is very frustrating to build a kit and for it not to work!

Brian V. Teece,
4 Eastville Avenue,
Rhyl, Clwyd LL183TH

SINTROM MICROSHOP
DIVISION



Sintrom welcomes **Microshoppers** with

SWTPC–Micropolis–Perex
Lear Siegler–DRI

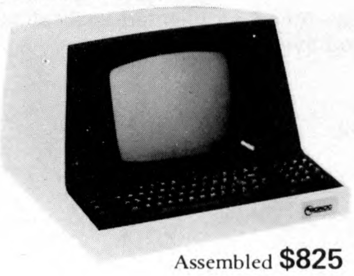
Microcomputer Hobby Shop
14 Arkwright Road,
Reading, Berks. RG2 0LS

Tel: Reading (0734) 84322
Telex: 847395

TALKING TERMINALS?

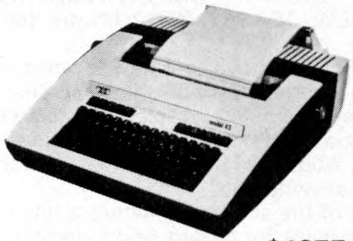
Talk to **SAS Computer Products.**
We'll help solve your terminal needs.

SOROC IQ 120



Assembled **$825**

TELEPRINTER 43



Model 4320 KP (RS-232) **$1377**

LEAR SIEGLER ADM-3A

Kit **$695** Assembled **$795**

SELECTRA-TERM

The ideal hard copy device with expanded features and greater reliability for your microcomputer system.

Assembled **$1750**

For detailed information contact:

SAS COMPUTER PRODUCTS, INC.

7742 Redlands St., No. H3021
Playa Del Rey, CA 90291 USA
Phone(213)821-0642 Telex18-1059

European Sales and Service
SAS Computer Products, Inc.
Badenerstrasse 281
CH-8003 Zurich, Switzerland
Phone 01/241 49 05 Telex 58738

OEM and dealer discounts. All prices FOB Los Angeles, CA, USA

THE

PCW

COMPETITIONS

THE PCW MICROCHESS CHAMPIONSHIP

An Open Event for small machines with up to 32K of memory.
Full details from the address below.
Entries accepted from all over the world.

Best Software: £200 Prize

Best Homebrew System: £200 Prize

Best School Application: £200 Prize

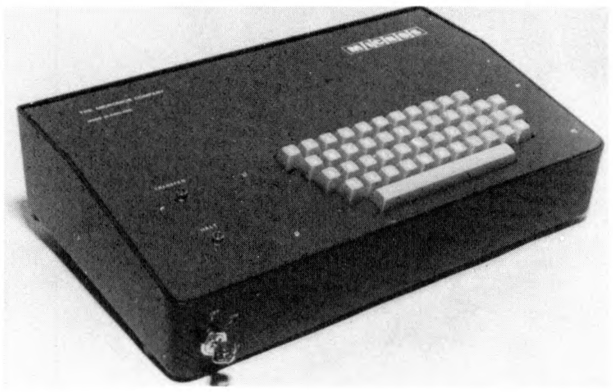
Best Home Application: £200 Prize

plus The PCW Trophy in each category.

plus Donated Prizes.

FURTHER DETAILS FROM
COMPETITIONS
PCW
62A WESTBOURNE GROVE
LONDON W2

Tid Bits PRODUCTS . . . . COMPANY NEWS . . .



MICROS PRICE REDUCTION A significant price reduction is announced for the MICROS microcomputer system, effective immediately it is now listed at £399 assembled and £360 in kit form. Based on the Z80 cpu it includes a 1K monitor EPROM, a 47 key solid state keyboard, video, TV, cassette and teletypewriter interfaces, serial I/O, 2 parallel I/O ports and 2K of RAM. 16K bytes of mixed ROM and RAM can be fitted in the instrument housing and up to 64K externally. Deliveries of an impact printer which can be plugged into the RS232 serial port will begin shortly and the anticipated price will be £150. Comprehensive specifications are now available on request.

THE MICRONICS COMPANY
1, Station Road,
Twickenham,
Middlesex.

NEW ½ Megabyte Floppy Disc Subsystem for S100 Z80 Systems
Xitan Systems announces the launch in the UK of a new 'not so dumb' version of the popular INFO 2000 floppy disc subsystem.

- ½ Megabyte online storage for only £1,800 assembled.
- Includes CP/M* disc operating system.
- Large range of optional development software available.
- Controller includes 2 serial I/O ports.
- Uses IBM format soft sectored diskettes.

The new version incorporates on one S100 board a floppy disc controller for up to 4 single drives or 2 Persci 277 dual drives, two RS232 serial I/O channels, 1½ parallel ports, 1K EPROM boot loader for the CP/M operating system, a second 1K EPROM for the Console and line printer I/O drivers. The board has room on it for a further 6K of 2708 EPROMs for customer use, addressed X'E800' to X'FFFF'.

Two drives are supplied in a single Persci 277 unit. It comes complete with a slimline case, power cord, and power supply, switchable to any of four standard supply voltages.

The CP/M operating system is in 8080 code and includes a text editor, dynamic 8080 debugger, 8080 assembler and various utilities. Optional software includes:—

- 8080 Disc Basic Interpreter;
- 8080 Commercial Disc Basic Compiler;
- 8080 Fortran;
- TDL Z80 software package of a Z80 Macro Assembler;
- Text editor;
- Text output processor;
- Basic Interpreter;
- TDL Z80 ANSI Fortran.

*CP/M is the trademark of Digital Research Inc.

Also available are various American oriented applications packages from Structured Systems Inc.
Further details from:

XITAN SYSTEMS,
31 Elphinstone Road,
Highcliffe,
Dorset,
BH23 5LL.

PERSONAL COMPUTING AND SMALL BUSINESS COMPUTER SHOW • PHILADELPHIA, PA, U.S.A. • AUGUST 24th - 27th, 1978 (RT.I. Box 242 WARF RD. MAYS LANDING, NEW JERSEY 08330 Phone (609) 653 1188)
Personal Computing '78™ till move to a new location for this year's presentation when the Philadelphia Civic Center plays host to the most complete display of personal computing exhibits east of the Rockies. The 1978 edition is slated to run four days, August 24 thru 27th, making it the largest show of its type anywhere.

The opening session of the show will feature a full-day industry trade show set aside for dealers and members of the industry as well as guests of exhibitors and representatives of TV and other segments of the media. Special meetings and seminars for dealers and retailers are planned for the opening night at convention headquarters located in the Philadelphia Sheraton.

John H. Dilks, III, show director, has announced he expects over 200 exhibitors will sign up for the 300 available booths well in advance of show time.

In addition to the exhibits, other attractions will include an art show, music festival, computerized mouse maze, and the highly successful Personal Computing College™. The college will again include over 80 hours of free in-depth seminars conducted by some of the country's leading names in the computing field. There will be topics of interest for everyone, whether the individual is beginner, student, hobbyist, educator or computer professional.

Applied Computing and Software Ltd., the London-based systems and software house, has announced a doubling of its turnover for the company's 1977 financial year.

Turnover for the year was £1.2 million, with profits of approximately £38,000. Some 40 per cent of revenue was from overseas business, chiefly from ACS America, the UK company's US subsidiary.

During the year ACS established specialist divisions and groups in a number of growth areas such as banking, broking and finance and the production field. Highlights of 1977 included the development of systems for shipping and freight forwarding agents, garment and fashion companies, steel and iron foundries, container and plastic manufacturers and a number of commercial and government organisations.

Systems were developed for more than 15 different manufacturers' computers, indluding IBM, ICL, Burroughs, Honeywell, Univac mainframes and small business systems. Also: DEC, Prime, Olivetti, Hewlett-Packard, BCL, Systime, NCR, Singer and ABS.

"Overall, the future looks bright," says ACS managing director, Peter Goldsmith. "As hardware continues to fall in price, the importance of software and systems competence at the implementation end is becoming increasingly recognised. Success will be limited by the ability to recruit, keep and develop capable staff."

For further information contact:

Peter Goldsmith
Applied Computing and Software
Tel: 01-637 0105

Personal Computers Ltd. are pleased to announce that California based Apple Computer Inc. has obtained funding through a recent equity offering. Mike Sterland, Managing Director of Personal Computers Ltd., said, "We are delighted, this will ensure that Apple becomes a major force in the Personal Computer market". The investors include Venrock Associates, Capital Management Inc., and Arthur Rock, three highly respected names in the U.S. venture capital community.

Initial financing for the company was provided by the Bank of America and by individuals within the company.

A.C. Markkula, Apple's board chairman, said "The proceeds will be added to our working capital and used to increase the company's production, new product development and world-wide marketing programs".

"Apple currently enjoys a 12-month market and technology leadership position, and we intend to maintain it."

Apple Computer, Inc. ha been in the personal computing business since January, 1976, and has been shipping the Apple II Computer since May of 1977. U.K. Distributor is Personal Computers Ltd., 18-19 Fish Street Hill, London, 01-623 1434. Contact: Mike Sterland.

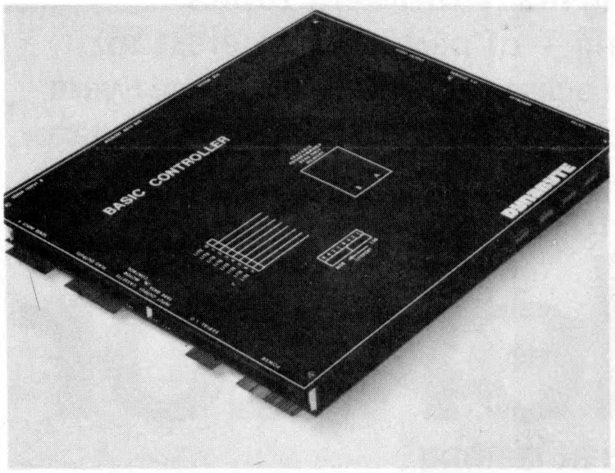
Intel have an extensive technical library which contains many works of reference on the subjects of memory and microprocessor design. Intel has decided to make the library generally available although, since the cost of producing this literature is very high, the company has to charge for the more substantial documents. However, many of the articles, reports and reference cards are supplied without charge, and up to five of these no cost items may be requested with each order placed. Order forms are available from *Intel, Broadfield House, 4 Between Towns Road, Oxford OX4 3NB.*

MICROCOMPUTER-BASED CONTROLLER WITH BASIC INTRODUCED BY DYNABYTE

Palo Alto, CA — A single-board programmable microcomputer system designed specifically for control applications has been developed by Dynabyte, Inc., a Palo Alto manufacturer of S-100 bus microcomputer components and systems.

The Building Controller™ can manage electrical systems a building, automate the control of laboratory test equipment, operate a model railroad train, and apply itself to hundreds of other applications.

The unique feature of the Basic Controller, according to Dynabyte, is that it allows the user to operate the computer and the external devices it controls with a BASIC language called ZIBL™ which was written by Dynabyte specifically for control applications.



FEATURES

- 2.5 MHz Z80 microprocessor
- 4K of RAM, expandable on board to 16K
- 4K of EPROM with programmer
- Two RS232 serial I/O ports, user configurable via software, one port also has 20ma current loop
- One parallel input port and one parallel output port
- 300 baud audio cassette interface with extensive file handling capabilities and motor control
- Keyboard input port
- 32 individually memory mapped flag outputs
- 32 individually memory mapped sense inputs (with resistive pullups)
- 8 relays: 4 reed relays that handle .75 amps and 4 relays that handle 5 amps
- 8 individually memory mapped LED's, intended as indicator lites
- One 8 bit liteport for display of data
- Video interface generates 16 lines of 64 characters with standard composite video output

Dynabyte, Inc.
4020 Fabian
Palo Alto, California 94303
Rick Mehrlich (415) 494-7817

A FIRST SELECTION OF PET COMPUTER SOFTWARE AVAILABLE FROM COMMODORE

Commodore have published their first Official Programme Library, available as of 1st May. Among entertainment programmes available are LUNAR LANDER, PONTOON and BIORHYTHMS.

Management, stock control and **inventory** programmes to first time business sytems users, who have neither time nor inclination to write their own programmes, are available from software houses, details of which will be included in future Users Club Newsletters. In addition to those programmes now generated through the PET USERS CLUB, more commercial, scientific and statistical programmes, including a programming instruction cassette, will be introduced in June.

HARDWARE: A new addition to PET will be a second external cassette deck unit available from May onwards at £55 plus VAT. In June, Commodore will also be introducing an 80 column impact printer, followed by a floppy disk, memory expansion and modem. The printer costs £425 +VAT.

Contact: COMMODORE SYSTEMS DIVISION 360 Euston Road, London NW1 (01-388 5702)

COMPUTER ASSISTS IN THE DIAGNOSIS OF MULTIPLE SCLEROSIS

(**PCW** This press release was so interesting, we decided to print a large part of it **PCW**).

At the National Hospital for Nervous Diseases in London, a Digital Equipment GT46 minicomputer is employed to aid in the early diagnosis of patients suspected of multiple sclerosis. GT46 is Digital Equipment's designation for a laboratory system which comprises a 16-bit PDP-11/40 (or PDP-11/34) computer together with VT11 display unit and other laboratory interfaces.

The technique used by Dr. David Small, Consultant in the Department of Clinical Neurophysiology, is based on the discovery that the volley of nerve impulses (action potentials) triggered through the eye in response to environmental signals produced by pattern reversal stimulation, are often abnormal in patients with multiple sclerosis.

Screening takes place in a small room where the patient is seated in a comfortable armchair facing a TV monitor. Electrodes are attached to the back of the patient's scalp. The averaged visual evoked potentials (VEP) to pattern reversal stimuli are then recorded in the dark from each eye separately. The stimulus consists of a black and white checkerboard pattern, which is projected via a mirror mounted on a pen motor, onto a translucent screen in front of the patient.

A pulse, synchronous with the onset of the mirror movement, is used to trigger the PDP-11/40 computer which acts as a multichannel signal averaging system. The evoked responses are sampled for a period of 256 ms, and 128 such responses are averaged together. The process is repeated to ensure the reproduceability of the response, and also to produce a final averaged visual evoked response comprising the average of 256 individual responses.

A typical record for a normal healthy subject consists of a major negative wave occurring around 70 ms and a major positive wave at 90 — 100 ms. The latency of response is measured from the onset of the sweep to the peak of the major positive wave. In the normal subject the response latency falls into a fairly narrow range whereas in multiple sclerosis it may be increased, frequently by as much as 20 ms in an affected eye.

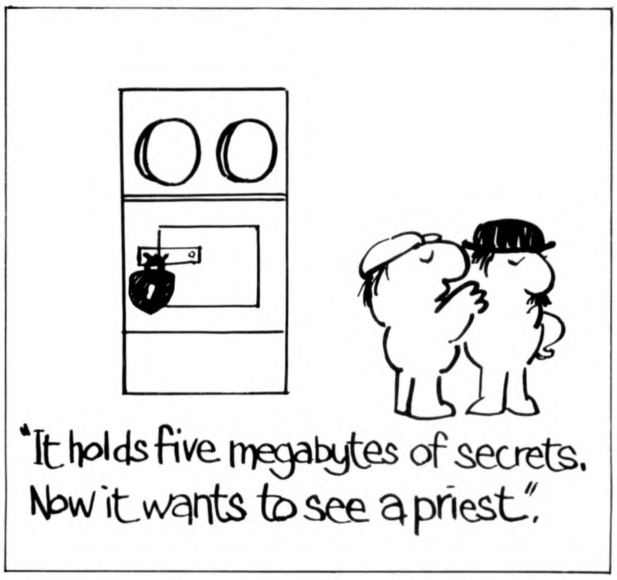
The method has proved invaluable because a common source of difficulty to the neurologist in the diagnosis of multiple sclerosis is the absence of objective evidence of neurological damage at more than one site in the nervous system. The discovery of a delayed visual evoked response can establish the presence of multiple lesions.

Contact:
Valerie Baillie,
Digital Equipment Co. Limited,
Digital House, Kings Road,
Reading, Berks.
Tel! — (0734) 583555

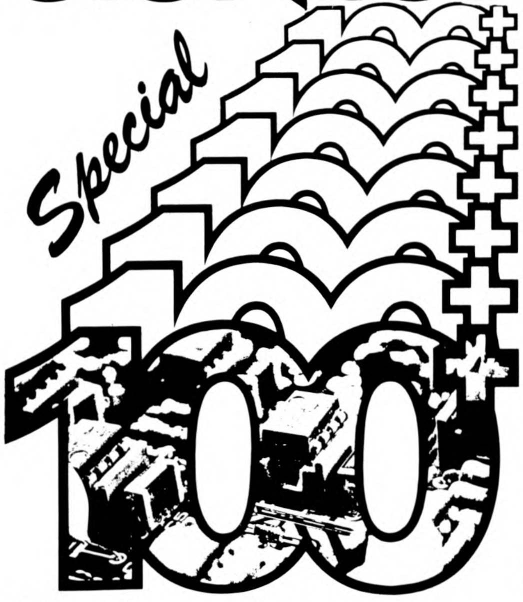
ERA News:
Three countries, Algeria, India and Mexico, sought **ERA's** assistance in the establishment of their own local technical organisations. In Algeria, ERA was retained to undertake a planning and feasibility study of a laboratory complex to provide all the basic services necessary for the development of the Algerian electrical and electronic manufacturing industry.

Microprocessors dominated Computers & Automation activities during 1977. In addition to undertaking development work for clients in the UK, Spain and Portugal, a new multi-client project "The Engineering of Microprocessor Systems" was launched to help engineers overcome the technical, financial and managerial problems involved in developing a microprocessor system.

Please contact for further information:
Michael Webb
ERA Limited
Cleeve Road,
Leatherhead
Surrey KT22 7SA
Tel: Leatherhead (03723) 74151


"It holds five megabytes of secrets. Now it wants to see a priest."

elektor electronics magazine



different circuit designs in our 'Summer Circuits '78' double issue. By tradition this issue will contain over 100 different circuit designs to stimulate the reader. Covering many different topics including — of particular interest to microprocessor enthusiasts — Software Siren, Debouncer, Cmos FSK Modulator, Hexagonal readout and Universal Data-Bus Buffer.

Also featuring

3 complete full length construction projects. The elektor 'preconsonant', 'consonant' and 'luminant' which together provide a top-notch, fully reliable audio preamplifier with an LED level display. Using standard components these projects include circuit design, components lists and layout with pre-drilled, ready etched printed circuit boards and front panel available from the publisher.

elektor

On sale at leading newsagents from June 16th

in cases of difficulty copies of this issue are available from the publisher, Elektor Publishers Ltd., Elektor House, 10 Longport, Canterbury, Kent CT1 1PE
Tel: 0227 54439/30

Price £1.00

Faster, More Efficient Programs

Sheridan Williams

What can you do to make your programs faster and more efficient? This article will hopefully provide the personal computer programmer with the ways of speeding up his programs. Ways already exist for those with money — companies and firms where ten thousand pounds here and there is easily found. Those with money have several options open to them: they can buy a faster computer, or buy a faster compiler[1], or pay a "software house" (or computer bureau)[2] to write the program for them. I have, however, yet to come across a program written by an amateur or professional that cannot be made faster.

It is not always a good idea to concentrate solely on making the program faster, because in doing so it may become longer and/or less accurate. (An article on accuracy will appear in the near future). Nevertheless there are ways that are well worthwhile implementing which do not make the program longer or less accurate.

What is the point in a program that takes one hour to reply to a noughts-and-crosses move? Note also that the slower the computer hardware the more worthwhile are the improvements in speed. Today's large mainframe computers are so fast that a 20% saving in a response time of one second is unnoticeable; but a similar saving to a micro user could save 10 seconds or more.

If you are like me then I think that you will get great personal pleasure improving your programs — even the ones that are supplied or bought (provided that you have the listing) are usually capable of improvement.

The amateur is lucky because computing is his hobby and he can devote time to his hobby which is free. I have written hundreds of programs and have yet to be entirely satisfied; they could all be improved in one way or another. I have a program that takes 15 seconds to reply to a 3-D noughts and crosses game; it used to take over 30 seconds, but by observing the points outlined later I managed to halve its time of response.

OK then, let's get started on some ways to speed up programs. These processes may only make the program 10-20% faster, but they could make a staggering improvement. I once wrote a program that would have taken 8600 years to perform some routine. It only required a small modification and the same routine worked in 0.4 seconds, a slight improvement. Here are some ways in which a student of computer science is taught to manage programs more efficiently. (What about doing a computer science evening class?)

ONE LINE IMPROVEMENTS

The statement LET X = A↑2 could be made 30% faster written in this way: LET X = A*A. Powers are much slower than multiplications.
Similarly replacing LET X = A↑3 with LET X = A*A*A will give a 15% improvement; non-integer and higher powers are not worth altering.
The statement LET X = 2*A could be made 6% faster written in this way: LET X = A+A. Multiplications are slower than additions.

FUNCTION EVALUATION

Remember that certain operations are faster than others, the usual order for decrease in speed is given below. If a statement can be replaced by one *further up* the list then usually it results in a faster execution.

```
+ -
* /
↑
INT SGN ABS RND
SQR
EXP LOG
SIN COS TAN ATN
```

Each line takes successively longer than the previous one. The order of these functions may vary with different compilers and should be checked with the manuals, or by running the benchmark tests mentioned later in the article.

ARRAY VARIABLES

Do not use array variables unless absolutely necessary, always replace LET A(1) = 3*A*B with LET A1 = 3*A*B unless of course you need the modified address that A(I) can provide. It takes most compilers 65% longer to find an array variable in store than an ordinary variable.

Another common occurrence that happens often is the use of the same array element on successive lines. The left hand example is 10% slower than the right hand example.

```
10 LET X = A(P)+3          5 LET T = A(P)
20 LET Y = A(P)+4         10 LET X = T+3
30 LET Z = 6*A(P)         20 LET Y = T+4
                          30 LET Z = 6*T
```

Successive calls of the same array variable are inefficient. Even though the right-hand program is longer it is faster.

FOR . . . NEXT LOOPS

FOR . . . NEXT loops are the fastest way of performing a loop, so should always be used in preference to the LET . . IF . . THEN loop. The former are staggeringly faster, often anywhere between 200% to 2000% faster. Example below

```
10 LET A = 0                      20 FOR A = 1 To 1000
20 LET A = A+1                    30 statements within loop
30 statements within loop         40 NEXT A
40 If A < 1000 THEN 20
```

Typical times are 12 sec and 4 sec respectively for execution. Note also that the right-hand program is also one line shorter.

Nested FOR loops are a source of slowness too, compare these two programs:—

```
10 FOR A = 1 TO 20        10 FOR B = 1 TO 4
20 FOR B = 1 TO 4         20 FOR A = 1 TO 20
30 LET X(A,B) = 0         30 LET X(A,B) = 0
40 NEXT B                 40 NEXT A
50 NEXT A                 50 NEXT B
```

The left-hand program will execute 25% slower than the right-hand program. Why? In the left method the inner loop is set up 20 times. Setting up means that the computer has to store the start, stop and increment parameters. In the right method the loop has only to be set up 4 times.

If at all possible stick to INTEGERS as parameters in a FOR loop. (The opposite of integers are REALS, integers are whole numbers that can be stored exactly; Reals are numbers that may not store exactly; try converting 2.1 into binary). Most compilers treat integers in a different way to real variables, and integer arithmetic is around 10% faster than real arithmetic.

```
1Ø FOR A = 1 TO 1ØØØ     1Ø FOR A = 1.1 TO
                         1ØØØ.1
2Ø NEXT A                2Ø NEXT A
```

There is a more important consideration than one of speed in the right-hand program, that is of accuracy, the right-hand program will only loop 999 times instead of 1000. (the reason will be printed in a future article).

Another common fault is evaluating a constant *inside* the loop, when it could have been evaluated outside it. Example:

```
1Ø FOR A = Ø TO 9Ø       LET X = 3.14159265/18Ø
STEP 1Ø
2Ø LET X = 3.14159265/   2Ø FOR A = Ø TO 9Ø
18Ø                      STEP 1Ø
3Ø LET B = B+SIN(A*X)    3Ø LET B = B+SIN(A*X)
4Ø NEXT A                4Ø NEXT A
```

The left-hand program is poor programming as well as taking 15% longer.

Here is an interesting program, the output achieved has been varied depending on the compiler used.

```
1Ø LET A = 1
2Ø FOR A = 1 TO 3*A STEP A
3Ø PRINT A;
4Ø NEXT A
```

What happens if we use as parameter for the loop the loop variable itself? Here are two outputs achieved with different compilers: 1 2 3 and 1 2 4 8 16 32 64 128 256 for ever. If when you run this program you do *not* get the output 1 2 3 then the following is another way of speeding up your programs:

```
1Ø LET B = 6             1Ø LET B = 6
2Ø FOR A = B TO 3*B      2Ø LET C = 3*B
STEP B/3
3Ø statement             3Ø LET D = B/3
4Ø NEXT A                4Ø FOR A = B TO C
                         STEP D
                         5Ø statement
                         6Ø NEXT A
```

The right-hand program will execute faster, because the loop parameters will only have to be calculated once, at the beginning of the loop only.

Apart from the last point made, all the previous points are universal and all compilers that I have used benefit from the above methods of speeding up.

BRACKETS

Brackets make no difference to the speed at which a function is evaluated. LET X = (((A + B))/C) is evaluated at the same speed as LET X = (A + B)/C. So the moral is always use too many brackets rather than too few. An example of too few brackets is given here in finding a solution to the quadratic $ax^2 + bx + c = 0$.

LET X1 = (−B + SQR(B*B − 4*A*C))/2*A would give a *wrong* answer

LET X1 = (−B + SQR(B*B − 4*A*C))/(2*A) is correct.

TRIGONOMETRIC FUNCTIONS

If your compiler will evaluate SQR faster than SIN, COS, TAN, then if you are evaluating more than one of the functions at the same time then it is worth using the following trig identities to evaluate the other: $\sin^2 x = 1 - \cos^2 x$

$\tan x = \sin x/\cos x$ $\quad \cos^2 x = 1 - \sin^2 x$

In a program if we evaluate sin x we can evaluate cos x and tan x from it:

```
1Ø  LET A = SIN(X)           1Ø  LET A = SIN(X)
2Ø  LET B = SQR (1 − A*A)    2Ø  LET B = COS(X)
3Ø  LET C = A/B              3Ø  LET C = TAN(X)
```

The left-hand program should be faster than the right-hand program.

LOOKING FOR THE PLACE TO SPEED THINGS UP

The place to look is inside a loop. If the loop is traversed enough times a saving of 10% each time around can make an appreciable difference. If a statement is not inside a loop it is unlikely to be worthwhile altering it.

TESTING YOUR COMPUTER/COMPILER'S SPEED

It is possible, indeed it is a very good idea, and only needs to be done once, for the amateur to check and list the speed of execution of various routines and functions. The tests used for measuring are called 'Benchmark tests' and you will find a series of 8 such tests in *PCW* Vol 1. No 1. Pg 57.

Benchmark tests are devised to find the speed and efficiency of a particular computer/compiler; using them it is possible to time a particular routine, addition say. Here is a method for timing addition, but it can be extended to time multiplication, SQR, SIN, RND etc.

```
1Ø   PRINT 'S'            Time this program
                          using a stopwatch
2Ø   FOR A = 1 TO 1ØØØ    from the printing
                          of the 'S' to the
3Ø   LET X = 2            printing of the 'E'.
                          Now replace the
4Ø   NEXT A               line 3Ø LET X = 2 + A
                          and retime the
5Ø   PRINT 'E'            program. The differ-
                          ence in time is
6Ø   END                  due to the 1000
                          additions performed
```

in line 3Ø, so divide the time difference by 1000 and you have the time for one addition.

Replacing statement 3Ø with 3Ø LET X = 2*A or LET X = SQR(X) and you can time various functions. Make a note of the various speeds and you can modify your programs to work faster.

One final word of warning, your efforts may only lead to a 10% saving in run-time but they could accumulate to something more worthwhile.

GLOSSARY

1. COMPILER A compiler is a program written in machine language, which translates a source program, written in a high-level language (BASIC, COBOL, ALGOL, FORTRAN) into a machine language program. Remember that computers cannot understand any other language except their own machine language, and all other languages must be translated before they can be executed.
2 COMPUTER BUREAUS are independent companies specially (SOFTWARE HOUSE) formed to provide computing services to clients. They offer many services, including program writing, and hiring of computer time. They also provide software packages.

To finish with here is a program to write:
Write (i) the fastest (ii) the fewest namber of statements a program to fill the array A(1) to A(100) with the first 100 prime numbers. (The lowest prime number is 2).

Send your entries to Sheridan Williams 22, Oakleigh Crescent, Whetstone, London N20 enclosing a stamped addressed envelope. The two winning solutions will receive £5 each from *PCW.*

'78

The 1978 Do-it-yourself Computer Show

The most significant event about microcomputers in business and at home

International exhibition, conference plus seminars and workshops plus debates.

22-24 June
West Centre Hotel, London

online

Please send me a fully detailed brochure about
The 1978 Do-it-yourself Computer Show

Dr/Mr/Mrs	*Initials*
	Name
	Job title
	Address

Return to:
Online Conferences
Cleveland Road, Uxbridge
England, UB8 2DD

Tel: (0895) 39262

A Day in the Life of a Retail Computer Store CIRCA 1981

Tim Moore
and Andrew Stephenson

08.30
Bill slung his muddy denim jacket on top of a visual display unit and subsided onto the stool before it. He peered at his own image reflected in the machine's blank screen, sticking out his discoloured tongue for inspection.

Jenny came out of her office, looked at him, then she glanced at Jim, who was stacking software manuals on a shelf. "Stock checking today," she said genially.

Bill groaned and went to the coffee machine. A token earned him a plastic cup wreathed in steam. Jim said nothing until the last of the books was safely balanced. Only then did he answer Jenny.

"No point re-ordering Nippexas parts", he said. They always ignore our requests. Their despatch computer invariably thinks up some ruse to foil us. Depend on it."

Bill sipped his coffee and gasped. In a choked voice he said "Blast Nippexas. If they will take 8 weeks to deliver ________"

Jenny interrupted him smoothly: "How did it go last night?"

With a scowl, Bill resorted to his coffee again, whilst Jim sniggered.

"Don't tell me," said Jim, "it beat you at tiddlywinks again?"

"No" said Bill defensively.

"Noughts and crosses?"

Even Jenny, who preferred not to take sides, was smiling.

"Chess?" enquired Jim. Still Bill did not reply. "Oh, come on!" exclaimed Jim. "Just how much have you managed to cram into that one-chip wonder of yours?"

"If you must know," blurted out Bill, sounding lightly aggrieved, "it was Intergalatic Space War . . . but it wasn't my programme. How was I to know those confounded starship routines allow for double hyperjumps?"

"Never mind," Jenny consoled him, "maybe you'll make it to the championship *next* year."

09.45
"Well?" asked Jenny. "What are we short of?"

"More accurately," snorted Jim, "what have we got?"

Bill surveyed his screen hardware listing moodily. "Halbri still owe us 200 of those "Best of Basic Software" comics of theirs. And, as ever, the 4.7K pullup resistors have got themselves lost in the post." He stirred the sludge in his coffee cup thoughtfully, using a legless 65 kilobyte RAM as a spoon. "You know," he said, "one of these days we might even hear from Fiyitel about their F-7800 handbook; they were promised for almost ten months ago."

Nodding, Jenny retired to her office again and began to make phone calls.

Bill set aside the i.c. and dropped the coffee cup into a waste bin on top of the pile that had accumulated there so far that morning. It made a wet, ugly sound. "I've been thinking," he remarked.

Jim raised his head politely from his own stocklist.

Bill continued: "The phone."

"Profound," observed Jim. "Is that the lot, Socrates?"

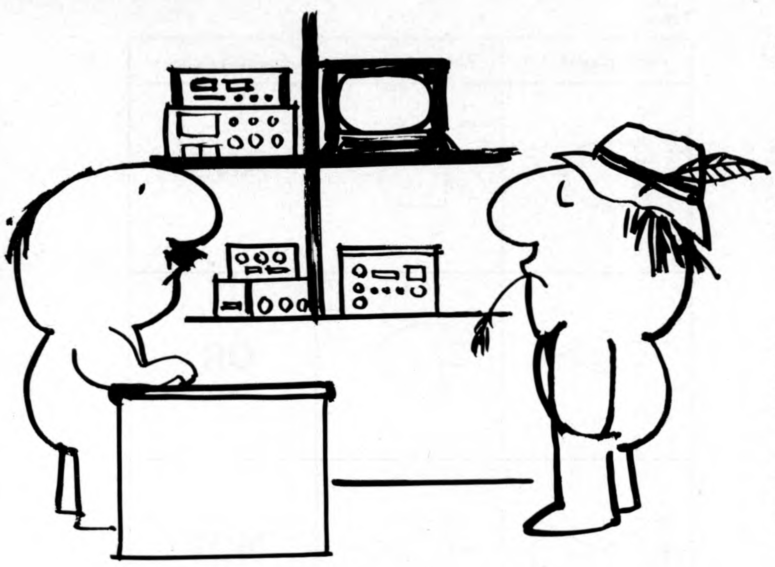
"Of course not," snapped Bill. "I meant, we may have the answering problem solved."

Jim's eyebrows expressed his scepticism, even before he said: "You've found a spare two thousand quid for an answering machine with printout?"

"We don't need one. Look, there's ample spare capacity on our stock control Z-5000. Add a couple of megabyte BRAMs for data storage . . . What's that? Sixty, seventy pence? . . . Then a standard speech analyser chip: another forty pence . . . Plus a couple of capacitors and pullup resistors, with switches: three pounds twenty more, if we go for economy . . . And we're on-line."

A glazed vacancy had entered Jim's expression. "Um," he said, "Who's writing the software?"

"You, of course," said Bill.



11.15
Naturally a customer arrived to spoil their morning. With him he brought a handbook, which he flapped helplessly around him as if swatting flies.

"This, ah, book," he ventured, eyeing Bill and Jim by turns. "It, ah, reads oddly."

Jim and Bill tried to appear concerned. They had both spotted the famous "Rising Sun" logo on the book's cover.

"In what way?" asked Jim helpfully.

"Well, ah, *very,*" said the man. "I mean, it has been, ah, *translated,* I suppose?"

Jim nodded sagely. "Why of course, sir."

"Oh." The man riffled through the pages, all thumbs in his diffident nervousness. "Well, I did wonder. For instance — "He paused on his search — "It says here: . . . *being particular make secure shifting multiplex bus C-56 is strobed in clamp — sense staticised before initiating transcendental function processor* . . ." His already quiet voice faded into silence. He looked helpless.

Bill coughed. "What, exactly, made you need to refer to the manual?" he asked.

"Well," said the customer, "the inbuilt instruction programme wouldn't work, so I thought . . ."

Bill gazed sadly at Jim, who nodded. "Just bring back the old chip," said Bill. "We'll replace it free of charge. No sense in ploughing through the printed handbook for the sake of a couple of quid."

Lunchtime in the pub
Bill belched gracelessly and set his pint of beer on the bar. "Saw on the T.V. last night," he announced to Jim, who was fitting his face around a ham roll, "that arsonists burned down the near-beer brewery. Question is, though: who did it? The "guild of good beer" people, fed up with picketing it for over a year? Or the brewery's owners, for the insurance money?"

"A blessing either way," mumbled Jim through half-chewed ham and bread.

"Filthy automated fizz factory."

Jenny said, from beyond Bill: "Oh, it was probably quite clean, really . . .

'Did you ever try their keg bitter?" demanded Jim. "Expensive, gassy. In fact, a disgraceful application of cybernetics, the sort of rubbish that gives honest computers a bad name. Our old clockwork office machine does better!"

Jenny grinned. "Perhaps the guild should change it's name. How would 'Guild of Good Programming' sound?"

"You still wouldn't catch me drinking that foul stuff. Give me no-nonsense straight-from-the-wood ________"

"Nectar?" suggested Jenny.

Jim did not rise to the taunt, but chewed his roll in silence.

Bill drained off some of his own beer. "Still," he said, "got to hand it to the brewery; they did choose a good computer."

"Fat lot it did for the product," said Jim. "Trouble is, no amount of variable architecture cleverness is going to make up for these idiots who insists on inventing new languages to confuse the machines. *That's* what Jenny's Guild of Whosits should concentrate on. What we need," he added, warming to his pet peeve, "is a machine that knows better than the programmer and stays one line ahead of him."

Bill drained his glass. "If you mean a computer that has a mind of its own," he said heavily, "I think I've got one of those!"

2.30 p.m.
Jim stuck his head around the door into Jenny's office. "The PROM — Burning Bar" . . . he began. "We're going to have to add at least four more stations."

"Aren't six enough?"

"They are if you want a queue down to the supermarket and people camping overnight on the pavement outside."

"Oh, damn!" said Jenny, brushing her hair out of her eyes and a folder of microdiscs off her desk as she stood up. Together they walked through to the connecting door into the front shop.

A crowd was milling about the counters; the two assistants were barely keeping pace with the press of business, whilst the vending machines were doing a non-stop trade in small spares.

Then Jenny smiled. "do you suppose," she asked, "that there'd be much demand for packed meals and soft drinks, as well as the coffee."

5.30 p.m.
The phone rang as Bill was on his way out. Unlocking the door, he re-entered and scooped the handset up to his ear.

It was one of the local farmers, an old and valued customer, so he bit back his standard line about being shut and even decided against claiming to work for the Battersea Dogs' Home. Instead, he allowed the worried man to ask:

"Bill, lad, where've they 24-bit micros got to? Tha' promised I'd have them prompt be Christmas for t'new cowshed."

"Got my doubts," said Bill. "Sorry. But you know how it's been lately, Mister Arkwright. Lot of American promises — They can design, all right, but can't deliver. The real quality's Japanese, these days. And you know I'd not palm off seconds on *you.*"

"Ee, lad, I knows that, but __"

"Just a few months longer, I promise. Maybe the Koreans will start second-sourcing soon."

"But —"

"Mister Arkwright, you want stuff that works, you got to go to the manufacturers with the cash for decent R & D, Right?" He waited for any objections. There were none. "We'll be in touch soon," he promised, as he replaced the headset. Making for the door and home, he felt content. Another day ended. Soon be the weekend.

THE GATES OF REASON

Patrick Sutton

Standard Logic

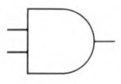
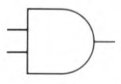
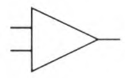
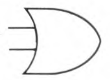
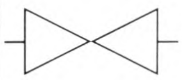
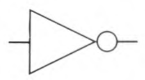
IN THE FIRST ARTICLE IN THIS SERIES, THE READER WAS INTRODUCED TO THREE OF THE BASIC GATES USED IN LOGIC DESIGN: THE AND, OR AND NOT GATES. IN THIS ARTICLE, I WILL DESCRIBE THREE MORE GATES AND GO ON TO INCLUDE ONE OF THEM IN THE CIRCUIT FOR A *FULL ADDER.* BUT FIRST OF ALL A NOTE ON THE LOGIC SYMBOLS WHICH WILL BE USED FROM NOW ON.

The gates described in the previous article were allocated symbols which are infrequently encountered in articles to logic design. It has been decided, that a standard symbol form will be adopted. Until 1973, two main standards were employed, these were: 1) The American Standard Graphic Symbols for logic Diagrams. IEEE STD 91-1962, ASA Y32 14-1962; and 2) The Military Standard Graphic Symbols for logic Diagrams. MIL STD 806B, 25th Feb 1962. In 1973 these two standards were superseded by: The IEEE Standard Graphic Symbols for Logic Diagrams. IEEE STD 91-1973, ANSI Y32 14-1973.

In this new standard, both rectangular and distinctive shapes were permitted in order to represent logic functions graphically. It is these symbols that will be adopted here. The reason for introducing a new symbol set is to minimise any confusion when comparing circuits discussed here with those presented by engineers and manufacturers of computers and related logic circuits. This particular set of symbols has it is most in line with usage.

Table 1 shows the half adder described in the first article, redrawn using the IEEE standard symbols. It is hoped that the slight inconvenience incurred now will be more than balanced by the future consistency of representation.

Table 1

Previous symbol	Standard symbol	Logical function
		AND
		OR
		NOT

Previous symbol denotes that symbol used in the first article. Standard symbol is the IEEE standard symbol for that particular logic function.

When is a gate NOT a gate?

Table 1 shows the symbol for the inverter or NOT gate. Note the circle at the *output* side of the symbol. This circle is often called an "inversion bubble" and can be added to the symbols for the AND or OR gates to represent their inverted forms, the NAND and the NOR gate. NAND simply stands for *not* AND; and NOR stands for *not* OR. These gates are shown in figures 2 & 3 along with their input-output tables (sometimes known as truth tables).

You can see that the NAND gate quite simply changes the output of an AND gate to its exact opposite, ie *inverts* its output. Similarly, NOR inverts the OR output. These additional gates allow further"decisions" to be made and "questions" to be answered by a logic circuit.

For the sake of simplicity, consider the two-input forms of these gates.

The NAND gate gives a logic 0 output when both of its inputs are at logic 1. It gives a logic 1 in all other cases. This gate signals the state that at least one of its inputs is not at logic 1.

The NOR gate gives a logic 1 output when both its inputs are at logic 0, and a 0 output for all other cases. This gate signals the state that both inputs are at logic 0.

Let's take a somewhat improbable example as an illustration.

Imagine a car park which has two parking spaces, both of which are fitted with sensors which indicate when a car is parked by outputting a voltage (logic 1). Now suppose the owners want to fix two lights at the entrance, one of which indicates that the car park is full and the other indicating that there is space available. We have encountered two gates which will perform this function — the AND and the NAND gates, see figure 4.

Compare the *light on* with the logic 1 in the input-output tables given for these gates.

You may consider the use of two lights somewhat excessive as it is obvious that if the car park is NOT full, then there is space for parking. Of course, in coming to this conclusion you have simply *inverted* the AND operation and performed a logical NAND operation. Even in this simple

Figure 1 The Half Adder

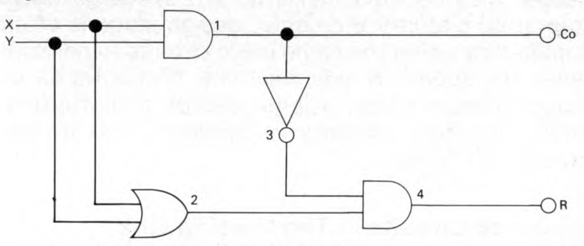


Figure 1.4A of "The Gates of Reason" PCW Vol 1 No 1, *redrawn* using IEEE STD symbols.

example we see that a human being can make a deduction which a machine must be wired up or programmed to do.

Figure 4 shows also the OR and NOR gates in relation to this problem. In this case they are answering the question: are there any customers? Again it is obvious that if the car park is empty, then there are NOT any customers. Here again the gates are performing trivial functions. However, suppose the car park is a multi-storey one. In this case the gates could be of considerable use in organising its daily running.

Of course, here the circuit would be far more complex, requiring multiple-input gates or many double-input ones.

Both the NAND and the NOR gate will be met again in future articles. It is important at this point to note that the NAND gate is probably the gate most frequently incorporated in logic circuit.

An Exclusive Gate

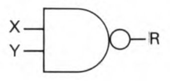
The OR gate we have examined is sometimes called an *inclusive* OR. This means that it gives a logic 1 output if either of the inputs is at logic 1, and if both of its inputs are at logic 1. It *includes* the state where both the inputs are at logic 1. It's a pretty generous gate, unlike the AND which will only let through (output) a logic 1 if both its inputs are logic 1s.

There is another form of the OR gate called the EXCLUSIVE--OR. Its symbol and input-output table is given in figure 5. It can be seen that this gate only outputs a logic 1 if its inputs are in opposite states; that is, one is at logic 1 and the other at logic 0. The exclusive OR is a pretty snobbish kind of gate; it can't stand being crowded: if both its inputs are in the same state it outputs a logic 0.

In order to illustrate the function of this gate, let us go back to the car park problem. Suppose that the

Figure 2 The Nand Gate Input Output Table

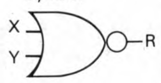
Symbol



Input		Output
X	Y	R
0	0	1
0	1	1
1	0	1
1	1	0

Figure 3 The Nor Gate Input Output Table

Symbol



Input		Output
X	Y	R
0	0	1
0	1	0
1	0	0
1	1	0

owners of the car park want to know if they are making a profit (car park full), a loss (car park empty) or breaking even (one car in the park). Obviously the AND gate can detect a profitable state and the NOR gate a loss. The exclusive OR can be used to unambiguously detect the state of breaking even. Notice here that the inclusive OR could not distinguish between profit and breaking even.

Another way of expressing the exclusive OR operation is: X OR Y but NOT X AND Y. As an exercise try to draw an equivalent circuit for the exclusive OR. In this case the word 'but' can be replaced by AND.

Back to the sums

Looking again at the input-output table for the half adder, compare the R column with the output column for the exclusive OR gate. As you can see, they are identical. The C_0 or carry out column of this table is identical to the output of an AND gate. Thus it can be seen that a half adder circuit can be

Figure 4 The Car Park Problem

Inputs	'And'	'Nand'	'Or'	'Nor'
Parking spaces	Full?	Spare for more?	Any customers?	Empty?
▢ ▢	□	✹	□	✹
▢ ▨	□	✹	✹	□
▨ ▢	□	✹	✹	□
▨ ▨	✹	□	✹	□

achieved by using only two gates; the exclusive OR and the AND gate. Figure 6 shows the circuit diagram for a half adder using this configuration.

Introducing the Full Adder

When addition is performed using a full adder, account is taken both of the two bits to be added and the carry in from any previous additions. The output consists of the sum and carry out. The input-output table was given in Table 1.1 of the first article.

It is easier, in order to design a full adder, to think of the full bit addition as two operations:

1) Addition of the two input bits to form a partial sum and a partial carry. 2) Addition of the partial sum and the carry in to form the sum and a second partial carry.

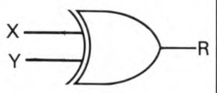
Either the first OR the second partial carry, produced by these operations, forms the *carry out.*

Table 2 shows an input-output of a full adder implemented by using two half-adders. The two half adders take care of steps 1 and 2 above. An OR gate forms the final carry out. The circuit for this full adder is shown in figure 7. Compare the input-output table given here with Table 1.1 in the first article.

Figure 5 The Exclusive — Or

Input Output Table

Symbol



Input		Output
X	Y	R
0	0	0
0	1	1
1	0	1
1	1	0

Figure 6 A Half Adder using 2 Gates

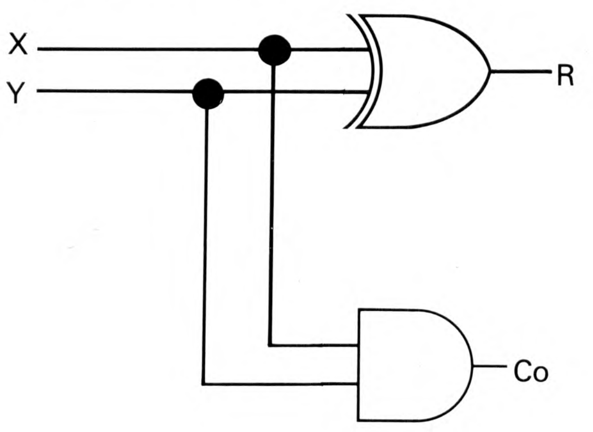
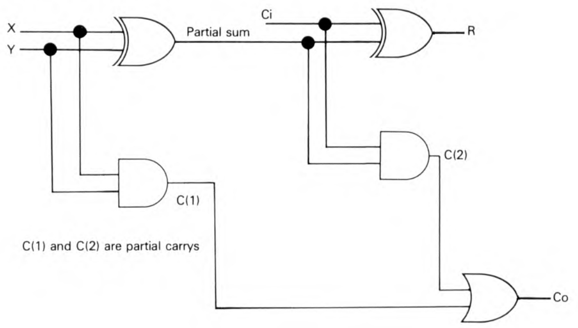


Figure 7 The Full Adder



C(1) and C(2) are partial carrys

In Table 2 the partial carry (1) is formed by the logic And operation (X AND Y). The first half adder forms the partial sum by the operation X exclusive OR Y.

The partial carry (2) is formed by the operation — partial sum AND Ci in the second half adder.

The carry out is formed by C(1) OR C(2). Note that the case where C(1) and C(2) are both logic 1 *does not occur* here. This fact will be used later when the circuit is constructed.

It is left to the reader to check the results in the various columns in Table 2. It may help to split up this table into separate input-output tables for each operation.

This particular circuit illustrates an important point in logic design. It is often preferable to construct complex circuits by combining two or more identical sub-units. This is because the more a circuit is used, the greater are the numbers produced and the cheaper they become. In terms of LSI (Large Scale Integrated) circuitry the circuit design stage is often simplified by using the same basic circuits to perform many operations. If you examine photographs of microprocessor chips, a large degree of patterning within the chips' circuitry is apparent, due to this approach to design.

Table 2 The Full Adder

Inputs						Outputs	
						Sum (R)	Carry (Co)
X	Y	Ci	Partial carry C(1)	1st half adder partial sum X Plus Y	Partial carry C(2)	2nd half adder partial sum plus C(1)	C(1) or C(2)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	1	0	1	0
0	1	1	0	1	1	0	1
1	0	0	0	1	0	1	0
1	0	1	0	1	1	0	1
1	1	0	1	0	0	0	1
1	1	1	1	0	0	1	1

Integrated Circuits — The Missing Link

Nowadays, construction of logic circuitry is greatly simplified by the introduction of a series of integrated circuits — the 74 series. These circuits are supplied in plastic packages with connecting pins at either side forming what is known as a *dual in line* or DIC package. The chips in this series have straightforward power requirements of +5 volts and 0 volts (earth). Building logic circuits, using these devices, is almost as simple as drawing the original diagram: they form the "missing link" between diagram and operational circuit.

The 74 series pin connections for the gates already described, are shown in figure 8. The notch, illustrated at the left hand side of each diagram, is present on the chips themselves as an indentation in the plastic package. This marker aids correct orientation of the package and is often supplemented by a dot, or circular indentation in the package, indicating the position of pin number 1. The pins are numbered so as to increase in an anticlockwise direction from pin 1.

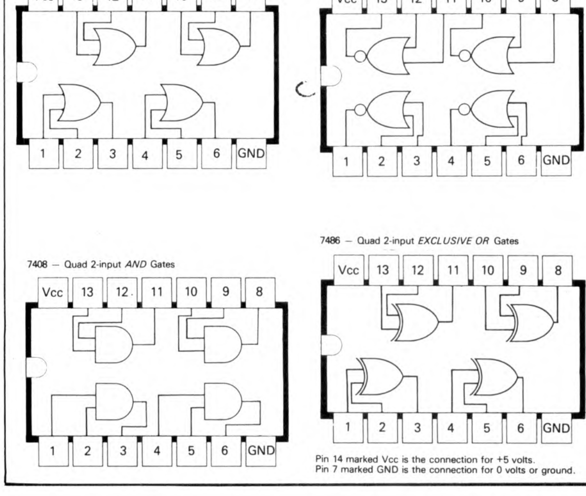
For more detailed information on these and other integrated circuits in the 74 series, the reader is directed to many of the excellent texts covering the topic of *transistor-transistor logic* (TTL) integrated circuits.

For the purpose of demonstration, we will confine ourselves here to simple logic experiments and to constructing the full adder circuit, using this series of chips.

Experimenting with the 74 Series

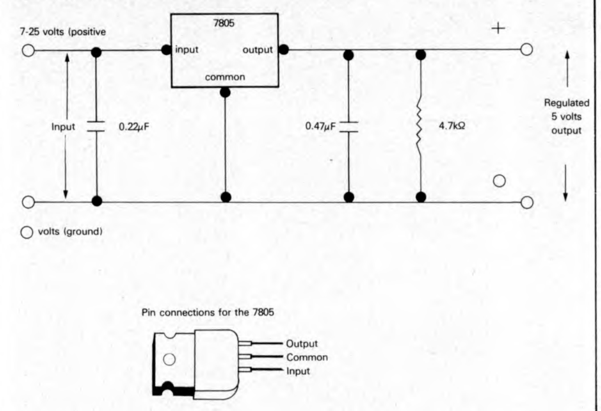
In order to experiment with these integrated circuits, it is perhaps advisable to purchase a

Figure 8 Part of the 74 Series



Pin 14 marked Vcc is the connection for +5 volts.
Pin 7 marked GND is the connection for 0 volts or ground.

Figure 9 Using the 7805 voltage regulator



prototyping board. This is a board containing many contacts, spaced at 0·1 inch intervals, forming a grid. The contacts are usually made from nickel-silver and are set into a plastic board. This arrangement allows both the integrated circuits and their connecting wires to be plugged directly into the board, thus eliminating the need for soldering. The major advantage to the experimenter is that the logic circuits may be built and taken apart very simply, allowing the re-use of each component and wire.

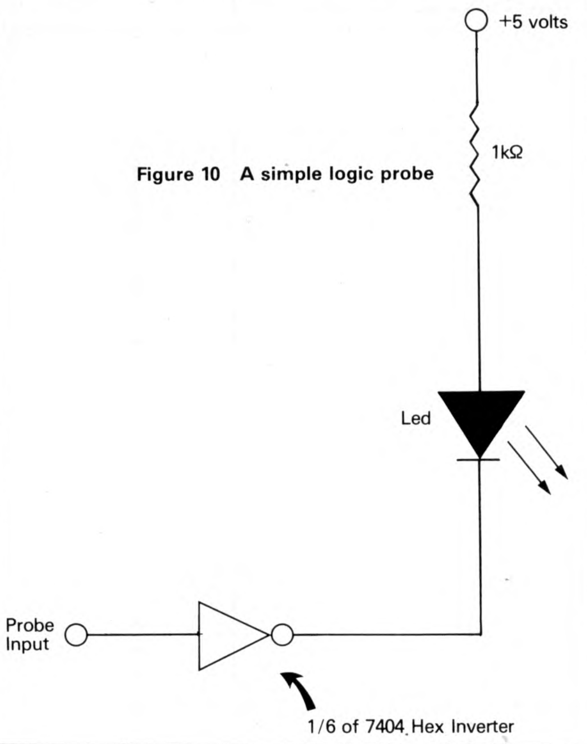
If you do not possess a power supply, it might be as well to buy one which supplies ± 15 volts. This supply can then be used for a variety of circuits. In order to generate the required +5 volts from such a source, another integrated circuit — the 7805 voltage regulator, may be used. Figure 9 shows a circuit for generating a regulated +5 volt supply from an input of 7 to 25 volts positive. If many integrated circuits are to be driven by the regulator (it can supply up to 1 amp), then a heatsink should be attached to the metal part of the package.

In order to check the outputs of the circuits and their various stages, a voltmeter or better still a logic probe may be used. Since this latter device is relatively expensive, a simple circuit for a logic probe is shown in figure 10.

This circuit uses an inverter or NOT gate and a light emitting diode (LED). One sixth of a 7404 chip provides the inverter. Refer to figure 8 to find the appropriate pin connections. This method of constructing a logic probe, has the advantage that it uses an inverter chip which can also be used for future logic experiments. The input of the NOT gate provides the input for the probe and should be connected to the output to be tested. If the LED is on, upon connection of the probe to the test point, then that part of the circuit is at logic 1. If the LED is off, then the output is logic 0. Note that even if no connection is made to this probe, the LED is still on. This is because, in the 74 series, the inputs to the logic gates 'float' to logic 1 if they are not connected. Should the LED prove too dim when on, the 1kΩ resistor in the probe circuit may be replaced by 470Ω.

For some introductory experiments, verify the input output tables for the gates described. In order to accomplish this, the relevant chip must have its Vcc pin (pin 14) connected to +5 volts, and the GND pin (pin 7) connected to 0 volts or ground. Connect the inputs of the gates to +5 volts (logic 1) or ground (logic 0) to form the various inputs combinations given in the tables, and check the output using the probe.

Figure 10 A simple logic probe



CONSTRUCTING THE FULL ADDER

Figure 11 is the connection diagram for a full adder using the EXCLUSIVE OR gates and the AND gates. It was noted earlier, that the partial carries from both the half adders employed in this circuit, are never simultaneously at logic 1. Because of this fact, one of the spare EXCLUSIVE OR gates in the 7486 chip may be substituted for the OR gate in figure 7.

The solid circles in figure 11 represent circuit connections and the hollow ones represent input and output connections. In this experiment, as in the last, vary the inputs by attaching them to +5 volts or ground and test the outputs with the probe.

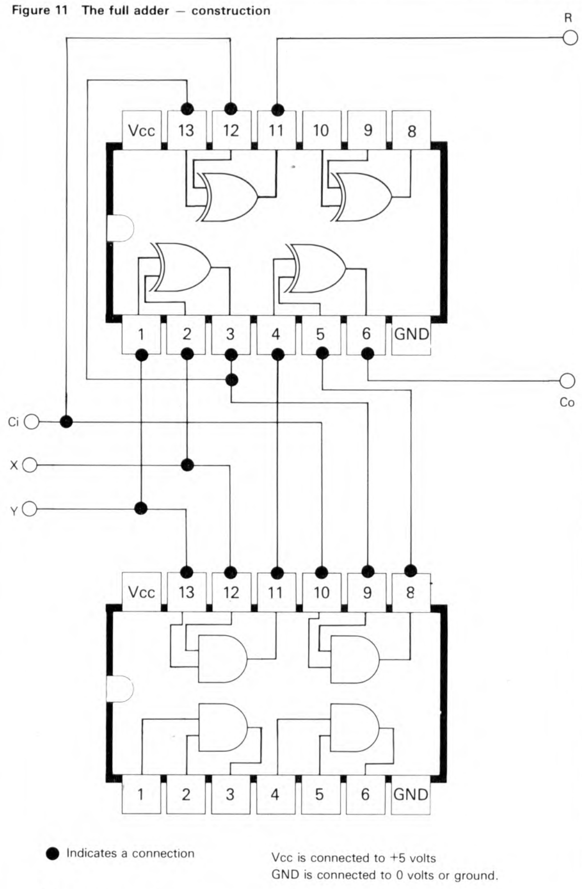
When you have satisfied yourself that this circuit does in fact fulfil the requirements of a full adder, further experiments may be undertaken. Here are some suggestions: —

1. Add a clocked circuit to the outputs of the full adder by using the two spare AND gates in the 7408 chip. In order to do this, connect the R output to an input of one of the AND gates, and connect the Co output to an input of the other spare AND gate. For the clock, connect a wire to both of the remaining inputs of the two AND gates. With the probe at the new R output, connect the clock wire to +5 volts and 0 volts alternately. Notice that the true sum output is only given when the clock wire is at +5 volts (logic 1). When the clock is grounded (logic 0), the output is at logic 0. Repeat this experiment with the logic probe connected to the new Co output. From this simple experiment, it should be apparant that a clock can make the outputs of logic circuits *predictable at certain times;* in this case when the clock is at logic 0.

2. What happens if you put the clocked AND gates at the inputs to the adder? For this experiment you will need one more AND gate.

NOTE In both the clocked gate experiments, if no connection is made to the clock inputs, of the AND gates, they will float high and their outputs will be enabled.

Figure 11 The full adder — construction


R
Vcc
13
12
11
10
9
8
1
2
3
4
5
6
GND
Co
Ci
X
Y
● Indicates a connection
○ Indicates an input or output
Vcc is connected to +5 volts
GND is connected to 0 volts or ground.

3. Construct two half adders using the two different diagrams given for this circuit; figures 1 and 6. Confirm that these circuits are functionally equivalent.

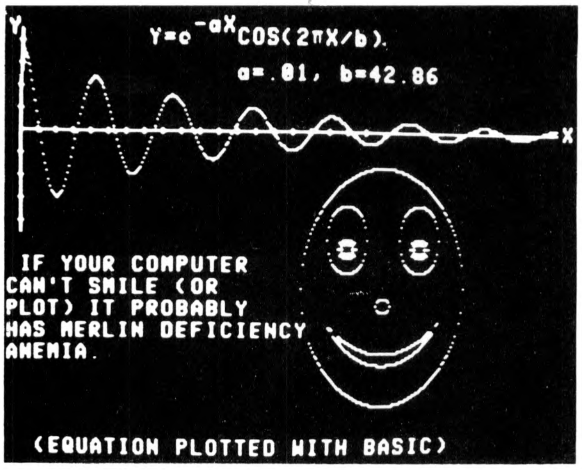
COMPONENT LIST

a) *GENERAL EXPERIMENTS*
(i) PROTOTYPING BOARD
(ii) POWER SUPPLY: +5 VOLT REGULATED OR 7-25 VOLTS (POSITIVE) IF USED IN CONJUNCTION WITH THE REGULATOR CIRCUIT.
(iii) A SELECTION OF THE 74 SERIES CHIPS.

b) *5 VOLT REGULATOR CIRCUIT — FIGURE 9*
1 7805 VOLTAGE REGULATOR CHIP & HEATSINK IF REQUIRED
1 0.22μF CAPACITOR
1 0.47μF CAPACITOR
1 4.7 kΩ $\frac{1}{4}$ WATT RESISTOR

c) *LOGIC PROBE CIRCUIT — FIGURE 10*
1 7404 HEX INVERTER CHIP
1 1 k$\mu\Omega$ WATT RESISTOR
1 LIGHT EMITTING DIODE

d) *FULL ADDER CIRCUIT — FIGURE 11*
1 7402 QUAD 2-INPUT AND GATES CHIP
1 7486 QUAD 2-INPUT EXCLUSIVE OR GATES CHIP.



```
Y=e^-aX COS(2πX/b).
a=.01, b=42.86

IF YOUR COMPUTER
CAN'T SMILE (OR
PLOT) IT PROBABLY
HAS MERLIN DEFICIENCY
ANEMIA.

(EQUATION PLOTTED WITH BASIC)
```

NOW THE FAMOUS MERLIN WITH SUPERDENSE IS AVAILABLE IN THE UK — and with improved graphics resolution and 25% more lines of text!
On European standard TV sets the best ASCII/graphics boards on the market for the S100 bus give 25 lines of ASC11 or 320 horizontal x 250 vertical points of DMA bit mapped graphics.

If you have been considering an ASCII/graphics interface for your system, or if you hadn't been considering one because you thought good graphics was too expensive, consider the MERLIN with SUPERDENSE add-on. It's the best there is and its at an affordable price . . . Kit, only £275

CONVERT A STANDARD SELECTRIC TYPEWRITER TO AN RS232 TERMINAL
The Sharp & Associates kit to convert an office Selectric is now available in the UK. Full kit with reader/punch port and RS232 interface, conversion instructions and manual, only £540.00

For full details of these and many options (and also our other personal computer equipment such as Assembled 8K memory boards for £120)

Write or Phone

J & A COMPUTERS
15 FLEETWOOD GARDENS
MARKET HARBOROUGH
LEICESTERSHIRE
0858 7620

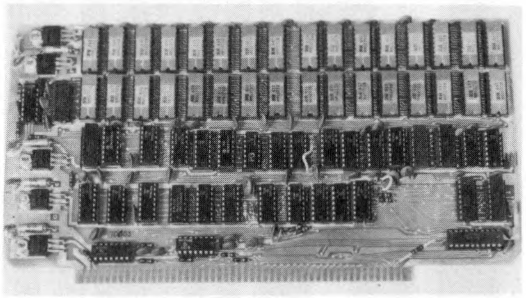
comart

SPECIALISTS IN MICROCOMPUTERS

Cromemco Z2 Processor

* Power of Z80 Processor Card with Power-on-jump
* S100 Bus 21 slot Backplane to configure the largest system
* 30 Amp Power Supply

  Z2 = Z80 + S100
* Range of Analogue & Digital Interfaces
* Powerful Disc based Software



Dynabyte Memory

16K Dynamic for Altair, Imsai, Poly 88.

16K Static with memory backing facilities, and individual 4K boundary location and write protect.

32K Static for large system economy.

Assembled, tested and 1 year's warranty at list prices.

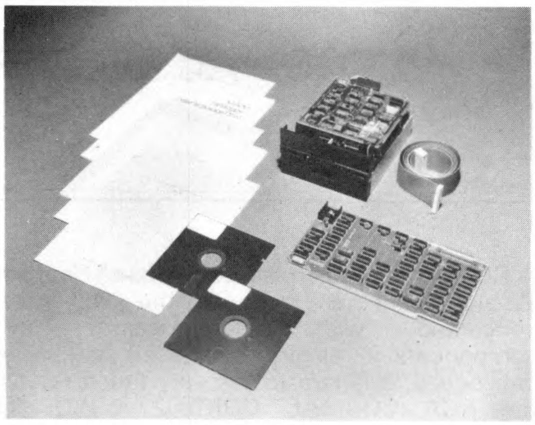
North Star Mini-diskette System

With operating system, File & String handling

BASIC

For all 8080 & Z80 S100 Bus Systems.

For greater calculation speed use the Hardware Floating Point Board.



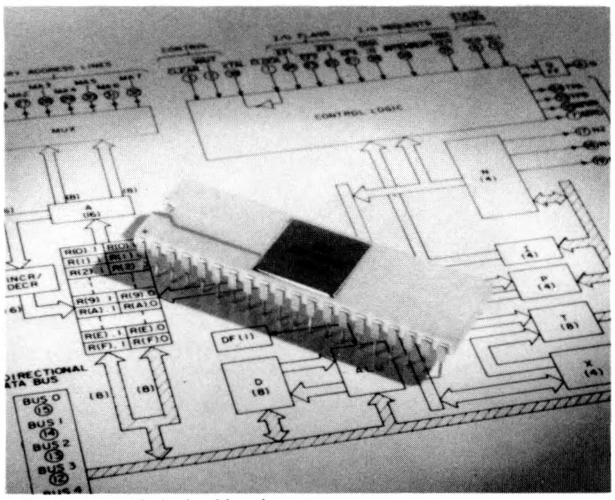
Comart systems are available by direct mail order or through local suppliers:-

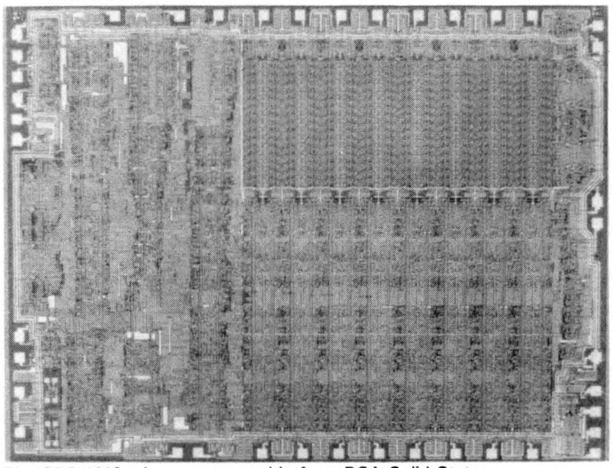
Xitan Systems Ltd.,
31, Elphinstone Road,
Highcliffe,
Dorset, BH23 5LL
Phone: Highcliffe 77126

Computer Bits Ltd.,
41, Vincent Street,
Yeovil,
Somerset.
Phone: Yeovil (0936) 26522

Our range of products is constantly being expanded - write or phone for catalogue of latest controllers, diskette and memory systems and options.

COMART LIMITED, P.O. BOX 2, ST. NEOTS, CAMBS. PE19 4NY TEL. 0480 74356


RCA's CDP 1802 single-chip microprocessor.


The CDP 1802 microprocessor chip from RCA Solid State.

A MIGHTY MICROMITE

A COSMAC MICROPROCESSOR BASED SYSTEM

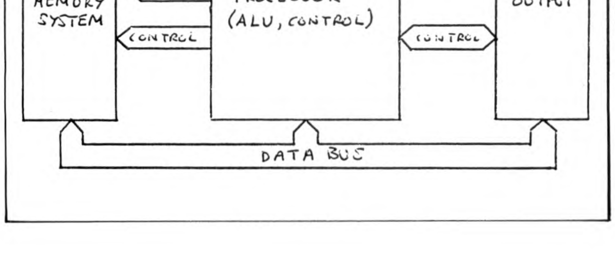
K. R. James

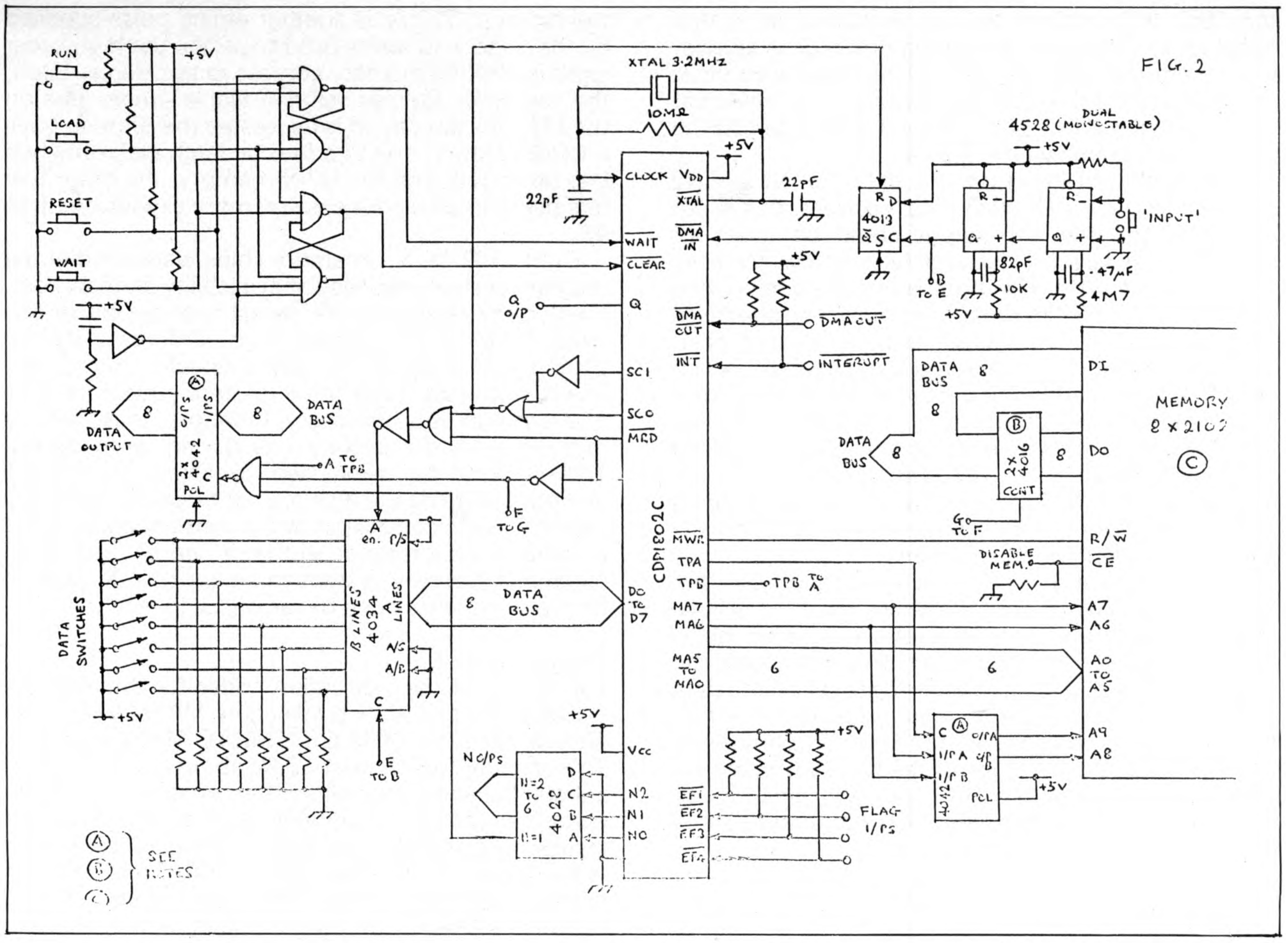
A COSMAC Microprocessor Based System
When scanning through the advertisements concerned with microprocessor (M.P.U.) components in electronics biased magazines and catalogues, it is rare to see very much reference to the RCA COSMAC CDP1802 M.P.U. This is reflected in a general lack of knowledge of this family of devices, which do in fact offer a number of advantages for people just starting out in the world of M.P.U's. This article is intended to show how the 1802 M.P.U. can be used in a fairly simple computer system, and to highlight the advantages of using this particular M.P.U. over other types of M.P.U.

The 1802 is an all C-MOS device which results in very low power consumption, so it is ideal for applications requiring small power supplies (e.g. batteries). This, however, does not mean that it is not suitable for other applications. The key features of the 1802 are:—

(i) Single power supply of 5V with the CDP1802C, and 3-12V with the CDP1802.
The CDP1802 can run twice as fast if the supply voltage is increased from 5 to 12V. For this reason the 1802 is supplied with two +ve power supply pins, allowing the internal circuitry to run at 12V, while all input and output levels are T.T.L. compatible (i.e. 5V max.)
(ii) Full C-MOS, T.T.L., D.T.L. and N-MOS compatibility
(iii) On chip clock oscillator
(iv) Simple control
(v) 16 sixteen bit registers which are all available to the user.
(vi) Serial output bit available
(vii) Direct control of up to 7 I.O ports.
(viii) On chip D.M.A. (Direct Memory Access).

In most other respects it is similar to other makes of M.P.U. i.e. it is an 8-bit parallel device that can be used with up to 64K bytes of mixed ROM and RAM. It also has a maskable interrupt input.





The D.M.A. facility is worthy of special mention, since it allows the easy entry of programs and data without special ROM's or extra external logic to gain direct access to the memory system. It is this D.M.A. facility that makes the 1802 particularly attractive to people new to M.P.U's.

The maximum obtainable speed for the 1802 is 1.25 us per machine cycle, using a clock frequency of 6.4 MHz, with most instructions requiring 2 cycles, although some jump and skip instructions require three. This is not the fastest M.P.U. on the market, but speed is rarely important, especially for newcomers to the field.

Basic System Requirements

The basic requirements of a simple computer system are as follows:— (Also see fig. 1).

(a) Central processor (This is the M.P.U.)
(b) A memory system.
(c) Input-Output (I.O) circuits.

RCA produce a full range of C-MOS memory products, but these are silicon on saphire devices, and are expensive; it was decided, therefore, to use standard N-MOS memories in the system described here. RCA also produce a complete range of I.O devices, including, I.O ports, decoders, a UART, bus separators, and a video controller. Unfortunately, and probably because of the present lack of interest in the 1802, the prices of these devices are expensive for what they are. It was decided, therefore, to use standard C-MOS devices for I.O and M.P.U. control functions. (It is worth noting that many users of other M.P.U's tend to do the same, but with T.T.L. instead of C-MOS).

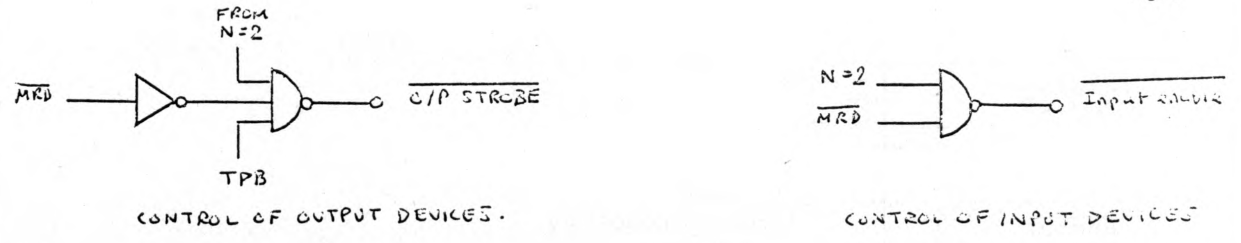
The basic system described here provides:—
IK of RAM.
One parallel output port.
One input port for direct program loading using the D.M.A. facility.
A serial output for use under program control.
An interrupt input.
4 program defined input flags.

It is also designed to be easily expandable, i.e. addition of more I.O ports, memory and more complex interrupt, etc.

Circuit Description

The complete circuit diagram of the system is shown in fig (2). The Reset, Wait, Load and Run Control logic is as described by RCA in their users manual for

Fig. 3



the 1802 (this manual will be discussed in further detail later). The standard oscillator circuit is shown, but an external clock can be fed into the Clock input pin if required; if this is done, however, it is important not to load the XTAL pin. With a 3.2 MHz crystal the machine cycle time will be 2.5 us.

The I.O control lines from the 1802, N0, N1 and N2 are used to drive a CD4028 octal decoder, this is not essential, just the N0 output could be used for example, but using a decoder here allows for easy I.O expansion. The N1 output is used in conjunction with $\overline{\text{MRD}}$ and TPB to provide an output strobe pulse to the two CD4042's which form the output port. TPB is a timing pulse (internally generated) which indicates that the 1802 is ready to move data from the memory to the output port, and $\overline{\text{MRD}}$ indicates the direction of data flow along the data bus. When $\overline{\text{MRD}} = 0$ (logic 0 = OV) data is read from the memory to the M.P.U. or output port, and when $\overline{\text{MRD}} = 1$ (logic 1 = 5V) data is read from the I.O device to the M.P.U. and memory.

The input port is used in conjunction with the $\overline{\text{DMAIN}}$ line. Two monostables are used to remove switch bounce and provide a short duration pulse (which must be less that 5 us), this pulse sets the CD4013 flip-flop, and strobes the data set on the data switches into the CD4034 input register. The flip-flop, after being set, asserts the DMA IN command, which, when the M.P.U. is ready establishes the DMA state code on the SC0 and SC1 pins. These pins are outputs that indicate the present state of the M.P.U. and their truth table is:—

State Name	SC1	SC0
S0 (Fetch)	0	0
S1 (Execute)	0	1
S2 (DMA)	1	0
S3 (Interrupt)	1	1

The DMA state is decoded to be combined with $\overline{\text{MRD}}$ to produce the A enable signal required to activate the outputs of the CD4034 input register. The decoded DMA state signal is also used to reset the flip-flop. T.P.A. is another timing pulse supplied by the 1802, and this is used to strobe the high-order eight bits of the memory address to an address latch, the low-order address being available shortly later on the MA pins directly. It is necessary therefore to have a CD4042 latch. This is a four bit latch with only two bits being required for 1K of memory, the other two bits being available for easy memory expansion up to 4K.

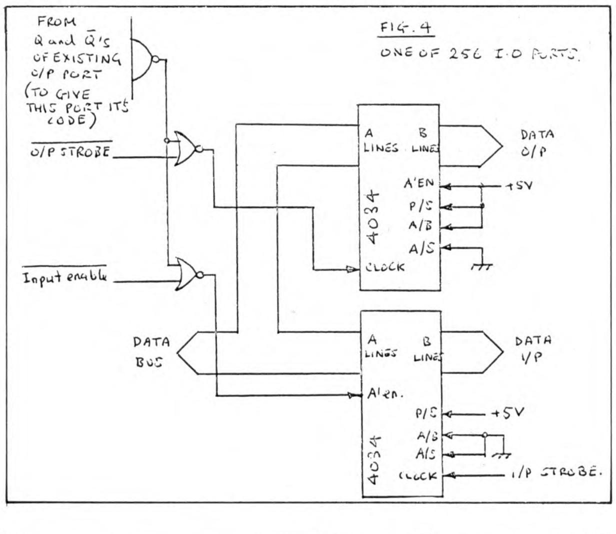


FIG. 4
ONE OF 256 I.O PORTS.
FROM Q and $\bar{Q}$'s OF EXISTING O/P PORT (TO GIVE THIS PORT ITS CODE)
$\overline{\text{O/P STROBE}}$
$\overline{\text{Input enable}}$
DATA BUS
A LINES
B LINES
DATA O/P
A'EN
+5V
4034
P/S
A/B
A/S
CLOCK
DATA I/P
A'en.
I/P STROBE.

Eight 2102 1K X 1 memory chips were used in the memory system, because of availability and low cost, these chips represent the lowest cost per bit of any static RAM at the present time. Unfortunately the 2102 is not designed for going directly onto 2-way buses, although it has got a tri-state output for ease of expansion. It is necessary, therefore, to isolate the outputs from the system bus during write cycles. This is done by using two packages of CD4016 anologue switches, which are controlled by the $\overline{\text{MRD}}$ signal. The $\overline{\text{CE}}$ control pin of the memory chips is tied to ground via a resistor so that it can be used in an expanded memory system to inhibit this IK page of memory, which is done by taking CE to = 5V.

Programming

Full details of the instruction codes for the 1802 are available in the RCA publication MPM-201A, User Manual for the CDP1802 COSMAC Microprocessor. This manual (which cost £3.00 about a year ago), is substantial and very well written, which is a rarity for computer books. It contains descriptions of all the instructions available to the user (a total of 255) along with details of interfacing the 1802, and detailed timing diagrams for all 1802 operations. It also contains a chapter on programming techniques. For anybody interested in the 1802, this book is an 'essential' along with the device data sheet, which RCA supplied me with, free of charge.

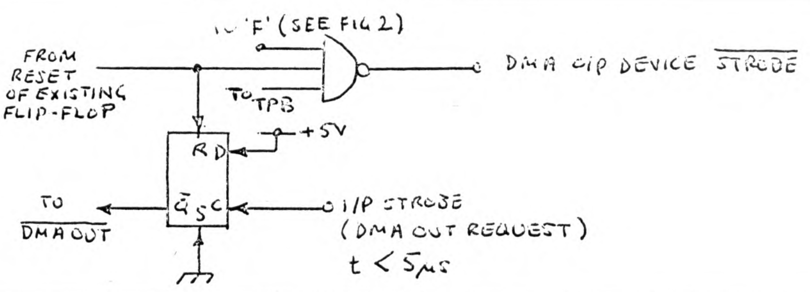
When a program has been written, it can be entered into memory in the following way:—

1). Press the run switch, followed by Reset and Load.
2). Set up the instruction on the data switch.
3). Press the "input" switch.

(3) strobes the data into memory location pointed to by register 0 (R(0)) which is always 0000 (M(0000)) after reset. The 1802 then automatically increments R(0) so that it points to M(0001) for the next instruction.

(2) and (3) can then be repeated until all the instructions have been entered. When the program has been fully entered the Reset switch should again be pressed, which returns R(0) to M(0000), the start of the program. Pressing the Run switch should, one hopes, allow the program to run. When running a single shot program, i.e. a program that should stop when it has been completed, the instruction 00 should be used as the last instruction. This puts the 1802 into and IDLE state.

Since R(0) is used for DMA operations, it is good practice to switch to another program pointer


to 'F' (SEE FIG 2)
FROM RESET OF EXISTING FLIP-FLOP
TO $\overline{\text{TPB}}$
DMA O/P DEVICE $\overline{\text{STROBE}}$
+5V
R D
$\bar{Q}$ S C
TO $\overline{\text{DMA OUT}}$
I/P STROBE (DMA OUT REQUEST)
t < 5μs
FIG. 5
DMA OUT

register, which can be any of the other fifteen scratchpad registers. This should be done in the initialization section of the program.

A useful and quick indication of whether or not the M.P.U. input register and memory system is working is to enter the instructions,

7B (Hex)	Set Q.
00	Stop.

This should when run set the Q output high, and should go low again when Reset is pressed. Also, by holding DMA OUT at OV, the memory address count can be checked, along with MRD, TPA and TPB. (See timing diagrams in the manual). If this is done just after switch on, any data on the data bus will be just random.

The output port can be tested using the following program: —

Instruction Code (Hex)	Description
F8 FF	Load FF into Data reg.
AF	Put data reg into R(F)
EF	Make R(F) the X register.
5F	Store contents of data reg in M pointed to by R(X).
61	Output M pointed to by R(X).
00	Stop.

This program when run should set all the outputs, on the output latch, high, thereby testing the output port. The use of R(F) reduces the number of data switch changes when loading the program.

Extending the system

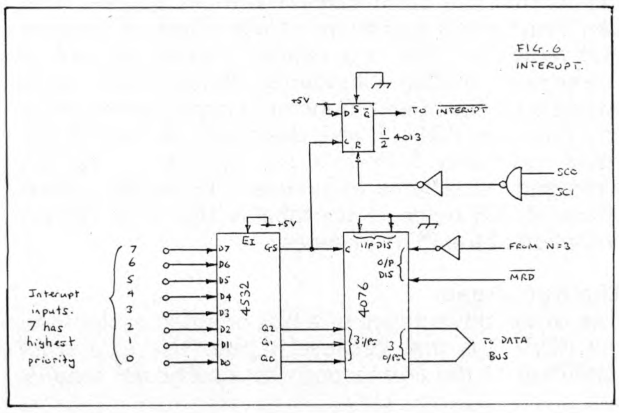
A number of extensions to the basic system are possible, they include: —

(i) Extension of I.O capability, this is easily achieved using the extra circuitry shown in fig (3). The signals produced by this logic could be used in conjunction with the data set up on the existing output port to allow access of up to 256 I.O ports. Fig (4) shows such a port.

The connection of a UART is described in some detail in the 1802 user manual.

(ii) The DMA OUT facility can be utilised, as shown in fig (5). This operates in a similar way to DMA IN, except that data is now read out of, instead of written into, the memory.

(iii) Interrupt facility. The single level of interrupt, provided by the 1802, can be expanded as shown in fig (6). This is a suggested circuit and had not been tried out at the time of writing, it is given as an indication of what is required. This circuit would have to be used in conjunction with a suitable program, pointed to by R(1), which interprets the information that is presented to it. R(1) is always used as the program pointer when the 1802 drops into the interupt mode.



Conclusions

This article is intended only as a brief introduction to the 1802 M.P.U. and further information is available from the following sources: —

1. COS/MOS Integrated Circuits, SSD-250, RCA.
2. User Manual for the CDP1802 COSMAC Microprocessor, MPM-201A, RCA.
3. Subroutine Library for RCA COSMAC Microprocessors, MPM-206, RCA.
4. COSMAC Microprocessor Product Guide, MPG-180, RCA.

Of the above, (4) should be available direct from RCA, at RCA Limited, Solid State-Europe, Sunbury-on-Thames, Middlesex. TW17 7HW. This publication also lists RCA's distributors in the UK, which is where the other publications can be obtained.

It is hoped that this article will arouse some interest in the 1802 M.P.U., which in turn may enable cost of the devices in the 1800 series family to reduce, enabling even further interest.

Notes for fig. 2.

(1) For 4042's	Pol is Polarity,	Pin 6.
	C is Clock	Pin 5.
	1/P's are D 1/Ps.	Pins 4, 7, 13, 14.
	O/P's are Q O/Ps.	Pins 2, 10, 11, 1.
(2) For 4016's	Connect a gate in series with each data output of the memory, and connect the control pins (13, 5, 6, 12) together. (This requires two chips).	
(3) For Memory Chips.	Connect corresponding address pins together for all eight 2102-2 chips. Connect all CE pins together. Connect all R/W pins together.	

Also: —

Supply voltage pins are not shown (except for the 1802) and should be 5V for VDD and OV for Vss for all devices.

Reasonable supply decoupling must be used, and this also is not shown.

In the Author's version the Data O/P's and the Q output were all connected to LED's with driver transistors.

Unless otherwise stated, all resistors are 33KΩ.

If the CDP1802 is used (instead of CDP1802C), VDD should be 12V and XTAL should be 6.4 MHz.

Integrated Circuits Used

1 X CDP1802CD
8 X TMS4034 = (= 2102-2)
1 X CD4001BE } Used for random logic,
1 X CD4011BE } including inverters.
1 X CD4012BE
1 X CD4023BE
2 X CD4016BE
1 X CD4028BE
1 X CD4034BE
3 X CD4042BE
1 X CD4013BE
1 X MC14528B.
Suffix D denotes ceramic package
Suffix E denotes plastic package.

IN PRAISE OF THE PDP-11

Mike Lord

Most people who come into contact with the PDP-11 soon develop an affectionate enthusiasm for it. This article attempts to explain why, and also to show why a design almost 10 years old is still held up as a standard against which new mini and micro computers are judged.

Background

Digital Equipment Corporation introduced the first PDP-11 in 1970. Since then, a whole family of PDP-11 processors and related devices has been produced so that today the PDP-11 is the broadest family of compatible computer products available. Thus as well as a complete range of peripherals including mark-sense readers, IBM-compatible magnetic tape drives, floppy discs, and VDU's with graphics capability, DEC make several processors which use essentially the same instruction set but whose raw processing speed varies by about 10:1. In recent years DEC have introduced the LSI-11 series which, apart from using a simpler bus structure, have the same instruction set and other fundamental characteristics as the original PDP-11's.

This breadth of capability, together with the sheer size of the DEC organisation, enables them to offer a system tailored to meet most users' requirements.

But DEC's size cannot, by itself, account for the success of the PDP-11. Indeed it is more likely that DEC's current leading position in the minicomputer business is a *result* of the popularity of the PDP-11 series. So to explain this success we have to examine the characteristics of the machine itself, and when we do so, two features stand out; the physical realisation of the machine's architecture, and its instruction set. In both cases, the PDP-11 represents an elegant solution to the problems of computer design and use, being designed in a way which is conceptually simple but which results in a powerful tool for the computer user.

Hardware

It is the nature of DEC's business that their machines are used for a wide variety of applications; from engineering design aids to machine tool controllers, from message switching systems to time-shared computers for classroom use. Thus any new range that was to be designed had to be flexible. It had to be able to work with almost any number and type of peripheral, it had to be able to provide a variable amount of computing power — in terms of cycle time and also number crunching ability — and it had to meet a range of budgets.

The Bus

To meet this challenge, DEC engineers developed the 'Bus' concept then coming into vogue, and they developed it to an extent which has not been surpassed in the 10 or so years since the PDP-11 design was started.

The PDP-11 is structured as a number of modules, which are the particular processor, memories and peripherals required for a given system, inter-connected by a high speed bidirectional bus (Fig. 1). This bus carries data, address and control signals between the modules, and transfer of information via

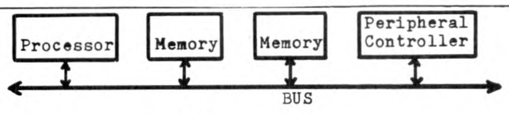


Fig. 1
PDP-11 BUS STRUCTURE

the bus is regulated by a 'Bus Controller', which is usually situated in the processor module. Each module is assigned a priority, and is allowed to use the bus provided that no other module with a higher priority is also wanting to use the bus at the same time. While it is using the bus, a module has complete control in that it determines the direction of data transfer and the source or destination for the data. In this way a peripheral capable of handling data at a high rate, for example a magnetic disc memory, can effectively take control of the system when it is ready to transfer data and route the data correctly *without requiring the processor's help.* This technique is commonly called DMA **(Director Memory Access)** and is usually used for transferring blocks of data between a fast peripheral and a pre-assigned area of main memory.

One interesting point is that the processor module *itself* has a bus priority which it can change under program control so that whereas it will normally have the lowest priority in the system (allowing other devices to use the bus when they want it) the processor can raise its priority to keep control of the bus when it is executing a time-critical part of a program.

Because the processor module can be made relinquish the bus, *multi*-processor systems are possible and the busses of two or more systems can be joined via 'Bus Switches' for high speed communication *between* systems.

Addressing

All memory locations and registers which can have access to the bus are assigned individual addresses from a common address space. The PDP-11 does not distinguish between registers, I/O controllers or memory locations; thus one can use a virtually unlimited number of I/O devices. (The top 4K of address space is usually reserved for peripherals, but this is a convention adopted for convenience and is not essential). This technique is known these days as 'Memory Mapped I/O'.

Although the PDP-11 is basically a 16 bit machine, addresses refer to half-words, or 8-bit bytes. This means that the basic PDP-11 with 16 address lines can deal with a maximum of 64k bytes of memory and registers. This is a relatively small amount of memory by modern standards (although 64k bytes would only have been found on a large mainframe at the time the PDP-11 was designed) and while the more powerful PDP-11's incorporate a memory management scheme to increase the usable address space to 256 bytes, it is probably the most serious limitation of the PDP-11 series.

The Bus, Again

One of the advantages of a bus oriented system like the PDP-11 is that because it gives rise to a *strict definition of the connections to a particular module* (or should do — S100 bus equipment designers please note!) then new modules can be designed as the need arises to work with or upgrade an existing system. To this end the designers of the original PDP-11 obviously spent a lot of time and effort in defining their bus and ensuring that it would still work reliably even in a large system. The resulting PDP-11 bus not only caters for 'backplane' connections (between boards in a chassis) but can also be implemented as a flexible cable linking different chasses — in a large PDP-11 system the bus can run for tens of feet between different units and even between different equipment cabinets. The large peripherals, such as disc or tape drives, have their controllers in-built, so the whole peripheral becomes a single physical module which is connected to the processor via the standard bus cable. Multiple peripherals are simply plugged into the bus in 'daisy chain' fashion.

Unfortunately the original version of the PDP-11 bus proved to be a bit too expensive and cumbersome for use with the new LSI-11 series of processor, so these have been designed around a simplified version of the bus. Nevertheless, the LSI-11 bus still retains all of the essential features of the original.

One further feature of the PDP-11 bus worthy of note is that it is asynchronous. The bus controller is told *by the modules* in use when the data transfer is complete. This means that *differing speed* memories can be mixed in a system with each used at its maximum speed. If a module doesn't respond in a reasonable time (usually because an address has been specified for which there is no corresponding memory or physical device on the system) then the controller will inform the processor of the problem so that an error routine can be started.

Peripheral Control

Two types of information are usually associated with peripheral devices:

- Control and Status signals which affect the operation of the peripheral. e.g. 'Step Paper Tape Reader' (a *control* signal) and 'New Data Received' (a *status* signal).
- The actual data which is being transferred.

To handle these two types of information PDP-11 peripheral controllers contain Control & Status register(s) and Data register(s) capable of connection to the system bus. The exact number and use of these registers will depend upon the particular peripheral.

As mentioned previously, specific addresses are allocated to these registers, and they appear to the processor as normal memory locations. Thus to send data via a peripheral the program merely contains a 'move' instruction which will cause the data to be placed in the peripheral controller Data register. All instructions can therefore be used with peripheral controller registers (in fact there are no special I/O commands).

Interrupts

Instead of seizing control of the system bus when it has information for transfer, a peripheral controller may instead interrupt the processor, causing it to pause in its normal operation while it attends to the peripheral. This approach is obviously slower than DMA, since it relies on the processor executing a special program to transfer data to or from the peripheral, but it requires less logic in the peripheral controller and is more flexible in that the operation is controlled by software rather than by hardware.

As with DMA, priority levels are assigned to the peripherals and to the processor (which can change its own priority), and the program can also enable or disable the interrupting ability of each peripheral.

When the processor is interrupted it has to determine *which* peripheral caused the interrupt. The PDP-11 does this by using a system of **'Vectored Interrupts'.** When an interrupt occurs the peripheral controller responsible gives a particular address to the processor, which then reads from that address the starting address of the appropriate interrupt handler routine. This system gives flexibility to suit different system configurations, and also allows the processor to respond quickly to an interrupt.

Nested interrupts are possible; a high priority peripheral can interrupt the execution of an interrupt handler routine for a lower priority peripheral. Automatic interrupts are also generated for power-up and power-down conditions and in the event of certain system malfunctions; such as the use of a non-existent address as mentioned previously.

Modules & Peripherals

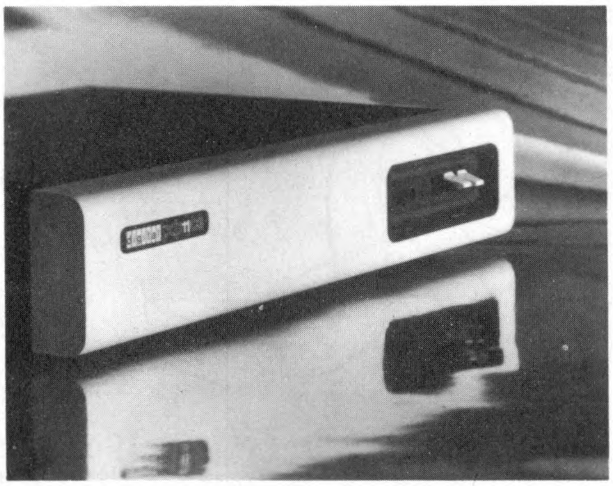
The range of modules which can be connected to the PDP-11 bus is impressive, especially as other companies have produced specialist 'PDP-11 Bus Compatible' modules for applications such as fast Fourier transforms.

As well as the ubiquitous TTY, DEC terminals include the popular LA36 30 char/sec 'Decwriter II' matrix printer/keyboard, VDU's, monochrome and colour graphic displays and line printers to produce those bulky reports so loved by software system designers.

Many types of memory module are available; ROM, PROM, core and semiconductor RAM are available in a range of sizes and speeds. Most impressive is a 64k byte MOS memory just released for the LSI-11 series; it is contained on a single 5" x 8½" board. Floppies, cartridge, fixed and moving head discs are standard options, as are paper, cassette and industry standard magnetic tape drives. For the network nut, there are a variety of data communications interface modules, from an acoustic coupler to a 16 lines multiplexer.

As well as these 'conventional' peripherals, DEC also produce numerous small devices, usually on a single board, which can be plugged into the bus to tailor the system to your requirements. Examples include a real time clock, a communications arithmetic card, and a remotely controllable power switch and bootstrap module which allows you to start up a system from a remote location. DMA or program controlled 'Interface Foundation' modules form the basis for constructing your own PDP-11 bus-compatible device.

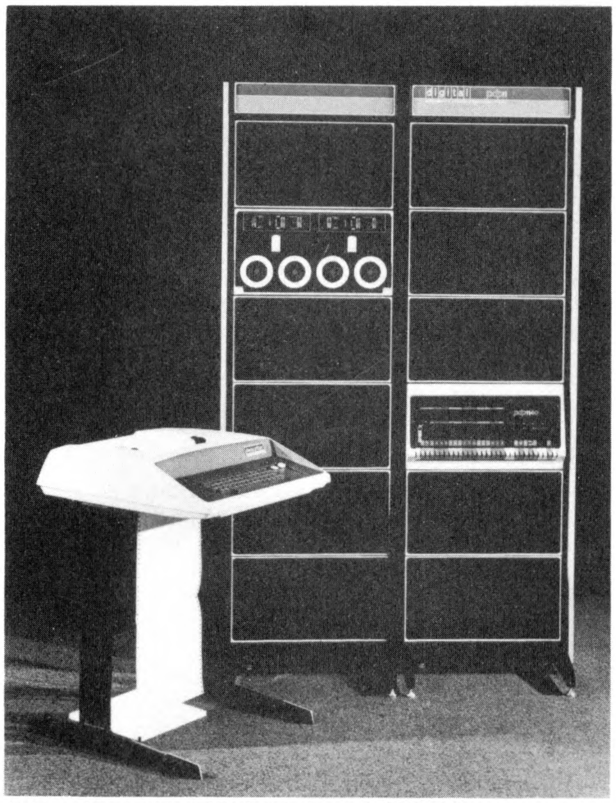
Users of the larger machines (PDP-11/40 and 11/45) may fit a separate Floating Point Processor,



which operates in parallel with the main processor to provide fast calculations to 17 decimal digits accuracy using 64 bit accumulators, while the smaller PDP's (/05, /10 & /20) can be fitted with an Extended Arithmetic Element which — connected to the bus and programmed as a peripheral — does fast 32 bit integer arithmetic. A similar 32 bit fast arithmetic capability is provided on the LSI-11 processor by merely plugging in another chip.

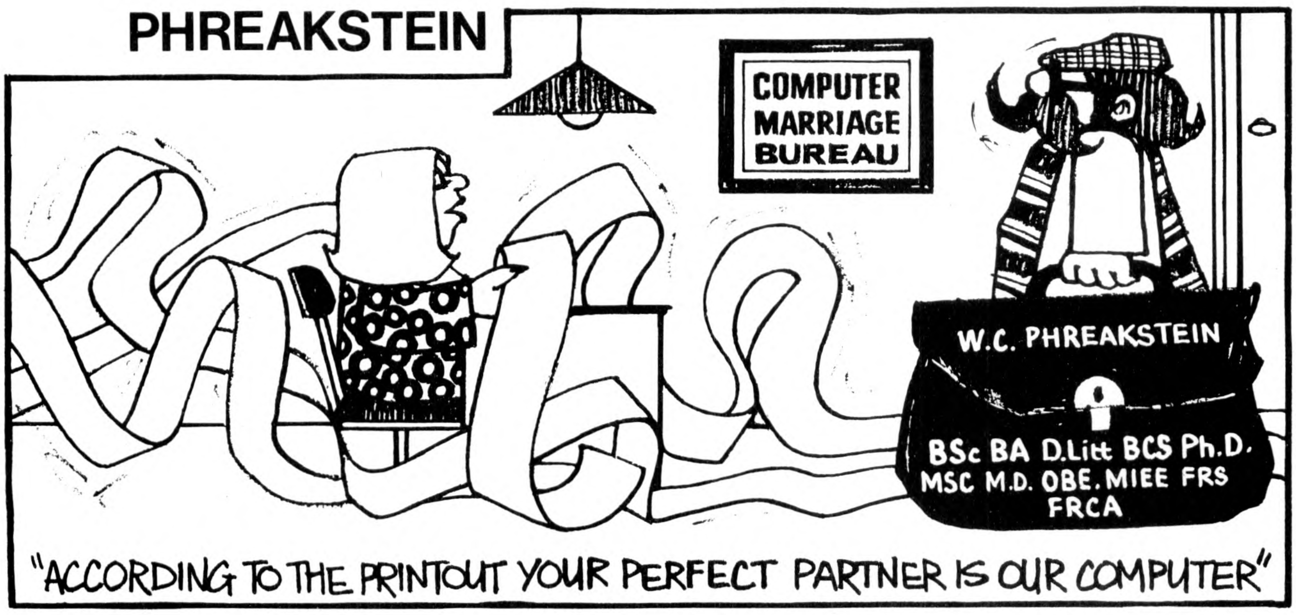
So Far So Good
In the preceding sections I have tried to give an overview of the general hardware features of PDP-11 systems, and the impression that emerges is of a complete family of compatible modules which can be configured to meet a particular user's needs by simply plugging them into the system bus. I have also tried to show how all devices which can communicate with the bus are treated as parts of a single address space, and how communication between the devices via the bus can either be under control of the processor ('programmed' data transfers) or, when data has to be transferred quickly, the module wanting to transfer data can itself temporarily take over the bus (DMA transfer).

PCW The PDP-11 inspires lyricism in its devotees, and in a future issue Mike Lord will go on to examine the architecture of the processor itself, as well as its instruction set. *PCW*


A dual cabinet version of Digital Equipment's PDP-11/40 computer. One of Digital's famous PDP-11 family of 16-bit computers.

PDP-11 & LSI-11 Processors

PDP-11/05	Small, relatively low speed processor designed for OEM use. 3.1uS register — register move time.
PDP-11/10	Similar to PDP-11/05 but intended for end-users, chiefly engineering laboratories and small educational systems.
PDP-11/20	The original PDP-11; still widely used as it is built in a larger box and can hence take more options than the /05 & /10. 2.3uS register — register move time.
PDP-11/40	The cheaper large processor, can use up to 256K bytes of memory. 0.9uS register — register move time.
PDP-11/45	The most powerful PDP-11, up to 256K bytes of memory. 0.3uS register — register move time.
LSI-11	Single board (8.5" x 10") processor built using LSI circuits with 8K bytes of RAM on the same board. 3.5uS register — register move time.
LSI-11/2	Similar to LSI-11 but without memory. 8.5" x 5" board.
PDP-11/03	Boxed version of LSI-11 with power supply & serial interface. A modern alternative to the PDD-11/05 & /10.


PHREAKSTEIN
COMPUTER MARRIAGE BUREAU
W.C. PHREAKSTEIN
BSc BA D.Litt BCS Ph.D. MSC M.D. OBE. MIEE FRS FRCA
"ACCORDING TO THE PRINTOUT YOUR PERFECT PARTNER IS OUR COMPUTER"

PROGRAMMING A PERSONAL APPROACH

Stephen Holden

Learning to write programs is, in a large measure, finding out that you are fallible just like everyone else. Since computers are very 'literal minded,' they will cheerfully do exactly what you tell them to do — even when you give them the wrong instructions.

The result is that when you get the program wrong, the computer gets the answers wrong. Sadly we are not yet at the stage where our computers can point out the mistakes in our programs. So any programming errors (bugs) we make must be discovered and put right (debugged). Sherlock Holmes would have been excellent at debugging, because the major requirement is the ability to think in a clear and logical way, freeing yourself of the preconceptions formed during the initial programming.

The best way to develop programs is to avoid writing bugs into them from the start, but this is far easier said than done. We can, however, adopt a defensive strategy with twofold objectives:

1. Make the bugs easy to find.
2. Make the bugs easy to remove.

The first is largely achieved by writing your programs as though you expected them to go wrong. Remembering Murphy's first law*, this is bound to happen anyway. So put in statements which print out the values of important variables *during* testing, and include checks that expected conditions have in fact occurred.

The second objective involves programming in such a way that changes to the logic or output of the program can be easily implemented. Make it easier to understand the operation of the program by including comments, and avoid clever programming tricks. I have seen programs where the logic, while very compactly represented, was so difficult to follow that the only way to make the program perform a slightly different function was to re-write large sections of the code.

You should also remember, during the development of a program, that the computer's time is far cheaper than yours. Many programmers are tempted into 'bit twiddling'' in the expectation of gains in execution speed which are unmeasurable. Get it right *first,* then speed it up if it doesn't run fast enough.

Since thousands of volumes have been written on the subject of programming, a single article will never be a definitive reference work. Instead, I invite you to look over my shoulder during the design and development of a small (and possibly useful) program. Hopefully, this will point you in the right direction and give you a start along the way.

*If a thing can go wrong, it will.

Define Your Objectives

Before ever approaching the keyboard with the gleam of battle in your eye, you must have a clear idea of what you expect the program to do. It is best if you have this in writing, and clearest if you specify the items of information the computer will have to be given when running the program, and the information which you expect in return.

These items do not have to be written in any special way, but should be used to clarify your mind about the task of the program. Already you have a yardstick by which to measure the program's performance.

Let us now consider a specific problem, that of reconciling your monthly bank statement with your own records. The idea is to enter your credits and debits from the statement, and then the credits and debits from your records. You will obviously want to know about any items for which no matching transaction exists.

This gives us the following specification of inputs and outputs:

Input:	1. Statement credits and debits.
	2 Records of credits and debits.
Output:	1. Unmatched statement transactions.
	2. Unmatched record transactions.

Professional systems analysts show in line-by-line detail the sequence of queries and outputs. This is perhaps taking things a bit too far for a reasonably simple program which is not part of a larger software system, though.

Design Your Data Structures

Now you know what data are going to be sent into and out of the computer, you will have to decide how best to represent these data in the computer memory. There is in reality no absolute best way of doing this, and you will often find at the beginning of your programming career that you have chosen a representation which leads to unnecessary programming complications. The only remedy at such a point is to go back and rework the definitions, rewriting the parts of the program which are affected by the change. In doing so, you can be confident that you are breeding better programming habits, and learning an approach which will pay for itself over and over again by saving time in the future.

For our sample problem, we will be trying to match the amounts in two lists of numbers (the statement transactions and the record transactions). A natural way of representing lists in a BASIC program is as the elements of an *array*. Such a representation allows you to get hold of the elements by using *subscripts,* which are expressions in brackets after the array name, specifying *which* element is to be used. For example, A(5) is the fifth member of the array A.

We will also need some indication of *which* transactions have been *matched* by the program, and we can use arrays for this purpose too by arranging that if a particular transaction in the *n*th element of its array has been matched then the *n*th element of *another* array will hold the value *one.* Otherwise, we will arrange for that element to hold zero.

Last of all we need some indication of how many items have been put into each of the lists, and we will do this by holding the subscript of the last element in use in a variable. Since one picture is supposed to be worth a thousand words, I will save a page of magazine by drawing Figure 1 to your attention.

Here the T1 array is used to hold the transaction amounts from the statement, and the M1 array holds the marker to show whether they are matched. The variable N1 contains the number of entries in use in these lists. A similar situation holds for the record items, which use arrays T2 and M2 and the variable N2.

You can see that there are five statement items, of which four have been matched, and six record items, of which four have been matched. You may also notice that all names associated with the statement end in the digit 1, whereas those to do with the records end in the digit 2. This may not seem important, but it helps to ensure a separation of the two sets of data in your mind.

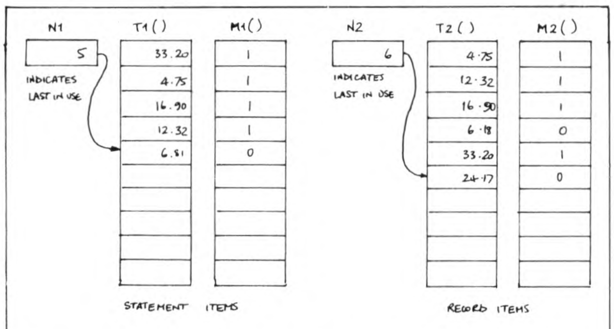


Figure 1. Data structures for the bank statement checker. The arrays have a maximum element number of ten because they were not set up by a DIM statement. There are five statement items and six record items.

How Would You Do the Job?

We have finally reached the point where we can start thinking about the sequence of operations the computer must perform. Just as an artist will begin his picture in outline, with broad brush-strokes, you should try to avoid getting bogged down in detail too early. Imagine how you would tackle the problem you are going to ask your computer to solve.

Better still assume that you are employing a complete moron with few brains and less imagination, but an unrivalled aptitude for figures and logic. (Don't laugh at this idea, because that's exactly what you are doing when you use a computer). How would you instruct him to give him the best possible chance of success?

One fairly concise statement of the task is: 'Read the amount on each of my *records,* and scan down the *statement* looking for an entry of the same amount. If you find one, tick both the line on the statement and the record, and in any case proceed to the next record. When you have dealt with all records, make a list of the *statement* lines you have not ticked, and another one of the *records* you have not ticked.'

Since your computer won't be able to understand these instructions, you must now translate this statement of the task into BASIC. You will notice, though, that much of what we have so far done is quite *independent* of the language in which the program is to be written.

You may also notice that under a certain perverse combination of circumstances the above instructions would fail to have the moron do the right thing. This error will be attended to later, but if you can see it now you can go to the top of the class. If not, you will appreciate the value of the debugging statements we are going to write into the program.

How to Instruct Your Moron

Since we are going to tell the computer to scan down the transactions on the statement, we must obviously give it instructions to read these into its memory before starting the program proper. This kind of initialisation code is often required, so even if you don't write it first (perhaps because you didn't realise it was necessary) you should leave enough free line numbers at the beginning to fit in later.

The initialisation of the statement is handled in lines 100 to 190 in the program listing, and should be easy enough to understand if you've had any exposure to BASIC. For those who haven't, line 110 sets the number of statement items to zero. Line 120 tells the user what to type (*never* assume that he'll know — if it takes a long time to tell him, ask him if he wants instructions). Line 130 reads a transaction amount from the system keyboard, and line 140 skips out of the section if the user says he's finished typing. An input of some 'sentinel value', in this case zero, is a commonly-used technique for getting out of an input loop. Since we must have read a real transaction if we get to line 150, the number of statement items is increased by one. Line 160 stores the transaction amount away in the appropriate array element, and line 170 sets the transaction flag to 'unmatched.' Lastly, line 180 sends the computer back to the input statement to enter another amount. Line 190 is a comment because that way you can put in and take out statements when modifying the program knowing that you'll never try to jump to a statement that doesn't exist.

To check that the statement has been correctly set up, we include at lines 200-230 some program to print the items out. These are debugging lines, which can be removed when we are satisfied that lines 100 to 190 work correctly.

The next piece of code to write is the bit to read in the record items. Since all of the statement will by this time be in the memory, we check these items immediately they are read in (lines 240-400). Lines 240 to 300 are closely parallel to the lines for the same job on the statement, and require no further comment. The next three lines are a programmed loop to look at all the statement transactions. Line 310 establishes the limits of the loop, saying in effect 'start with I as one and keep adding one to it until it becomes greater than the number of statement transactions.' Line 320 causes the computer to exit early from the loop if it finds that the transaction amount matches, to a piece of program which marks both the record and statement transactions as matched.

Line 330 says that we must go back and try another value for I if we haven't yet tried all possible values given in the FOR statement. If we have, of course, the conclusion is that we have no matching statement transaction. Consequently, we mark this record as unmatched, and go back for another.

This section also includes debugging statements, at lines 340 and 370, to inform us of the computer's actions in either case. This is in line with our philosophy of expecting trouble and being suspicious, rather than pleased, when it doesn't appear.

Lastly, the program scans up each of the marker arrays and prints out the transactions which the program hasn't marked as matched. This code (lines 410-999) is unexceptional, bar the fact that there is a special case when all transactions are matched to account for why the program printed a heading with no transactions. This is largely a matter of taste, but I personally don't like my programs to act in a stupid fashion. Again there are debugging statements at lines 445 and 545 to check that the computer has done the right thing.

Testing

For realism, the preceding part of this article was written before the computer was programmed. In entering the program, some typing mistakes were made which caused pathological behaviour soon revealed by the debugging statements (one of which had itself been incorrectly typed).

The following test cases were tried and found to work.

1. No statement items.
2. No record items.
3. Some of each, but none matching.
4. Some of each, with *all* matching.
5. Some of each, some matched, others not.

The deliberate mistake (see output 1) was 'discovered' by entering two statement items of the same amount, and matching them both. Line 385 told me that they had not both been matched, and the reason is clear. At line 320, when checking that an item was matched, no attempt was made to see if it had been previously matched. Line 320 was changed to

320 IF T = T1(I) AND M1(I) = 0 THEN 370

and the program was adjudged to be working.

I have, however, kept a copy with the debugging prints in it, because somewhere at the back of my mind I keep hearing Murphy's law . . .

```
100   REM INITIALISE STATEMENT ITEMS
110   LET N1=0
120   PRINT "STATEMENT INPUT - END WITH ZERO"
130   INPUT T
140   IF T=0 THEN 190
150   LET N1=N1+1
160   LET T1(N1)=T
170   LET M1(N1)=0
180   GOTO 130
190   REM END OF STATEMENT INPUT
200   PRINT N1;" STATEMENT ITEMS"
210   FOR I=1 TO N1
220   PRINT T1(I), M1(I)
230   NEXT I
240   REM READ AND MATCH RECORD ITEMS
250   PRINT "RECORD INPUT - END WITH ZERO"
260   LET N2=0
270   INPUT T
280   IF T=0 THEN 400
290   LET N2=N2+1
300   LET T2(N2)=T
310   FOR I=1 TO N1
320   IF T=T1(I) THEN 370
330   NEXT I
340   PRINT "ITEM DID NOT MATCH"
350   LET M2(N2)=0
360   GOTO 270
370   REM ITEMS T1(I), T2(N2) ARE MATCHED
375   LET M1(I)=1
380   LET M2(N2)=1
385   PRINT "MATCHED STMT ITEM ";I
390   GOTO 270
400   REM END OF RECORD INPUT
410   REM PRINT STATEMENT ITEMS
420   PRINT "UNMATCHED STATEMENT TRANSACTIONS"
430   LET C=0
440   FOR I=1 TO N1
445   PRINT T1(I), M1(I)
450   IF M1(I)=1 THEN 480
460   PRINT "ITEM ";I,T1(I)
470   LET C=C+1
480   NEXT I
490   IF C<>0 THEN 510
500   PRINT "NONE"
510   REM PRINT RECORD ITEMS
520   PRINT "UNMATCHED RECORD TRANSACTIONS"
530   LET C=0
540   FOR I=1 TO N2
545   PRINT T2(I), M2(I)
550   IF M2(I)=1 THEN 580
560   PRINT "ITEM ";I, T2(I)
570   LET C=C+1
580   NEXT I
590   IF C<>0 THEN 610
600   PRINT "NONE"
610   PRINT
620   PRINT "*** FINISHED ***"
999   END
```

```
RUNNH
STATEMENT INPUT - END WITH ZERO
? 12.32
? 25.67
? 10.00
? 15.32
? 10.00
? 0
 5  STATEMENT ITEMS
 12.32         0
 45.67         0
 10            0
 15.32         0
 10            0
RECORD INPUT - END WITH ZERO
? 12.32
MATCHED STMT ITEM  1
? 10.00
MATCHED STMT ITEM  3
? 15.32
MATCHED STMT ITEM  4
? 45.67   .
MATCHED STMT ITEM  2
? 10.00
MATCHED STMT ITEM  3
? 0
UNMATCHED STATEMENT TRANSACTIONS
 12.32         1
 45.67         1
 10            1
 15.32         1
 10            0
ITEM  5        10
UNMATCHED RECORD TRANSACTIONS
 12.32         1
 10            1
 15.32         1
 45.67         1
 10            1
NONE

*** FINISHED ***

-----------------------------------------------------------------
```

Output 1. Statement item 3 was marked twice, and item 5 not at all.

```
  RUNNH
  STATEMENT INPUT - END WITH ZERO
  ? 33.43
  ? 12.76
  ? -9.03
  ? 45.86
  ?.5.00
  ? 0
*  5  STATEMENT ITEMS
*  33.42         0
*  12.76         0
*  9.03          0
*  45.86         0
*  5             0
  RECORD INPUT - END WITH ZERO
  ? 5.00
  MATCHED STMT ITEM  5
  ? -12.45
  ITEM DID NOT MATCH
  ? 45.86
  MATCHED STMT ITEM  5
  ?.33.43
  MATCHED STMT ITEM  1
  ? 33.43
  MATCHED STMT ITEM  1
  ? 0
  UNMATCHED STATEMENT TRANSACTIONS
*  33.43         1
*  12.76         0
  ITEM 2        12.76
* -9.03          0
  ITEM 3        -9.03
*  45.86         1
*  5             1
  UNMATCHED RECORD TRANSACTIONS
*  5             1
* -12.45         0
  ITEM 2        -12.45
*  45.86         1
*  33.43         1
*  33.43         1

  *** FINISHED ***

--------------------------------------------------------------------
```

Output 2. A typical debugging run. Lines marked with an asterisk would not appear when debugging statements were removed. Note that the deliberate mistake also causes an error here.

```
RUNNH
STATEMENT INPUT - END WITH ZERO
? 22.45
? 45.65
? 12.45
? 12.45
? 0
RECORD INPUT - END WITH ZERO
? 12.45
? 12.45
? 22.45
? 16.89
? 0
UNMATCHED STATEMENT TRANSACTIONS
ITEM  2        45.65
UNMATCHED RECORD TRANSACTIONS
ITEM  4        16.89

*** FINISHED ***

------------------------------------------------------------------
```

Output 3. A final run of the corrected prrogram in its production form. Note that the "deliberate mistake" causes no error here although given every opportunity.

Richard R. Waller, *CAP MicroSoft Ltd.*

Do-It-Yourself MICROS FOR COMPANIES

This is addressed to the 90% of current microcomputer buyers who are paying for equipment with company money. It takes the form of a straight question. Are you man enough to make a case for real money and put up a plan to your company to do the job properly? if you can see your way to asking for up to £25,000 then in six months you could have a company microcomputer centre of knowledge which could be of real service to dozens of potential company micro users.

More than this, your leadership could set a path for a communication network within the company; an automated office if you like; and put you months or even years ahead of your competition in handling your admin tasks.

If your company doesn't do this, then micros will still come. They will come with each new computer terminal, within each new computer peripheral and in items of office machinery. Word Processing is a natural for micros, and building services will increasingly find micros in air conditioning, security gates and lifts. Small Business Machines are already creeping into ordinary offices and stores. The factory is still fair game for micro control.

All this will happen. And no one will ask the Board of Directors, or the DP Manager. Why should they; the price of any one unit is much less than the limit of discretionary spending of each individual manager concerned.

I think you can do better for your company. All it needs is initiative; the costs are not great compared with the current DP budget, much less the cost of the total paper-work handling for the organisation; it has been estimated that DP tends to cost 2% of turnover, and paper handling 40% of turnover.

The micro centres we have helped set up tend to have 2-4 people full time to start with, plus part-time services of a contract electronic wireman hired by the hour to do the difficult bits. A key man is the leader, requiring management/analysis/salesmanship skills. A programmer or two will do the work, and there is normally room for a junior/secretary/general dogsbody to keep the paperwork in order, help users, demonstrate (and make the coffee). After a few months there will be major programs to write, which can be sub-contracted to the DP department who have skilled resources for such work.

It is necessary to make a start somewhere. The first use of micros in a company starting from scratch should: —

a) Be very simple; complexity can come later.
b) Be non time critical; in case it has teething problems.
c) Be supplied free of charge to the user.
d) Be useful and visible.

It always seems to be difficult to come up with ideas for this first project. May I suggest: Index of Names and Addresses, Mailing Lists, Telephone Directory, Staff lists, Weekly diary of appointments, Database of staff skills, Index of available micros and small business systems, library magazine circulation records, inventory of furniture or office machines . . . all fairly simple but useful and demonstratable.

Real business applications tend to be big. Sales Ledger is typically 30 programs, Purchase and Nominal Ledgers perhaps 20 each, Stock Control is 25. Each program from our experience tends to be between 200 and 700 MicroCOBOL statements and thus each of these packages represents a very major investment, certainly not much less than a package produced for a main frame computer. As an estimate for budgeting purposes, each program takes about one week to design, write, test, and document, and costs about £1.50 per statement.

Replication is the only solution; a major package must be used by as many users as possible. Small Business System suppliers do this and some of their supplied software is very good. They are understandably not willing to modify the basic package unless you pay for the costs involved. A company package can often be specially designed and written — in which case spreading the cost of a stock control package over, say, 60 depots can bring the software cost down to less than £1000 per site. This would run on a computer costing £5000 and thus the bill to any individual site is quite reasonable.

Three others things can be done to help users. There should be a growing library of utility programs to Sort, Merge, Copy, Print and generally massage data on files; we have 18 on our library. Then there will be a number of 'Executive Tools' like a desk calculator facility, Discounted Cash Flow, a diary of appointments, etc. These will arrive almost without effort, but could usefully be disseminated from the MicroCentre. And lastly there should be help with repairs; units that do go wrong should be replaced with spares held in stock and either fixed on site if the skills are available, or more likely taken back to the supplier for repair. If there are several micros on one site some contract maintenance is possible. A smooth service to help the user in distress can do a lot to create satisfied micro centre customers.

All this may have left you with the idea that standardisation is the only answer. Many organisations have gone this way; they have bought a number of Intel 8080's or whatever, with their appropriate prototyping systems and it is going to be quire difficult for them to move on to the next generation even within the Intel range. Or they have programmed in BASIC and are living with the considerable difference between the different interpreters, on the several different micros they have acquired.

In CAP we were determined not to get into either of these boxes.

- The CAP MicroCOBOL Language is supported by Interpreters on five micros and minis so far and more will come. When a new micro is announced we will write a new interpreter and slide the whole of our software support across onto the new system.
- We have also gone strongly onto floppy disk files and our universal file support provides Relative Sequential and Indexed Sequential access methods for all our programs.
- Program development on the host LSI-11 micro (or PDP-11 mini if you insist) provides us with an excellent source program editor and libraries, and runs the Cross translator from MicroCOBOL into the Intermediate Commercial Code for use by any of the target micros. We can also handle Assembler modules where necessary and burn our own PROM's for device control and engineering use.
- Business Operating System on each of the target micros provides job control, a range of utilities, the seven testing and debugging commands, and of course the Interpreter for the specific micro concerned.

This is what we are recommend our clients to go for. The costs of the whole package are shown in the Budget. If you have access to a PDP-11 already then the cost is even less. The advantages of micros are clear to you. Going this way, your company can build on our experience and obtain these benefits without further delay. It is really up to you.

OBJECTIVES OF THE MICRO CENTRE

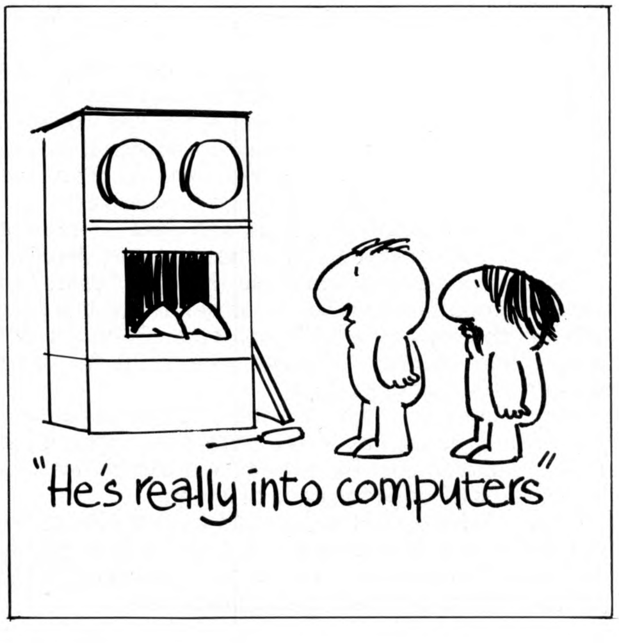
1. To give leadership in the company in the use of microcomputers.
2. To make the most effective use of all microcomputer equipment within the company.
3. To promote the common use of application software and utility programs and the transfer of software between users.
4. To ensure the compatibility of data for interchange between users.
5. To provide expertise on project control and other services to users in cases where these are more economically provided centrally.

ACTION PLAN FOR THE MICRO CENTRE

1. Be aware of current state of the art (magazines, manufacturers and suppliers literature etc)
2. Find out what equipment is available (price lists, surveys etc)
3. Learn to program (some or all of:— Machine Code; Assembler; BASIC, PL/M, MicroCOBOL etc)
4. Know about Prototyping Systems and Bureau Simulators (Brochures, bureau literature)
5. Install one simple microcomputer project (maybe two months work)
6. Be aware of every existing and planned use of micros within the company
7. Install one or two more simple micro projects using the experience from No 5 above (perhaps two more months work)
8. Offer services to all users as required to help them install their own micros (analysis, programming, maintenance, back-up, electronic wiring, project control)
9. Set up a 'swop-shop' of users redundant equipment so that users may upgrade their hardware when necessary at minimum expense.
10. Provide a forum for the exchange and common use of all available software between users.
11. Assist the transfer of data between users by suggesting standard data formats etc.

BUDGET ITEMS FOR THE MICROCENTRE

A.	**Program Development System**	
	MicroAde software for MicroCOBOL and Assembler programs with Business Operating System and Interpreter for Intel 8080	£9000
	Universal File System for Midos 500 (PerSci 277 and 1070 floppies)	£1500
B.	**Host Hardware**	
	DEC PDP-11/VO3 PDP-11, RXV-11, RT-11, LA36, 32K words)	£7000
	VDU	£600
	DEC DLV-11 (link to microcomputer)	£200
C.	**Target Hardware**	
	Intel System 80/10 (Intel 8080 microcomputer)	£1100
	Intel SBC-032 (32K byte memory board)	£1100
	Penny and Giles Midos 500 (PerSci floppy disk system)	£1400
	VDU	£600
	Printer	£1200
		£23700



JOHANN SEBASTIAN BYTE

Michael Inggs

One is becoming so accustomed to the power of computers these days, it perhaps not surprising to most to hear how they can fairly simply be set to work to play music. This article will attempt to show some of the ways the most rudimentary microprocessor system can be made to play simple tunes which have programmed from sheet music.

Because of the variety of machines available at present, there is no point in getting involved in specifics, so most of what follows will be presented in flow chart form for suitable implementation on a particular system. Some code will be given in BASIC, since this is often available and is sufficiently standard.

Music; what it means

(What follows is very simplified, so incipient Mozarts should be prepared to do some extra reading.)

Consider the simple scale presented in fig. 1. We see two promising characteristics; music is digital in time and frequency! The rather baffling collection of notes may be plotted as a frequency versus time graph. The *position* on the staves gives the frequency or note to be played, and the *shape* of the note gives its value or length.

So, with practice, reading music becomes quite simple. Getting one's fingers to pluck the right string or whatever is another story.

Before leaving the subject of music, a few more details are necessary. In fig. 2 we show the positions and names of the 8 notes making up an octave based on A. The table shows the actual frequencies according to the most commonly used scale, the "Equally Tempered Scale". Notice that High A (an octave above A) has its own extra line called a "ledger line". These lines may be added above or below the stave (set of five lines) shown. The next note above High A is B again, then C and so on.

We will not mention sharps and flats — a tame musician friend can soon fill in the necessary details.

Some more details about the shape or "value" of notes is important and fig. 3 has details. Notice how the length drops by a half each time. Putting a dot after a note means, "hold the note for its full value plus another half of its value."

What does a "beat" mean? It depends on the tune. Slow music has fewer beats per minute than, say, a Scottish Jig. By instinct and experience one soon learns to judge the pace of a tune.

Again, other complexities such as time signatures and bars have been skipped. A musician can supply the details which are not really necessary for the simple-minded approach being taken here.

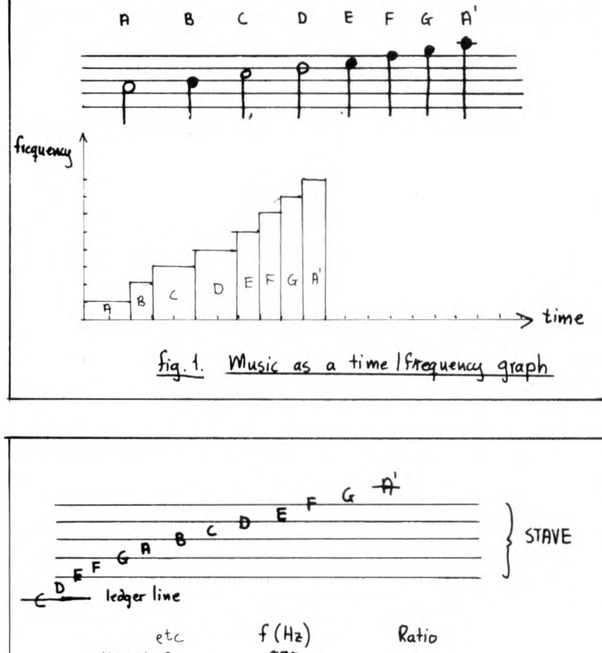
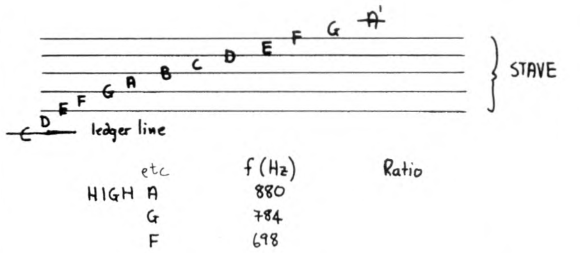


fig. 1. Music as a time/frequency graph



		f (Hz)	Ratio
	etc		
HIGH	A	880	
	G	784	
	F	698	
	E	659	
	D	587	$1,0594 = \sqrt[12]{2}$
	C	523	
	B	494	
	A	440	
LOW	G		
	etc		

fig. 2. The Equally Tempered Scale

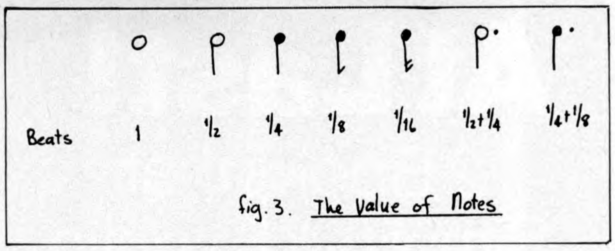


fig. 3. The Value of Notes

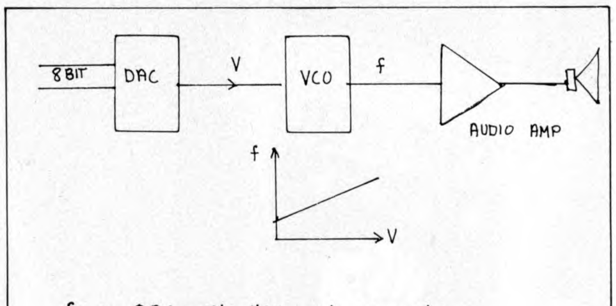


fig. 4. DAC with VCO note generator

Making FRED* vocal (*FRED . . . Ridiculous Electronic Device!)

We now investigate ways of getting notes from the computer, other than 50 Hz or 2 MHz.

Conceptually the approach of fig. 4 is simplest. The 8 bit bus of a computer feeds an 8 bit digital to analogue converter (DAC). This in turn feeds a Voltage Controlled Oscillator (VCO) whose characteristics are shown as well. The Oscillator then feeds a suitable audio transducer. 8 bit computer words can now be converted to suitable audio tones.

This sophistication is not entirely necessary: all we need is one bit of the bus! If we choose the least significant bit (isb) and send alternate odd and even numbers down the bus, we would have alternate high and low values appearing on the lsb.

The flow chart of fig. 5a turns a processor into a square wave signal generator whose frequency is set by the clock frequency and the number of clock cycles required to implement the instructions shown.

Fig. 5b shows how to make the frequency variable by means of software. Calculation knowing the cycle time of the processor soon gives the values of N needed to give the notes of the scale. Access to a frequency counter makes life very simple.

Getting the Beat

We are able now to simulate the notes of the scale; to vary their length is straight forward. The Note generator of fig. 5b is just inserted into another

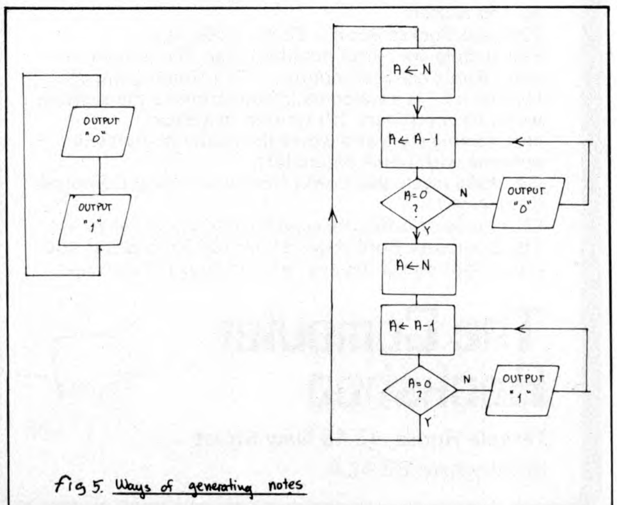


fig 5. Ways of generating notes

suitable loop with an initial value L setting the length for which the note is played, as in fig. 6.

This process of course does slightly alter the frequency of the notes but generally the note frequencies involved are low and the extra processing involved in generating the length is negligible. It is important that the rate at which B is counted down is set by the processor clock and not every cycle of the N loop, for example. This would make the note length dependent on the note being played, which is undesirable.

Again suitable calculation or "cut and try" methods soon give the correct L values to simulate the various note values found in music.

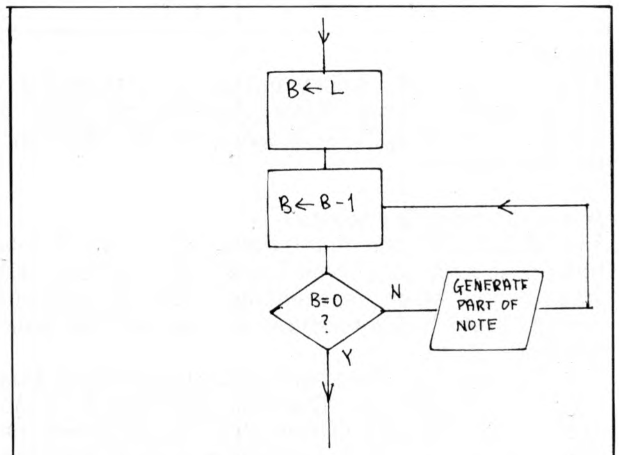


fig. 6. Varying the length of notes.

Interpreting Music

By this stage, most readers will probably have their processors wailing quietly away; some further tips may be found useful. First we will discuss methods of storing tunes in memory.

The most obvious method is to use alternate words of memory to store first a note and then its value. This is not as wasteful as it sounds, because most simple tunes have surprisingly few notes.

A more devious method is to split each word according to fig. 7. This restricts the range of notes and values available to 16 each. This is quite sufficient for most applications.

It is a tedious task entering music into the processor memory and this is where a BASIC or even ASSEMBLER interpreter comes in useful. It is simple to devise a MUSIC assembly language. An example is given in fig. 8. for an 8 note language.

Previous experimentation has given the correct values for N and L. Fig. 9 shows how these values are used in a simple BASIC MUSIC interpreter. This program converts the alphanumerics input into suitable N and L values; in this example they are then stored in a free part of memory, to be available for an OPCODE program to use and play back.

All sorts of sophistications are easily built in. Perhaps an essential one is a simple editor. There is nothing more frustrating than punching in note 158 to find that note 3 was accidently left out . . .

The author's version also caters for repeats ie. certain phrases of music have to be repeated before continuing with the rest of the tune. The symbol "R" is then inserted into the sequence of notes. On encountering an "R" the machine goes back to find the nearest "S" inserted into the notes and repeats from there. Similarly chorus may be catered for.

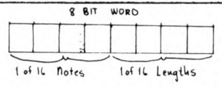


fig.7. Storing a note and its value in one word of memory.

Note	MUSIC representation	Value	MUSIC representation
A	A	𝅗𝅥	1
B	B	♩	2
C	C	♩.	3
D	D	♪	4
E	E	♪.	5
F	F	etc	
G	G		
A'	H		
etc			

fig. 8. Hypothetical MUSIC assembly language.

Storage
This is a very useful facility and saves the tedious job of re-entering tunes. Most systems will have a cassette tape dumping and loading facility which can be used directly.

A Syn-phoney Orchestra?
The simple sine or square wave output which has been described up to now is a far cry from any real instrument. Most instruments have a definite "timbre" which is a function of their rich harmonic content.

A simulation of these harmonics digitally would be horrendously difficult. The best approach is to go back to the ADC and VCO method The VCO may be made suitably complex; basically one could interface directly with an electronic organ or Synthesizer.

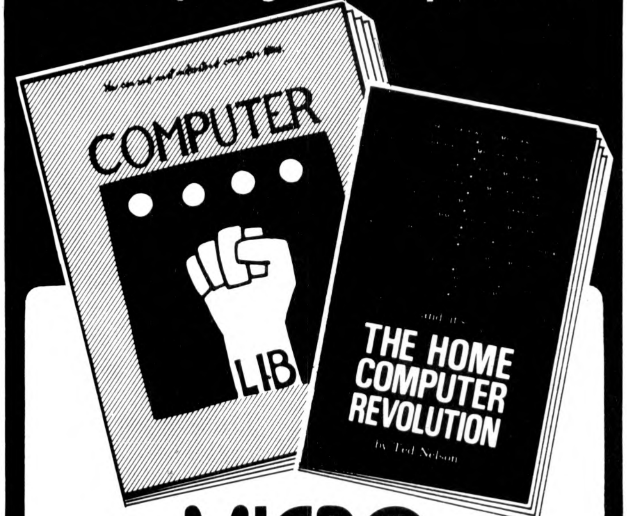
The permutations and combinations are large; hopefully this article will spark off some original ideas suitable for anything from a 256 Byte SCIMP to a 65K Z-80. But don't forget, any musician will tell you that the music is soulless — he will be quite right — FRED never makes any mistakes!

```
100 J = 48000
200 PRINT "NOTE? "
300 INPUT A$
400 PRINT "VALUE? "
500 INPUT L
600 IF A$ = "A" THEN N = 123
700 IF A$ = "B" THEN N = 127
800 IF AS = "C".......
...............
.............
1000 IF L = 1 THEN L1=5436
1100 IF L = 2 THEN Li=5458
1200 IF L = ...........
.............
..........
........
2000 POKE J,N
2100 POKE J + 1,L
2200 GOTO 200
```

fig. 9 BASIC version of MUSIC assembler.

At Last!

2 Books that Simplify & Explain

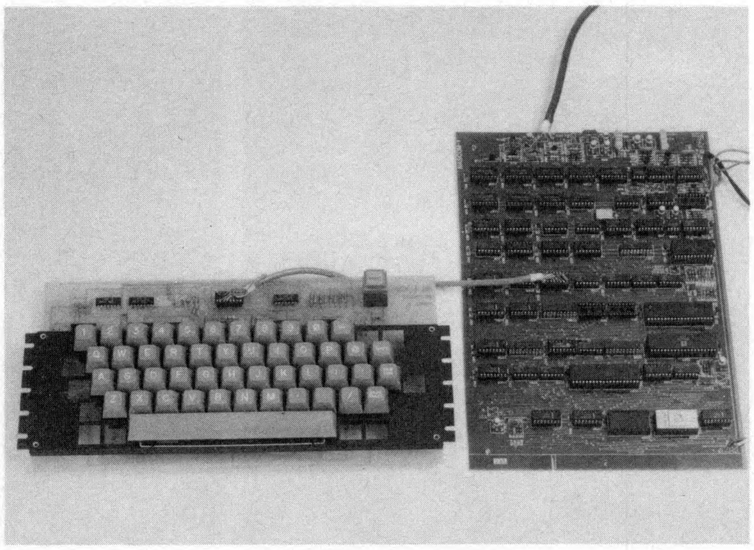


MICRO COMPUTERS & COMPUTING!

COMPUTER LIB & DREAM MACHINE
by Ted Nelson
127 pages, 10½" x 14" – £5.95 + 50p p.p.
This brilliant "duo" book, now in its 5th printing, outlines the past, the present and the future for micro-home computer and computing. "Computer Lib" which is one of the most popular books among students and computer hobbyists, explains simply and concisely why micro-computers are such fascinating wonderful machines and how they can better our lives.
The second half of the book "Dream Machine" is a journey of fantasy and imagination which explores new techniques that can be used on present day hardware.

THE HOME COMPUTER REVOLUTION
by Ted Nelson
224 page Pocket Book – £3.45 + 50p p.p.
Few people are better qualified than Ted Nelson to write about home computers. "The Home Computer Revolution" is a welcome introduction to the exciting world of computers. It's written in a clear, entertaining style and leaves the reader completely at-home with home computers.
Get these invaluable books from your Local Computer Store Now!
Or, in case of difficulty send cheque made out to – The Computer Bookshop. Allow 70p for p & p if you order both books together and 28 days for delivery.

The Computer Bookshop
Temple House, 43-48 New Street, Birmingham B2 4LA.



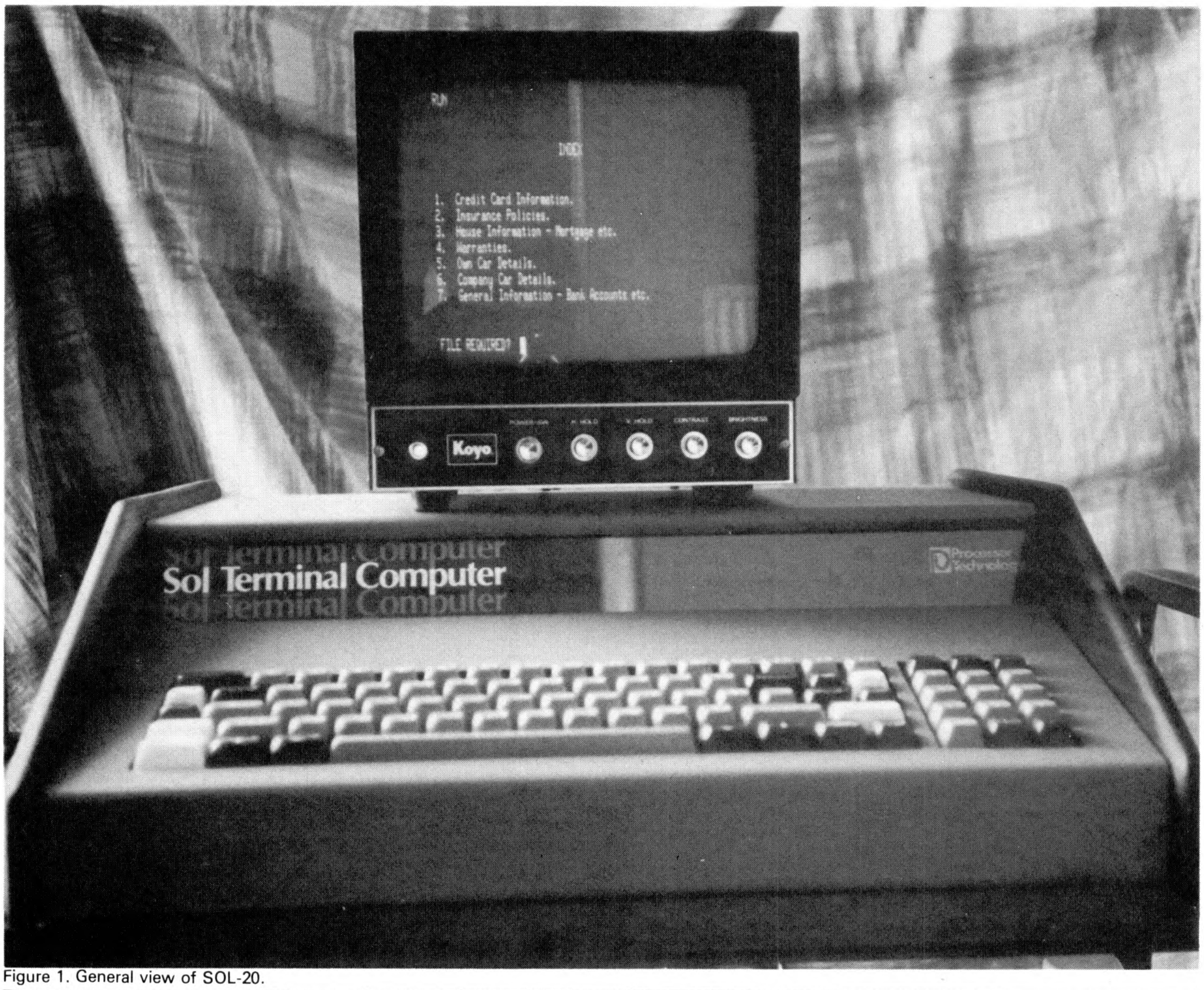
NASCOM 1
Z80 MICROCOMPUTER KIT

Includes interface for:
TV or Monitor — cassette,
dump — Teletype — spare,
PIO,
Expansion RAM-
BOARD up to 32K.

Potential:
High Level Language,
Mini Floppy Disk,
Excellent Z80
evaluation kit.
£197.50+VAT.

Write for details or send order to:
NASCO SALES LIMITED, Dept. PCW ,92 BROAD STREET,
CHESHAM, BUCKS. Tel: 02405 75151, Telex: 837571.

OWNER'S REPORT



```
RUN

                    INDEX

1.  Credit Card Information.
2.  Insurance Policies.
3.  House Information - Mortgage etc.
4.  Warranties.
5.  Own Car Details.
6.  Company Car Details.
7.  General Information - Bank Accounts etc.

FILE REQUIRED?
```

Figure 1. General view of SOL-20.

THE SOL-20 COMPUTER

Ken Wheeler

When I learned that I was to work in Los Angeles for about a year or so I was, of course, delighted. Not only because of the well known attractions of the Californian climate and the high standard of living, but also because I knew that the 'home computer' was already established there, with a variety of machines already available in computer stores and thriving clubs springing up all over the state. I am a mechanical engineer by profession but I have been an enthusiastic electronics hobbyist for many years. My interest in personal computers had been aroused during an earlier business visit to the USA in 1975 when I saw for the first time the early Altair and Imsai microcomputers. Although, when I went to California I was keen to get started in personal computing, events prevented me from getting a machine for quite some time. For that reason I was unable to acquire as much knowledge or equipment as I would have liked but I did eventually make a start and the machine I chose is the one illustrated in Fig. 1.

With all the well established names such as Imsai, Altair, SWTPC and others all anxious to relieve me of my hard earned dollars, why did I choose a SOL-20? Well, the first thing I did before going out to shop around was to write down a list of the things I thought were important to me:—

1. Whatever I decided to buy had to be available in kit form since I wanted the experience of putting it together myself. I felt that the knowledge gained in this way would stand me in good stead after my return to the UK and the manufacturer was no longer a convenient telephone call away.
2. I wanted a unit which would be as compact as possible, partly because of considerations of shipment back to England at the end of my contract in America and also because of space limitations in my home. My wife was already complaining that the

house was stretched almost to bursting point with hi-fi gear, test equipment and other electronics wizardry from my earlier enthusiasms.

3. I wanted a unit which, when assembly had been completed, would be 'up and running' and capable of being programmed in some form of BASIC. I had decided that all those laborious switches and LED's were not for me; neither would I be satisfied with a hexadecimal keypad and machine language programming.

4. I wanted to be sure that the manufacturer had a good range of software, extension boards and peripherals available with plenty of additional expansion in all areas planned for the future.

With these points in mind I set off on a tour of the greater Los Angeles area in search of computer stores and a chance to actually try out whatever units I could find. This turned out to be quite a task in itself because Los Angeles is spread over such a vast geographic area: from Hollywood in the north to Irvine in the south for example is a drive of over sixty miles. After looking at and operating most of the equipment which was currently available on the US market I decided that the SOL-20 was for me. In the first place it actually was available in the store, and in kit form. Many of the competitive products were only available to order and I already had experience of long delivery delays on other items which I had purchased in California. Also, several letters had appeared in magazines such as 'Kilobaud' mentioning delays as long as nine months for some items of home computer gear. In the second place, SOL-20 seemed to meet all the requirements which I had listed plus quite a few I hadn't even thought about.

Fig. 1 shows a general view of the completed computer and you can see that it is a very attractive self contained unit. It has, mounted in the front panel, a solid state keyboard which provides a full 70 key alphanumeric set plus a separate 15 key arithmetic pad. The arithmetic pad operates in parallel with the number keys on the main keyboard but is more convenient for entering large quantities of numeric data. This keyboard is undoubtedly one of the best I tested in my travels around L.A. and is a real pleasure to use.

SOL-20 is based on the Intel 8080 microprocessor and uses the Altair/Imsai S-100 bus structure. I did carefully consider a machine based on the Z80 microprocessor because the Z80 is capable of operating at a higher speed and has a better instruction set than the 8080. However, to take advantage of the higher speed capability of the Z80 means that all memory chips must be able to operate faster and this means more expensive memory boards; generally in the order of 30% more for boards with an access time of 250ns against those with an access time of around 500ns. Also, I had read the results of a series of benchmark tests which had been run on a number of machines using various BASIC interpreters and they showed quite clearly that the efficiency of the software is far more important than that of the hardware. Further, for most home computer applications which are not on a time sharing scheme the operating speed of the CPU is relatively unimportant. Although I would have liked the instruction set of the Z80 there simply wasn't a machine on the market which was based on that microprocessor *and* which compared with the SOL in other respects. Time was important to me and to try and assemble a machine from a Z80 CPU board with the rest made up from a wide variety of sources would have involved me in too many time consuming problems and, in any case, I didn't feel there would be enough benefits to justify the effort which would have been involved.

As far as the S-100 bus is concerned, it may not be considered ideal by many people but I felt that it compared well with other available bus systems; it was well tried and proven and a wider range of peripherals and extension boards were available from a great many sources for the S-100 bus than for any other. In particular, memory extension boards are made in a range of byte sizes from 4K right through to 64K on one board.

The decision having been made, I bought a SOL-20 kit which came with 8K of static RAM on one extension board in addition to the memory facilities provided on the main PCB. An alternative kit is available with 16K of dynamic RAM but I decided to keep the initial cost down a bit and, on the advice of experienced computer friends, I decided to start with a reasonable quantity of static RAM in my system.

The kit was beautifully packed and protected and each separate circuit section was individually wrapped and labelled. The systems manual came in the form of a large loose leaf binder called an 'easel binder'. The cover is hinged from side to side as well as down the spine, so that when the binder is open the cover may be folded in half such that the top half forms a prop; and the entire binder will stand on a workbench at a convenient angle for reading and following assembly instructions. I can only say that the manual is one of the most complete I have ever seen, comprising some twelve sections and covering every aspect of the SOL-20 computer. The whole kit was complete right down to the last tiny washer and all instructions were delightfully clear and straight forward, making the machine relatively easy to put together with little trouble. The only problems I experienced were the power supply — which I traced to a faulty SCR — and with the displays which should have appeared as the result of tests carried out during construction. The manual calls for a number of tests to be made at various stages in construction, some of which produce a display pattern on the video monitor and photographs of the

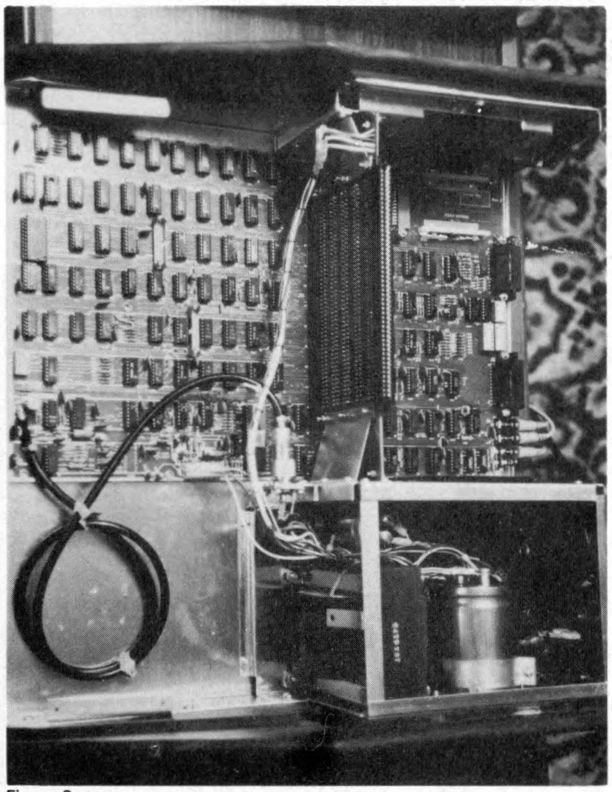


Figure 2.
Main PCB, backplane and power supply fitted into chassis.

correct displays are printed in the manual. The patterns which appeared on my monitor were not the ones shown in the manual so a brief telephone chat with the manufacturer was necessary to sort the problem out. In the first case the solution was to add an additional two IC's to the board before running the test and, in the second, it transpired that the waveforms shown in the manual had been printed upside down — both modifications have now been incorporated in the manual by Processor Technology who are the makers of SOL computers.

The cabinet and chassis are well designed with reinforcing of the chassis to support the heavy power supply transformer; the side cheeks are of solid walnut and the exterior metal-work is finished in a very good blue crackle paint. The aesthetic appeal of the machine is enhanced by its low line which is made possible by the design of the circuit boards. The backplane with its extension slots is mounted vertically from the centre of the motherboard which, in turn, is mounted in the bottom of the cabinet. This arrangement can be seen, together with the installed power supply, in the photograph shown in Fig. 2. Contrary to the practice of their competitors at that time, but since copied by others, Processor Technology built a complete computer on a single printed circuit board. That PCB, in addition to mounting the CPU, is also equipped with the following:-

1. **A video display module** which generates 1024 characters in sixteen lines, each sixty-four characters long. The cursor has multiple programmable circuitry and may be selected blinking or solid by a DIL switch on the main board. The VDM allows for white on black or black on white display, again by means of a single switch, and variable speed scrolling is provided up to a maximum of 2000 lines per minute, the speed being set from the keyboard during normal operation of the computer. A second display output is provided for expanded special applications.

2. **A monitor programme** in modular form contained in 2048 words of ROM, called a 'Personality Module' by Processor Technology, and plugged into a socket on the main board. This module is available in three versions and is accessible from the rear of the cabinet so that it may be changed easily. An end view of the module can be seen in Fig.4 on the right hand side of the rear panel. It can also be seen in more detail as the smaller of the two circuit boards shown in Fig.3. Details of the three modules are as follows:—

A) SOLOS. This is the ideal unit for stand alone computer applications. It is the one normally fitted for hobby computers and is the one I have in my machine. It provides control of the computer immediately from 'power on', because it resides in ROM, and enables machine language programming, cassette tape functions, control of I/O ports, control of the VDM, baud rates etc.

B) SOLED. This module sets up the SOL as an advanced editing terminal system. It allows for remote editing of files and cassette tapes, also large cassette data files or text messages can be transmitted and received automatically from remote locations.

C) CONSOL. A simple, low cost module giving minimal capability to the system.

3. **An audio cassette interface** is provided which is capable of controlling two recorders at 1200 baud in the Processor Technology CUTS (Computer Users Tape System) format. The interface is, however, also compatible with the slower but popular Kansas City standard at 300 baud and this format may be set from the keyboard under SOLOS control. The CUTS system is, of course, much faster — four times as fast to be precise — and will load the 8K BASIC interpreter, which I have, in less than a minute. It also has AGC provided in both read and write modes so there is little chance of losing bits even at 1200 baud.

4. **A UART** (Universal Asynchronous Receiver Transmitter) is provided for controlling data transmission through the serial I/O port and for controlling data written to or read from tape. The RS-232 and 20mA current loop standards are included and the SDI is accessed via a 25 pin 'D' type connector mounted on the main board but terminating at the rear panel. It can be seen in Fig.4.

5. **The parallel data** I/O port socket can also be seen in Fig.4: it is the male 'D' type connector adjacent to the cassette tape control sockets on the left hand side of the rear panel. This port provides eight data bits for input and eight data bits for output at standard TTL levels. The output bus is tristate for bidirectional interfaces. Four handshaking signals and three control signals are provided which allows up to four devices to share the parallel data interface connector.

6. **1024 words** of low power static RAM are provided for use as a scratchpad.

The computer comes with a software tape cassette which is recorded in CUTS 1200 baud format on side one, and Kansas City 300 baud format on side two. It contains BASIC/5, a small version of this most popular high level language, plus two

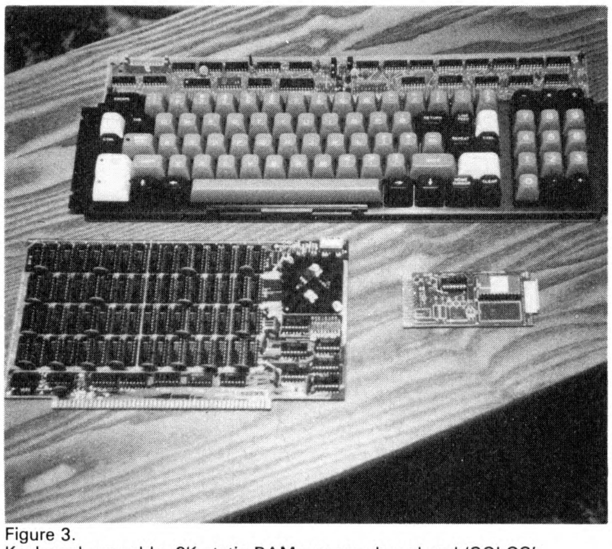


Figure 3.
Keyboard assembly, 8K static RAM memory board and 'SOLOS' personality module.

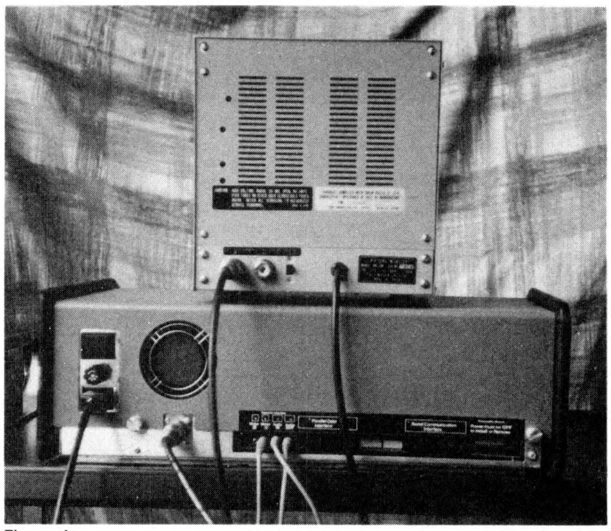


Figure 4.
Rear panel of SOL-20.

simple computer games. The BASIC interpreter, although only a small version, is nevertheless quite versatile, having a range of file operations, ARGUMENT and CALL functions which allows subroutines written in ASSEMBLY language to be linked into a BASIC programme, standard trig' functions plus ABS, INT, RND, SGN, SQR and TAB.

The first game on the tape, called MTCHS, is the ever popular match game more universally known as NIM, where the object is to force one's opponent to take the last match from a pile, or piles, with rules governing the number of matches which may be taken at each turn. The other game is called LUNAR and is one of an infinite number of games available for computers — and some programmable calculators — where the operator acts as the pilot of a spaceship, with limited fuel, trying to land on the surface of a planet. Height, speed, fuel remaining and elapsed time are printed out after each adjustment has been made by the pilot.

The 8K static RAM memory board is the larger of the two boards which appear in Fig.3. All address and data lines are fully buffered and extensive noise immunity circuitry is built in. Maximum worst case access time is 520ns and the module features switch selectable address selection which allows its starting address to be offset in 1K increments from 0 to 65K. Standby power facilities are provided with the use of two standard torch batteries to prevent data loss during power interrupt conditions; a very useful feature in home installations where the wrong plug is easily hauled out of a socket by a member of one's family — always, of course, just at the point where a long programme has been laboriously entered and debugged but not recorded on tape. I have subsequently bought a second 8K static RAM board made by another company which, although it operates perfectly and is rather less expensive than the Processor Technology board, lacks the standby facility.

Other items which I have acquired for the system include two software tapes which contain some quite complex games, a tape produced by MSL which contains an 8K BASIC interpreter, and a video monitor because this gives a far superior readout to a modified TV set. The MSL BASIC was purchased because at the time the Processor Technology 8K BASIC was not available and would not be so before my return to the UK.

I already had two cassette tape players and both of them seem to operate without any problems with all the P.T. software. The MSL BASIC does, however, tend to give some trouble on the 'LOAD' instruction, and settings of tone and volume are critical in order to achieve a satisfactory load of a programme from tape into memory when under the control of this interpreter. In addition, the documentation leaves a lot to be desired and I now find that I spend a lot of time in experimenting with small sections of programme in order to find out for myself exactly what the limitations are in certain areas. I am sure there are some areas I cannot fully exploit with this interpreter because I cannot establish the exact syntax. Nevertheless, it is a useful addition to the system and I have used it extensively, especially for programmes requiring complex string manipulations.

Supporting software for the SOL-20 includes some very useful and interesting programmes and a few brief details of the following will help to show the comprehensive range of the system.

1. ALS-8 programme development system. This is an assembler which allows up to six source programmes to be stored in memory as named files and called at will to be listed, edited, assembled or simulated. ALS-8 has the unusual ability to dynamically adjust the system's I/O handling configuration. For example, if the system includes a CRT terminal, a high speed line printer, paper tape reader/punch and a teletype, then with ALS-8 the system can print a listing to the line printer, then input from the paper tape reader and return console control to the CRT terminal or teletype, all under programme control.
2. SIM-1: this is an interpretive simulator programme which operates as though it were an 8080. With the SIM-1/ALS-8 combination it is possible to simulate 8080 programmes on any S-100 bus structure microcomputer without actually running them in real time. All registers, flags, stack and programme counter are simulated and programmes can therefore be tried out with no fear of crashing the system if something goes wrong. The system doesn't lose control if a programme error is encountered such as an incorrect jump or call. With SIM-1 one can set breakpoints, enable or disable register or memory content printout. I/O instructions can be run in real time, simulated from the system console, or set to predetermined values for any I/O port address. This is a very powerful debugging tool indeed for 8080 progamming.
3. TXT-2 text editor. Through the use of this programme it becomes possible to insert, delete and move single characters, entire lines or portions of lines. Complete text files can be scanned at several user controlled rates up to almost 2000 lines per minute.
4. 8080 FOCAL high level maths language.
5. TREK-80 — a very sophisticated game based on the well-known series Startrek. You can warp through one hundred quadrants of hyperspace, fire phasers, photon torpedoes, antimatter pods and experimental rays, all in real time. The Klingons lurk in many of the quadrants and are out to destroy you — they will too if you don't keep your wits about you!
6. GAMEPAC 1. This tape contains four video games which keep family and friends highly amused and they demonstrate well some of the capabilities of the machine. The problem is, they are so enthralling that once they've been introduced the poor owner can't get near his computer for hours.

Well, so much for software. To complete the system there is a range of interface boards and peripherals which include wire wrap and extension boards; 3P+S input/output module; PROM programmer; multi-channel analogue interface; joystick controls and a high speed paper tape reader. Processor Technology also publish for owners, and others who may be interested, a house magazine called 'Access' which prints updated information on systems development, new software, readers letters, interesting programmes etc.

Finally, for those to whom money is no object, there is what might be described as the most desirable additional piece of equipment which might be yearned after by any computer hobbyist.

This is the HELIOS II dual floppy disc drive complete with controller, system diskette with disc operating system, power supply, case, all interconnecting cables, full systems documentation and a 12K assembly language programme to test and report on every aspect of the unit. Standard Helios II storage capacity is over 750,000 bytes and will load an 8000 byte programme with a look-up in the system directory in 0.3 seconds. Disc BASIC — of course.

Ah well, I can dream can't I?

I'M A MICRO-COMPUTER WIDOW

Linda A. Grand

Alright, so I'd accepted that having a scientific husband meant that calculators were likely to feature prominently in our lifestyle. I raised no objections when the old model (just simple little features like square root and logs) was superseded by a *scientific* calculator (complete with an 'inv.sin.lnx.' which sounded to my tender brain like a particularly nasty form of spots). Alas, this instrument lasted merely a few weeks before it too was obsolete, and a longing for a 'programmable' began to possess my beloved. At first I thought this device had something to do with remote control television but I was enlightened quickly when, on shopping trips, instead of me doing the window shopping over dresses, fur coats and other little desirables, it was *him*, drooling over what looked to me impossibly large square objects rather like the things they carried on Star Trek. But, at last, the weeks of indecision were over and his fervour came to rest on a Sinclar Cambridge programmable. Quite honestly, he was like a father with his first child after he had bought it — have you ever sat in a multi-storey car park for *two hours* whilst sir got the hang of his new gadget? Not to mention that he played with it whilst waiting in traffic jams!

Well, we'd bought the calculator of his dreams and the next week or so was spent thinking up programmes to satisfy the beast's seemingly insatiable appetite. But, sadly all too soon (about two weeks) having only a mere 36 steps began to pall and my husband began perusing calculator advertisements, like an alcoholic newly off the bottle, in search of a better, newer, bigger . . .

So, there we were in the multi-storey again with his new toy (a Casio fx201p) but this time I'd craftily rushed him off without buying any batteries, in the hope of getting home in time to watch the epilogue on TV, until (gnash, gnash) he discovered that the makers had thoughtfully *included* batteries.

"It works in Fortran," he said, in tones worthy of someone looking at the Mona Lisa for the first time. "Very nice dear," said I, "is it a new type of breakfast cereal?" . . . it's a long way from that multi-storey to our house, you know!

All was quiet in our household for a few weeks, the new toy was *fortranising* busily, I was allowed to take the old one to the supermarket and my husband began reading electronics magazines again. Now, that should have warned me that something was about to happen, but I carried on blissfully unawares, until one day I noticed he was gazing at the tele with a somewhat possessive air. "Thinking of buying us a new one?" I enquired, "No, no," was the absent minded reply, "it'd make ever such a good v.d.u.". Now, I'd heard they had confidential clinics for that sort of thing at the local hospital so I retired to the kitchen to think about it.

Later that night, my beloved waxed enthusiastic about his new passion (not me, sadly). Words like bytes and nibbles (which were what I'd thought I was for), and strange codes like CPU, RAM and EPROMS began flying about the bedroom like singularly irritating flies. "MICRO-COMPUTERS," he exclaimed, leaping about the bed, in a way he hadn't done since our honeymoon, "they're the thing of the future, and we're going to have one, NOW." (To be absolutely honest, he sounded like a bad washing powder advertisement.)

So for the next few weeks, we experienced the familiar agonies of indecision, magazines were read until the pages dropped off, and instead of discourses on the price of meat the relative desirability of having more bytes of additional RAM, or even (God forbid) a floppy disc! (no less), began to dominate our evening conversations. That is, they dominated when my beloved was not sunk in gloom, trying to decide whether 'twas better to have a 77-68 (which was cheaper) or a Nascom (expensive, but playable with straight away). I became accustomed to being assaulted at all times of the day or night, and having 'THE DECISION' announced, the trouble was it changed every time!

Then a new problem presented itself, as himself casually announced one day, "Of course, you know, it'll need a Basic interpreter," — "But we haven't got room for a lodger!" I exclaimed in some trepidation. Happily, the aforementioned individual turned out simply to be yet another magic box, which apparently converted from something called hexadecimal (whatever that might be — my husband did explain, but I didn't have enough toes to keep up).

Well that's the position our household is in at the moment, we (or at least he) know we're going to have one, the only problem is what sort of a one. But, at least I've got one comfort, and that is that the damn thing will come through the post, so at least I won't have to endure the multi-storey until some ungodlike hour this time! ". . . you're what dear? . . . oh, you're thinking of driving down to London to buy it? . . . Well, *I'm* not coming!"

P.S. I've caught on to all the terminology and have decided that the only way to gain my beloved's attention at the moment is to be a computer, so here goes with my specifications:—

I am a CCU with DWP and optional BNTHM.

(in English — Central Cooking Unit, with Dish Washing Peripherals and additional facilities for Being Nice To His Mother).

THE EXPANDABLE GENERAL-PURPOSE MICROCOMPUTER

RESEARCH MACHINES 380Z

COMPUTER SYSTEMS

A professional microcomputer system with excellent support for scientific and educational users.

380Z systems come complete with:

- Z80A Processor
- RAM Memory
- 3K ROM Operating System
- VDU Interface
- Cassette Interface
- Parallel Input/Output
- Keyboard
- Room for Expansion
- Documentation

Only an inexpensive cassette recorder and a television are needed to make the system fully operational. No teletype or VDU terminal are required.

The standard 380Z system is expandable to include:

- Additional Memory
- Printers
- Serial Interfaces
- Parallel Interface with Real Time Clock
- Floppy Disc Systems
- Analogue Interface

The RESEARCH MACHINES 380Z is a professional general-purpose microcomputer. It has found broad application in Scientific Research and Education with a large number of systems in use in the field. The system is supported by an excellent range of software including Text Editor, Basics and Assemblers.

RESEARCH MACHINES 380Z System
with 16K Bytes of RAM

£965.00

RESEARCH MACHINES 380Z System
with 32K Bytes of RAM

£1158.00

RESEARCH MACHINES Computer Systems are distributed through SINTEL, P.O. Box 75, 209 Cowley Road, Oxford. Telephone: OXFORD (0865) 49791. Please contact SINTEL for the 380Z Information Leaflet. Prices do not include VAT @ 8% or Carriage.

PCW OPEN PAGE

The Amateur Computer Club View

Mike Lord

More Local Groups
1978 seems to be the year of the local group. The established ones have, in general, been going from strength to strength (the one sad exception being the collapse of the London Group, if only for personal reasons I'd like to hear from anyone willing to start a new ACC group in the London, or Essex areas), and new groups are constantly being formed.

The Cambridge University Processor Group is open to anyone in the Cambridge area, not just members of the University, and they would especially welcome anyone willing to help with their next constructional project, having just completed a 2650 based system. Your contact man is **Tim Hopkins,** Magdalene College.

The Thames Valley Group is thriving, anyone interested in activities in the Reading area should get in touch with **Bob Cottis,** 'Pippins', Boulters Lane, Maidenhead SL6 8JT.

Kentish folk should get in touch with Mr. **A. Aylward** of 149 Balmoral Rd., Gillingham, who wants to start a local group in that area.

For ACC members in Devon, I received two letters on the same day, both about starting groups in that area. They were from Mr. **D. Carne**, 44 George St., Exmouth EX8 1LQ, and Mr. **G. V. Barbier** of Palmers Mill, Calverleigh, Tiverton.

Again, a reminder that if you write to any of the above, a stamped addressed envelope would be appreciated.

More Microcomputers
Having begun to hear from people who have now got NASCOM, PET & TRS-80 systems running I'm finally convinced that these machines actually exist! In fact, the owners of all these machines seem equally pleased with their purchases and no complaints have reached my ears. There seems to be a place for independent user groups for these systems — apart from the manufacturers' sponsored ones — and the ACC will support any moves in this direction.

I haven't heard yet from anyone who has bought a 'Science of Cambridge' MK14, although surely the low price and good publicity must have sold several hundred kits by now.

More Exhibitions
The Manchester Group of the ACC appeared at the recent Amateur Radio Exhibition and have reported that their stand was almost besieged at times by interested visitors. It seems that a lot of amateur radio enthusiasts are now taking an interest in microprocessors — so other local computer groups should take note and get in touch with their nearest radio club.

The big event of the year, of course, promises to be the 1978 Do-It-Yourself Computer Show from 22 to 24 June at the West Centre Hotel, London. Three solid days of conferences, exhibition and what are quaintly referred to as 'side shows' (pugilistic contests between the scholars supporting the Eurobus design and the businessmen with their S-100?). Although details haven't been settled at the time of writing, the ACC will be there and will welcome all visitors to their stand.

Finally
If you would like your local group's activities to be publicised in this column, just drop me a line (allow 6 to 8 weeks for publication delays) or, for more details of the Amateur Computer Club, send an SAE.
Mike Lord 7 Dordells, Basildon, Essex SS15 5BZ
PCW We're very pleased to inform readers that the ACC will have a free stand at our exhibition being held on 21st-23rd September. **PCW**

I should like to draw your readers' attention to courses run at the College which might be of interest. In the evenings over the past year we have run introductory courses each of six weeks duration on microprocessor systems. These courses are:
a) Introduction to Microprocessor Hardware
b) Introduction to Software
c) Microprocessor Workshop

The Microprocessor Workshop course is completely laboratory based where students can use the College systems which include 6800, F8 and Z80 systems. For some of the systems we have available co-resident editor assemblers, cross-assemblers for running on the College computer and PROM programming facilities. Also available for student use is the College test equipment which includes oscilloscopes, signal generators, etc. for investigation of the hardware operation on the College or their own systems.

The fees for these courses are modest, being £2.00 for each course of six weeks duration (1977 prices).

I noticed that your editorial policy was to encourage the formation of Personal Computing Clubs and it might be possible to form such a club at the College, based on the Workshop evening.

In passing, I should also like to mention that we do also run Microprocessor system design courses aimed at the professional electronics engineer who requires updating in this particular field. These courses are run on a part-time day basis over a period of nine weeks.
W. R. Wittams,
School of Engineering
Merton Technical College,
London Road, Morden, Surrey.

MUSE NEWS: Summer Course 1978. Held at Westhill College, Wedley Park Rd, Selly Oak, Birmingham.

The course is organised so that all participants will opt to do one of the A Units (3 sessions) and one or two of the B Units (total 2 sessions).

A Units

1. **The "ED80" Assembly language** Roy Atherton (Bulmershe Coll)
   An educationally viable subset of the Z80 machine language developed for the Research Machines 380Z. The course will cover: Addressable bytes, Monitor and Usable memory, ROM and RAM, Registers and Memory bytes, Extended & Immediate addressing, Indirect addressing, Output.
2. **Writing major programs in BASIC** Hugh Williams (Trent Poly)
   Top-down analysis of problems: structuring BASIC programs: simulation programs: documentation: portability and transportability.
3. **BASIC for beginners** Erica Moody (Westhill Coll)
   Just starting computing? Or bring a colleague who wants to get started. There will be opportunties for practical work on the College system.
4. **Computers in the Classroom** Bob Eadie, Charles Sweeten and David Pegg (Bedford School and Oundle School)
   Three experienced users of small computers in schools will describe the use of the computer and of packages by teachers with little knowledge of computing. The course will cover writing of packages for other departments, a foolproof operating system, use of files in the classroom, and examples of test packages.
5. **M6800 Machine Language Programming** John Coll (Oundle School)
   The process of creating a source program using an Editor, and creating the machine language program using an Assembler. The course covers the main elements of Assembly language: registers, accumulators, index, modes of addressing memory, branching. Also covered will be the architecture of the M6800, input/output routines, the parallel interface, use of the Monitor program subroutines.

B Units

1. Comparing high level languages
2. Microcomputer architectures
3. O/A projects using M6800 Assembler
4. Demonstrations of various machines
5. Practical work in BASIC

Booking Arrangements

The full cost of the course is £21 (MUSE members £20). If more than one person comes from the same institution there will be a reduced rate of £18 per person (MUSE members £17.50) for everyone from that institution.

Applications should be sent to the course organiser, Hugh Williams, 90 Harrow Rd, West Bridgford, Nottingham to arrive by 30th June.

Final details will be sent out in the first week in July. If the course is oversubscribed those we cannot take will be notified at once.

A deposit of £5 per person should accompany the application.

Please indicate order of preference for Units. Where numbers are limited they will be filled in order of application.

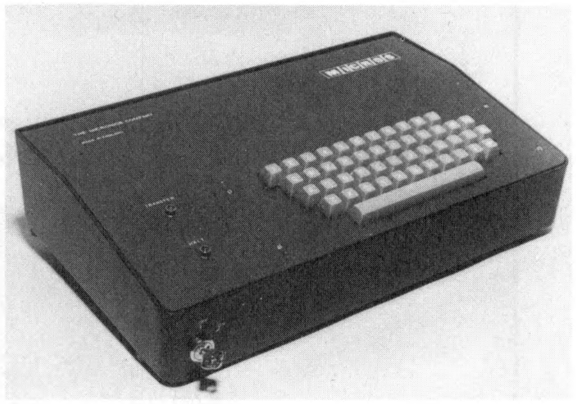
It may be possible to reduce the cost to members living near Westhill who do not need accommodation and all meals. Please write to the organiser indicating what you want to come for and he will quote you a price.

Further information (not bookings) from the organiser phone Nottingham 48248 ext 3253 (weekdays until 7 July) Nottingham 233418 (at other times).

ANNOUNCING THE MICROS

JOIN THE MICRO REVOLUTION

Access We welcome Access — master charge THE INTERBANK CARD — EUROCARD



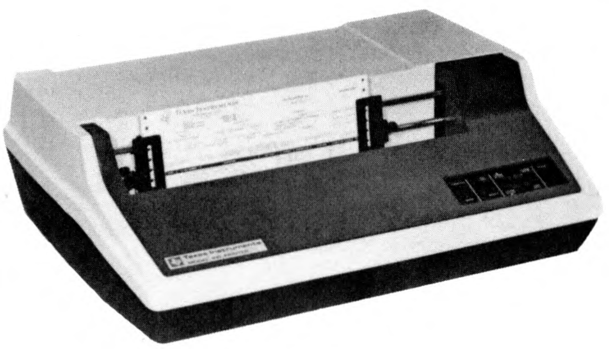
£399 for a Z80 based microcomputer, built and tested

Designed for educational establishments, personal computing and small business users

- * Includes 1K monitor Eprom, 47 key solid state keyboard, video, TV, cassette and teletypewriter interfaces,serial i/o, 2 parallel i/o ports, 2K bytes RAM, power supplies and instrument housing.
- * Connect to domestic TV or video monitor to complete the system
- * 48 x 16 character video matrix
- * 47 key contactless ASC11 keyboard
- * Hard copy on teletypewriter
- * 2 TTL compatible parallel i/o ports
- * RS232 serial i/o port
- * Load and dump programmes on unmodified cassette recorder
- * Up to 16K byte mixed RAM and Eprom in table top housing
- * Expandable up to 64K bytes
- * Security locked power switch
- * British designed and built
- * Available in kit form for £360
- * Credit terms available

THE MICRONICS COMPANY
1, STATION ROAD TWICKENHAM MIDDLESEX

PART OF THE MICRO REVOLTION
Prices exclusive of VAT and carriage



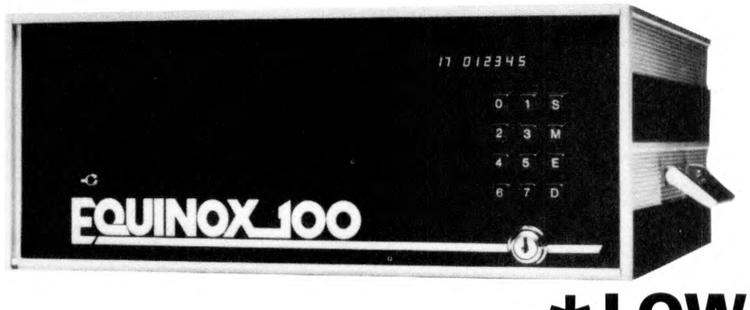
★ 132 COLUMNS
★ SERIAL INTERFACE
★ BI-DIRECTIONAL
★ 9 x 7 DOT MATRIX
★ FORMS UP TO 15" WIDE
★ 150 CPS (UP TO 440 LPM)

★ 15" SCREEN
★ BLOCK MODE
★ FULL ASCII CODE
★ 95 KEY KEYBOARD
★ 24 x 80 CHARACTERS
★ PROTECTED FIELDS

EQUINOX

THE RELIABLE COMPUTER SYSTEM

★HIGH PERFORMANCE



★LOW COST

EQUINOX SYSTEM ONE PLUS OTHER SYSTEMS, PERIPHERALS AND S-100 COMPATIBLE COMPUTER BOARDS ARE AVAILABLE FROM:

INTERAM COMPUTER SYSTEMS LTD.,
59 MORETON STREET, LONDON SW1
TELEPHONE: 01-834 0261/2733

INTERAM

PSEUDO RANDOM NUMBER GENERATOR FOR PROGRAMMABLE CALCULATORS

C. CLARK

The following article is written specifically for the Sinclair Cambridge Programmable but it is applicable to any programmable that has the 'interrupt' and decision facilities. The flowchart is shown in Figure 1 with commentary.

The article as a whole will enable beginners to get to grips with programming by modifying the program to their particular requirements. Detailed explanations of the steps are given for building the program in Figure 2. The more experienced will be able to experiment with the 'infinite loop' program in other applications.

Five keystrokes are necessary to obtain the desired result when the calculator is running in the infinite loop and seven keystrokes are required to re-run the program again. This may seem a cumbersome way of obtaining a random number, but the disadvantages are offset against the fact that psuedo random number formulas will always produce the same series of numbers for the same starting point where as the described program is truly random dependent upon the interrupt instant of time. By following the step by step explanations it will help the novice programmer to appreciate what is being keyed in and out of the microprocessor and its memory contained in the calculator. This appreciation will help one develop an understanding of more complex systems built or bought at a later date.

Uses for this program can be many, a few suggestions are:—

1. Random number generator for statistics. (Up to 100)
2. Dice or double dice thrower. (6 or 12)
3. Invisible roulette wheel. (36)
4. Pools number selector. (56)

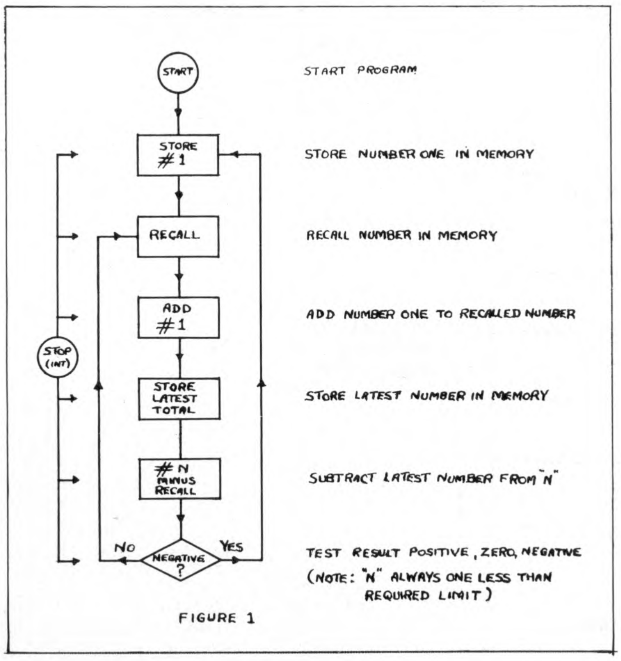

START
STORE #1
RECALL
ADD #1
STORE LATEST TOTAL
#N MINUS RECALL
NEGATIVE ?
NO
YES
STOP (INT)
START PROGRAM
STORE NUMBER ONE IN MEMORY
RECALL NUMBER IN MEMORY
ADD NUMBER ONE TO RECALLED NUMBER
STORE LATEST NUMBER IN MEMORY
SUBTRACT LATEST NUMBER FROM "N"
TEST RESULT POSITIVE, ZERO, NEGATIVE
(NOTE: "N" ALWAYS ONE LESS THAN REQUIRED LIMIT)

FIGURE 1

KEY STROKE	CHECK SYMBOL	STEP NUMBER	COMMENTS
△/▽			KEY SEQUENCE ADDRESSES MICROPROCESSOR MEMORY SECTION AVAILABLE FOR STORING OR RUNNING PROGRAM. (36 LOCATIONS 00-35)
△/▽			
2			
0			
0			
△▽			SWITCH TO "LEARN" MODE. KEYS NOW PRESSED ENTER SYMBOLS AND DATA INTO MEMORY SECTION FOR BUILDING A PROGRAM
LEARN			
#	3	00	STARTING WITH NUMBER ONE
1	1	01	
=	–	02	REQUIRED AFTER SINGLE NUMBER ENTRY
STO	2	03	STORE NUMBER IN CALCULATOR MEMORY
RCL	5	04	RECALL NUMBER INTO WORKING REGISTER
+	E	05	ADD ANOTHER ONE TO NUMBER
#	3	06	
1	1	07	
=	–	08	RESULT
STO	2	09	STORE LATEST NUMBER IN MEMORY
#	3	10	NUMBER "N" = ONE LESS THAN LIMIT REQUIRED
5	5	11	
5	5	12	
–	F	13	SUBTRACT NUMBER IN MEMORY FROM "N"
RCL	5	14	
=	–	15	
▽	A	16	TEST RESULT POSITIVE OR NEGATIVE
GIN	1	17	
0	0	18	IF NEGATIVE GOTO START WITH NUMBER ONE. IF POSITIVE OR ZERO PROCEED TO NEXT STEP (20)
0	0	19	
▽	A	20	GOTO STEP 04. WHICH CONTINUES TO ADD ANOTHER NUMBER ONE TO LATEST TOTAL
GOTO	2	21	
0	0	22	
4	4	23	
C/CE	SIGNALS END OF BUILDING PROGRAM. BACK TO CALC. MODE		

FIGURE 2

Figure 3 shows the operating actions for running the program along with a detailed commentary.

This program demonstrates the fact that a computer can be running in an invisible infinite loop, intentional in this case, but on other occasions unintentionally due to a programming error (software fault). Yet it would appear that the electronic device has developed a fault (hardware fault) (i.e. nothing showing in the display). So the lesson to be learned is to run a simple test program first before jumping in with a soldering iron or returning the wrongly suspected hardware to the makers.

The programmable calculator is a valuable starting point into the world of electronic data processing (EDP). It helps one to visualise the speed, power and pitfalls in larger systems.

KEY STROKE	COMMENTS
△/▽	SWITCH TO OR "LOAD" STARTING ADDRESS IN THIS CASE LOCATION OR "STEP" NUMBER 00
△/▽	
2	
0	
0	
RUN	START PROGRAM RUNNING
	THE DISPLAY REGISTER GOES BLANK BUT PROGRAM RUNS INTERNALLY. PROGRAM NOW RUNS IN THE "INFINITE LOOP" COUNTING BY ONES UP TO SET NUMBER "N" PLUS ONE AND THEN RETURNING TO NUMBER ONE UNTIL INTERRUPT IS KEYED.
÷	INTERRUPT SIGNAL. PROGRAM STOPPED AT THAT INSTANT IN TIME WHEREVER THE FLOW SEQUENCE OF EVENTS HAPPENS TO BE. IGNORE WHAT IS DISPLAYED WHEN INTERRUPT OCCURS. THE CONTENTS OF THE MEMORY STORE ARE NOW REQUIRED TO BE SWITCHED INTO THE DISPLAY.
C/CE	TWO "CLEAR" KEYING ACTIONS ARE USUALLY REQUIRED TO NORMALISE THE PROGRAMMABLE. THE DISPLAY SHOULD THEN SHOW A SINGLE ZERO
C/CE	
△/▽	THE REQUIRED RANDOM NUMBER IS NOW DISPLAYED
RCL	
C/CE	CLEAR DISPLAY AND REPEAT ABOVE OPERATIONS AS REQUIRED

FIGURE 3

A GUIDED TOUR OF THE Z80

NEIL HARRISON

Early in 1976 Zilog introduced their 8 bit microprocessor, the Z80. In spite of its high initial cost it received an enthusiastic welcome from the well established American personal computing community who were quick to appreciate its capabilities. Here in the UK, at the beginning of our own home computing explosion, the Z80 is available for very little more than the cost of perhaps its greatest rival the 6800. The fact that the majority of recently designed British microcomputer systems have used the Z80 means that it is likely to be the most popular microprocessor for home, school and small business use for some time to come. Here then is a Guided Tour round the Z80, by no means exhaustive, but sufficient to show the powerful features available to the programmer.

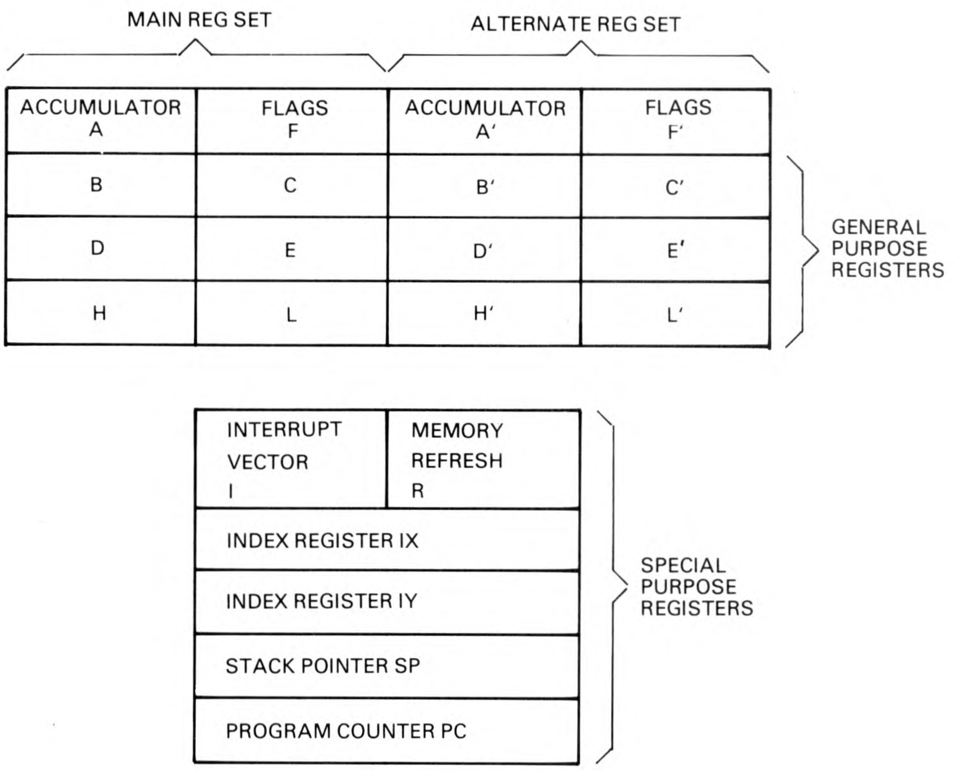


Figure 1 Z80 Registers

Inside the Z80

There are 208 bits of read/write memory inside the Z80 CPU available to the programmer. This memory is arranged as eighteen blocks of 8 bits and four of 16 bits called Registers. Each register is referred to by a one or two letter name for convenience. All the registers and their names are shown in fig. 1.

There are two 8 bit accumulators, A and A′ and two associated flag registers F and P′. Only A and F are available to the programmer directly; however, these can be exchanged with A′ and P′ by a single instruction. The accumulator is used for general storage and data transfer and holds the results of all 8 bit arithmetic and logical instructions. The flag register contains a number of single bit *status* flags which indicate special conditions resulting from the last operation. The format of the flag register as seen by the programmer is as follows:—

S	Z	X	H	X	P/V	N	C

X — Not used

S:	The Sign flag indicates whether the result of the last operation was positive or negative (1 = —ve, 0 = +ve)
Z:	The Zero flag is a 1 if the result of the last operation was a zero. Otherwise it is 0.
H and N:	These two flags are used by the Z80 to implement Binary Coded Decimal (BCD) arithmetic if required.
C:	The Carry flag is a 1 if the result of the last operation caused a carry (or borrow) from the most significant bit of the accumulator.
P/V:	This flag is dual purpose and indicates parity after a logical operation and overflow after an arithmetic instruction.

There are two sets of six general purpose 8 bit registers B,C,D,E,H,L and B^1,C^1,D^1,E^1,H^1,L^1. Only one set can be used at a time and they can be exchanged *as a block* by a single instruction. The six registers can be *grouped into pairs* to form three 16 bit registers called BC, DE and HL. These can be used (1) to hold memory addresses (2) as 16 bit counters or (3) for double precision arithmetic.

The remaining registers are used for special functions. The Program Counter, PC, is a 16 bit register used to hold the address of the current instruction being fetched from memory. It is automatically incremented after each instruction as the program is executed.

The Stack Pointer, SP, holds the address of the top byte of a 'stack' located anywhere in external RAM memory. The stack is arranged as a last in, first out memory. The contents of CPU registers can be 'pushed' onto the stack or 'popped' off the stack for temporary storage. The last data pushed onto the stack is always the first popped off. The stack is particularly useful for subroutine calls when it is used for storage of the return address.

The Index registers IX and IY hold independent 16 bit addresses that are used in the Indexed mode of addressing as 'pointers' to a region of memory where data is stored.

The Interrupt Page Address register, I, is used in one of the Z80's three interrupt modes. Using the contents of this register and 8 bits supplied by the interrupting device a 16 bit address is formed. This address points to a table of interrupt routine addresses allowing the Z80 to perform an indirect subroutine call to anywhere in memory in response to an interrupt. The Memory Refresh Register, R, acts as a counter for refresh of dynamic memory. It is incremented after every instruction fetch and is not normally used by the programmer.

The Outside World

To the programmer the world outside the CPU is in two sections, memory and input/output ports. The Z80 uses 16 bit addresses and so can individually access up to 65536 (64K) memory bytes. This can consist of read/write memory (RAM), read only memory (ROM) and is generally used for program and data storage. External hardware devices such as keyboards and printers can be connected to the Z80 busses so that they respond to particular memory addresses. This technique is called *Memory-mapped* input/output and is widely used in mini and micro computers. As an alternative the Z80 has up to 256 input and 256 output ports for peripheral devices accessed using special Input/Output instructions

The Instruction Set

The Z80 can execute 694 different instructions which are variations of 158 basic types. The instructions can be broken down into the following major groups:—

1. Load and Exchange
2. Arithmetic and Logical
3. Shift and Rotate
4. Jump, Call and Return
5. Input/Output
6. Block Transfer and Search
7. Bit Manipulation
8. General CPU Control

These groups will be discussed individually later, before we need to know how the Z80 specifies the location of the data operated on by an instruction. The data may be stores in a register, a memory location or an I/O port. We refer to where the data comes from as the Source and where it is eventually stored as the Destination. The source and destination are specified using one of the ten possible Addressing Modes. Here is a brief description of how each addressing mode specifies the location of the data.

1. Register addressing
   The data is stored in one of the CPU registers. An example would be to load the E register (destination) with the contents of the accumulator A (source). Register addressing is used for both source and destination.
2. Immediate addressing
   The data is stored in the memory byte immediately following the instruction. This mode could be used when loading a register with a fixed value.
3. Immediate addressing Extended
   This operates in the same way as Immediate mode except that 16 bit (2 byte) data is involved. It might be used to load the BC register pair with the two bytes immediately following the instruction, the first byte into C, the second into B.
4. Extended addressing
   In this mode the two bytes following the instruction are used to define a 16 bit memory address. This address may be the source or destination of some data or it might be an address which the program will jump to when the instruction is executed.

5. Relative addressing
   The byte immediately following the instruction is used as a displacement from the present memory address so that a relative jump can occur. Relative addressing is used to jump to nearby locations and is useful because it is only a two byte instruction instead of the three bytes of a normal jump using Extended addressing. The displacement byte is treated as a signed number allowing jumps backwards and forwards in memory. The jump range is between 128 bytes before and 127 bytes after the address of the next instruction.
6. Implied addressing
   In some instructions the source of destination is determined by *the instruction type itself.* Some examples of this are the arithmetic and logical instructions where the destination of the data is always the accumulator.
7. Register Indirect addressing
   This mode of addressing uses the contents of one of the 16 bit register pairs (BC, DE or HL) as the address of the data.
8. Indexed addressing
   In this mode the byte of data following the instruction contains a signed displacement which is added to one of the 16 bit Index registers. The resulting 16 bit number is the address of the data. The contents of the Index register are unaffected by the operation.
   Indexed addressing is a powerful way of addressing tables of data since the index register can point to the start of any table. Any memory byte in the range —128 to +127 bytes of the address in the index register can be used directly in arithmetic, logical, shift or rotate instructions.
9. Bit addressing
   The Z80 has instructions for manipulating individual bits in any CPU register or memory address. Bit addressing defines which of the 8 bits in the byte (0 to 7) is specified. The location of the byte itself is specified by Register, Register Indirect or Indexed addressing.
10. Modified Page Zero addressing
   There are 8 single byte subroutine calls or Restarts which use addresses in Page Zero of memory (location 0000 — 00FF). These instructions have the power of a 3 byte Call instruction and can save memory if frequently used subroutines are located at these Page Zero addresses.

Instruction Groups

Load and Exchange

The Load instruction, mnemonic LD simply moves data from the source to the destination. The addressing modes used determine whether 8 bit or 16 bit data is to be moved. The general form of the instruction is,
LD destination, source.
Some examples of the load instruction at work:—

LD B, D	Load the data in D into B.
LD HL, 1234	Load the value 12 into H and 34 into L.
LD (HL), A	Load the data in A into the memory location pointed to by HL.

The last instruction shows the use of brackets to indicate that HL contains the address of the data rather than the data itself. This is the Register indirect addressing mode.

The Exchange instructions swap data between the source and destination. An example is the EX DE, HL instruction which swaps the data in the DE register pair with HL. The alternate set of registers are accessed using two exchange instruction. EX AF, AF[1] swaps the current accumulator and flags with the alternate set and EXX swaps the two sets of six general purpose registers.

Arithmetic and Logical

All 8 bit arithmetic and logical operations are performed between the accumulator and an Operand specified by one of the addressing modes. the result is stored in the accumulator and the flags are set or cleared accordingly. The mnemonics for these instructions and the operations performed are as follows,

ADD A, Operand	A = A + Operand
ADC A, Operand	A = A + Operand + Carry bit
SUB Operand	A = A — Operand
SBC A, Operand	A = A — Operand — Carry Bit
AND Operand	Logical AND of A with Operand
OR Operand	Logical OR of A with Operand
XOR Operand	Logical Exclusive OR of A with Operand
CP Operand	Compares A with Operand and sets the flags. A is unaffected.

This group also contans the Increment and Decrement instructions

INC Operand	Operand = Operand + 1
DEC Operand	Operand = Operand — 1

The accumulator is not affected by these two operations.

Sixteen bit arithmetic between CPU register pairs is carried out using either HL, IX or IY as a 16 bit accumulator.
For example:—

ADD IX, BC	BC is added to IX, result in IX

Shift and Rotate

The Z80 can shift or rotate the contents of the accumulator, any register or any memory location. A BCD digit in the accumulator can be rotated the two digits in a memory byte pointed to by HL.

Jump, Call and Return

The Jump instructions transfer program execution to a memory address specified by the operand. Jumps can be conditional on the state of one of the CPU flags. If the condition is true the jump is executed, if not the program continues to the next instruction. Absolute Jumps, mnemonic JP, specify a condition and a 16 bit address to allow program transfer to any memory address:

JP C, address	Jump if the Carry flag = 1
JP NC, address	Jump if the Carry flag = 0
JP Z, address	Jump if the Zero flag = 1
JP NZ, address	Jump if the Zero flag = 0
JP PE, address	Jump if Parity flag = 1 (even parity)
JP PO, address	Jump if Parity flag = 0 (odd parity)

JP M, address	Jump if Sign flag = 1 (minus)
JP P, address	Jump if Sign flag = 0 (plus)

Relative Jumps, mnemonic JR transfer program execution an address within — 128 and + 127 bytes of the next instruction. Jumps can be conditional on the Carry and Zero flags.

The 'Decrement and Jump if Not Zero' instruction is particularly useful for program loops. This two byte instruction decrements the contents of the B register and performs a relative jump if the result is not zero.

A subroutine Call works in the same way as an absolute jump except that the address of the instruction following the call is pushed onto the external stack in memory. When a Return instruction is executed the address is popped off the stack into the Program counter, continuing execution in the main program. Subroutine calls can be 'nested', that is more subroutines can be called within a subroutine, to a depth limited only by the memory available for the stack. Calls and returns can be conditional on the same status flags as absolute jump instructions.

Input/Output

There are two types of I/O instructions for transfer of data between registers and the 256 input and 256 output ports. The port can be specified by the byte following the I/O instruction in which case data is transferred through the accumulator. Alternatively the port can be specified by the contents of the C register and data can be transferred to or from any general purpose register.

Eight special instructions are used to transfer blocks of data via an I/O port. The HL registers are used as a memory pointer, the B register is used as a counter and C defines the port being used. Since B is 8 bits long blocks of up to 256 bytes may be transferred. The four special instructions for input are . . .

INI	Input from the port defined by C to the memory address in HL, increment HL, decrement B.
INIR	As INI but repeat until B = O.
IND	Input from the port defined by C to the memory address in HL, decrement HL, decrement B.
INDR	As IND but repeat until B=O.

The four block output instructions are similar to the above but with data trnsfer from memory to the port.

Block Transfer and Search

This group contains a number of very powerful instructions for moving blocks of data around in memory.

The three register pairs are used by the instructions for the following;

HL points to the source address
DE points to the destination address
BC is the byte counter.

Once the data in these registers is set up correctly one of the four block transfer instructions can be executed. The 'load and increment' instruction, LDI, moves the byte at the address pointed to by HL to the address pointed to by DE and BC is decremented. HL and DE are incremented to point to the next memory locations. Load, increment and repeat performs succesive LDI instructions until BC is zero. LDD and LDDR move blocks in the same way except that DE and HL are decremented after each move.

In the compare and increment instruction, CPI, the contents of the accumulator is compared with the memory loation pointed to by HL. The CPU flags are set, HL is incremented and BC decremented. Compare, increment and repeat, CPIR continues comparison until a match is found or BC equals zero. There are equivalent instructions CPD and CPDR which decrement HL after comparing.

Bit Manipulation

There are three basic operations, Bit test, Set and Reset. Bit test sets the Zero flag to indicate the state of the bit, Set forces the bit to a one, Reset to a zero. Used with the various addressing modes these instructions can test, set or clear a bit in any register or memory location.

General CPU Control

The No operation instruction does nothing and the Halt instruction suspends CPU operation until an interrupt or hardware reset is received. The remainder of this group control the response of the Z80 to an external Interrupt. One of these modes of interrupt response may be selected;

Mode 0	The Z80 expects the interrupting device to put an instruction on the data bus which is then executed.
Mode 1	The Z80 performs a subroutine call to address 0038H.
Mode 2	The contents of the I register and a byte from the interrupting device are combined to form a 16 bit address. This points to two bytes in a table of interrupt routine addresses. The Z80 gets the address of the interrupt routine from the table and performs a subroutine call there.

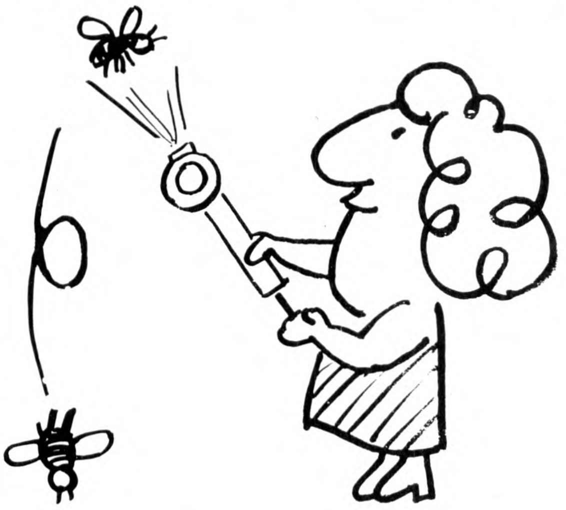
Interrupts can be enabled or locked out using the Enable Interrupt, EI, and Disable Interrupt, DI, instructions.

So why choose the Z80?

It is a fact that the CPU chip in your personal computer is likely to be one of the least expensive parts of the system. More expensive by far will be memory and peripherals so it makes sense to choose a CPU chip which will make best use of the system resources. The Z80 is undoubtedly the most powerful 8 bit microprocessor available to the amateur and is ideally suited to the task.

"Your honour, it says it isn't obliged to testify against itself."

8080 DEBUG



ROUTINE

Margaret Berg

This debug routine can:-

	Command: 1-G
Modify memory	1-G
Display memory	V
Print memory	P
Set quit points	Q
Go to a memory location	J

When debug is running the prompt character "/" appears at the beginning of a new line of the terminal. Any other letter than the allowable commands are ignored and the prompt character is redisplayed. Automatic formatting of lines is provided making the keying of spaces unnecessary.

MODIFY MEMORY

The number of bytes to be changed is entered followed by the hexadecimal address of the first byte followed by the data (in hexadecimal). From 1 to 16 bytes represented by 1 to G can be entered.

```
e.g.  No     Address    Data
      /3     1Ø ØØ      C3 ØØ 11
```

Display/Print Memory (See R.H. of page)

SET QUIT POINTS

When testing a program, execution can be forced to quit to permit examination of data areas etc. To set a quit point enter Q and A four digit hexadecimal address. The byte displaced is displaced on the screen and should be noted, so that it may be replaced when the quit is no longer required. When a quit point is reached the registers A, B, C, D, E, H and L are displayed followed by the address of the quit point. All are displayed in hexadecimal.

e.g. AA BB CC DD EE HH LL AD DR

The quit point must be set to replace an operation code. The code for the quit point is FF.

To use this facility the restart 7 address must branch to the address of routine "regs".

ie. Ø38 = C3 11 Ø1

e.g. /Q 11 Ø3 XX – where XX is the byte displaced

JUMP

Use this command to start testing from a specific address.

e.g. /J 1Ø ØØ

N.B.

This debug routine uses the internal registers and resets the stack pointer. Thus if a quit has been set in a subroutine, the return address will be lost. The jump instruction should normally jump into the main line program rather than into a subroutine.

DISPLAY/PRINT MEMORY

ENTER V OR P AS REQUIRED FOLLOWED BY THE FIRST TWO DIGITS OF THE ADDRESS. SIXTEEN LINES OF SIXTEEN BYTES ARE DISPLAYED GIVING THE ADDRESS OF EACH LINE, THE DATA IN HEXADECIMAL AND THE DATA IN ASCII.

```
E.G.    /V  1Ø

1ØØØ  ØØ ØØ ØØ ØØ  ØØ ØØ ØØ ØØ  ØØ F3 31 FF  Ø3 3E ØD 32           1 > 2
1Ø1Ø  CF Ø1 CD 4C  Ø1 3E ØA CD  4C Ø1 3E 2F  CD 4C Ø1 CD O ML > ML >/ML M
1Ø2Ø  39 Ø1 FE 56  CA 93 ØØ FE  51 CA Ø2 Ø1  FE 5Ø CA F2 9  VJ   QJ   PJ
1Ø3Ø  ØØ FE 4A CA  FE ØØ FE 3Ø  CA 49 ØØ FE  48 D2 49 ØØ    JJ   ØJI  HRI
1Ø4Ø  CD 72 Ø1 57  CD BE Ø1 CD  7F Ø1 77 23  15 C2 87 ØØ M  WM> M   # B
1Ø5Ø  C3 49 ØØ CD  7F Ø1 67 2E  ØØ 3E ØD CD  4C Ø1 3E 1Ø CI M      > ML >
1Ø6Ø  3D F5 3E ØA  CD 4C Ø1 7C  CD 8F Ø1 7D  CD 8F Ø1 F5 =>  ML   M   M
1Ø7Ø  CD 4A Ø1 CD  4A Ø1 7E CD  8F Ø1 23 7D  E6 ØF CA C9 MJ MJ  M  #   JI
1Ø8Ø  ØØ E6 Ø3 CA  BØ ØØ C3 B3  ØØ CD 4A Ø1  Ø6 ØF D1 1A    JØ C3 MJ   Q
1Ø9Ø  E6 7F FE 2Ø  DA DC ØØ FE  6Ø DA DE ØØ  3E 2Ø CD 4C      Z\   Z > ML
1ØAØ  Ø1 13 Ø5 F2  CF ØØ 3E ØD  CD 4C Ø1 F1  CA 49 ØØ C3      O > ML  JI C
1ØBØ  AØ ØØ CD 7F  Ø1 67 3E Ø8  32 CF Ø1 C3  97 ØØ CD BE    M   > 20 C  M>
1ØCØ  Ø1 E9 CD BE  Ø1 CD 4A Ø1  7E 36 FF CD  8F Ø1 C3 49    M> MJ   6 M  CI
1ØDØ  ØØ CD C7 Ø1  78 CD C7 Ø1  79 CD C7 Ø1  7A CD C7 Ø1  MG   MG   MG   MG
1ØEØ  7B CD C7 Ø1  7C CD C7 Ø1  7D CD C7 Ø1  E1 2B 7C CD  MG   MG   MG  + M
1ØFØ  3Ø 31 32 33  34 35 36 37  38 39 41 42  43 44 45 46 Ø123456789ABCDEF
```

```
2700                    ORG  0040H
0040 00          START NOP
0041 00                NOP                      SOME
0042 00                NOP                      SPARE SPACE
0043 00                NOP                      THAT CAN BE USED
0044 00                NOP                      TO INITIALISE
0045 00                NOP                      I/O DEVICES ETC
0046 00                NOP
0047 00                NOP
0048 00                NOP
0049             * *
0049             * *
0049 F3          BEGIN DI
004A 31 FF 03          LXI    M,STACK      SET UP STACK POINTER
004D             * * BEGIN A NEW LINE ON THE CONSOLE
004D 3E 0D             MVI    A,0DH
004F 32 BB 01          STA    DEVSW             SET VDU SWITCH
0052 CD 4C 01      CALL OUTT
0055 3E 0A             MVI    A,0AH
0057 CD 4C 01      CALL OUTT
005A 3E 2F             MVI    A,"/"
005C CD 4C 01      CALL OUTT
005F             * * GET A CHARACTER FROM THE VDU INTO "A" REGISTER
005F CD 39 01      CALL IN1
0062             * * DO WE WANT TO DISPLAY MEMORY
0062 FE 56             CPI    "V"
0064 CA 93 00          JZ     VDU               YES
0067             * * DO WE WANT TO SET A QUIT POINT
0067 FE 51             CPI    "Q"
0069 CA 02 01          JZ     QUIT              YES
006C             * * DO WE WANT TO PRINT MEMORY
006C FE 50             CPI    "P"
006E CA F2 00          JZ     PRINT             YES
0071             * * DO WE WANT TO START TESTING FROM A SPECIFIC ADDR
0071 FE 4A             CPI    "J"
0073 CA FE 00          JZ     JUMP              YES
0076             * * MAKE SURE 0 IGNORED
0076 FE 30             CPI    "0"
0078 CA 49 00          JZ     BEGIN
007B             * * DO WE WANT TO MODIFY MEMORY
007B FE 48             CPI    "H"
007D D2 49 00          JNC    BEGIN             IGNORE AS INVALID
0080             * *
0080             * * ROUTINE TO MODIFY MEMORY
0080             * * CHECK FOR VALID COMMAND AND CONVERT TO BINARY
0080 CD 72 01      CALL HEX
0083             * * SAVE COMMAND IN D
0083 57                MOV    D,A
0084             * * GET HEXADECIMAL ADDRESS OF THE FIRST BYTE TO BE
0084             * * CHANGED INTO H AND L REGISTER
0084 CD AA 01      CALL HADDR
0087             * * GET THE BYTE TO BE CHANGED INTO "A" REG
0087 CD 7F 01    NEXT1 CALL CON
008A             * * STORE BYTE IN POSITION IN MEMORY TO BE CHANGED
008A 77                MOV    M,A
008B             * * POINT TO NEXT POSITION IN MEMORY
008B 23                INX    H
008C             * * SUBTRACT 1 FROM NUMBER OF CHARACTERS TO BE CHANG
008C 15                DCR    D
008D C2 87 00          JNZ    NEXT1             CONTINUE PROCESSING
0090 C3 49 00          JMP    BEGIN             UNTIL VALUE ZERO
0093             * *
0093             * * ROUTINE TO DISPLAY ON THE VDU
0093             * * GET HIGH ORDER ADDRESS BYTE INTO "A" REG
0093 CD 7F 01    VDU   CALL CON
0096 67                MOV    H,A               SAVE IN "H" REG
0097             * * GENERAL PURPOSE ROUTINE
0097             * * FOR DISPLAY OR PRINTING PURPOSES
0097             * * SET LOW ORDER ADDRESS BYTE TO ZERO
0097 2E 00       DSPLY MVI    L,00
0099 3E 0D             MVI    A,0DH             ISSUE A
009B CD 4C 01      CALL OUTT
009E             * * SET UP THE NUMBER OF LINES TO DISPLAY/PRINT
009E 3E 10             MVI    A,10H             SET = 16
00A0             * *CAN OBVIOUSLY BE MODIFIED TO DISPLAY MORE OR LESS
00A0 3D          DSPL1 DCR    A             SUBTRACT 1 FROM NO LINES
00A1 F5                PUSH   M      SAVE NO OF LINES IN "A" REG
00A2             * * SPACE TO NEW LINE
00A2 3E 0A             MVI    A,0AH             ISSUE
00A4 CD 4C 01      CALL OUTT
00A7             * * DISPLAY HIGH ORDER ADDRESS BYTE IN HEX
00A7 7C                MOV    A,H
00A8 CD 8F 01      CALL HXOUT
00AB             * * DISPLAY LOW ORDER ADDRESS BYTE IN HEX
00AB 7D                MOV    A,L
00AC CD 8F 01      CALL HXOUT
00AF E5                PUSH   H             SAVE START ADDRESS OF LIN
00B0 CD 4A 01    INCR1 CALL SPACE
00B3 CD 4A 01    INCR  CALL SPACE
00B6             * * GET BYTE TO DISPLAY
00B6 7E                MOV    A,M
00B7 CD 8F 01      CALL HXOUT
00BA             * * INCREMENT TO NEXT BYTE TO DISPLAY
00BA 23                INX    H
00BB 7D                MOV    A,L      GET LOW ORDER ADDR BYTE
00BC             * * SEE IF 16 BYTES OF HEXADECIMAL DATA DISPLAYED
00BC E6 0F             ANI    0FH
00BE CA C9 00          JZ     CHAR              YES
00C1             * * MAKE SURE TWO SPACES EVERY FOURTH BYTE
00C1 E6 03             ANI    03H
00C3 CA B0 00          JZ     INCR1
00C6 C3 B3 00          JMP    INCR
00C9             * * DISPLAY THE DATA IN ASCII IF POSSIBLE
00C9 CD 4A 01    CHAR  CALL SPACE
00CC             * * SET NO OF BYTES TO DISPLAY
00CC 06 0F             MVI    B,0FH
00CE             * * RESTORE START ADDRESS OF LINE
00CE D1                POP    D                  IN "D" AND "E" REGS
00CF 1A          GETC  LDAX   D                 GET BYTE
00D0 E6 7F             ANI    7FH                SET OFF 80 BIT
00D2             * * SEE IF BYTE CAN BE DISPLAYED IN ASCII
00D2             * * IF NOT OUTPUT A SPACE
00D2 FE 20             CPI    20H
00D4 DA DC 00          JC     WRSP
00D7 FE 60             CPI    60H
00D9 DA DE 00          JC     WRCHR
00DC 3E 20       WRSP  MVI    A,20H              OUTPUT A SPACE
00DE CD 4C 01    WRCHR CALL OUTT
00E1             * * INCREMENT TO NEXT BYTE TO DISPLAY
00E1 13                INX    D
00E2             * * SUBTRACT 1 FROM NUMBER OF BYTES TO DISPLAY
00E2 05                DCR    B                 CONTINUE DISPLAYING
00E3 F2 CF 00          JP     GETC               UNTIL NUMBER = 0
00E6 3E 0D             MVI    A,0DH             ISSUE A
00E8 CD 4C 01      CALL OUTT
00EB F1                POP    M                 RESTORE NO OF LINES
00EC             * *                                INTO "A" REGISTER
00EC             * * CONTINUE DISPLAYING LINES UNTIL NO OF LINES
00EC             * * SET TO ZERO
00EC CA 49 00          JZ     BEGIN
00EF C3 A0 00          JMP    DSPL1
00F2             * *
00F2             * * ROUTINE TO DISPLAY ON THE PRINTER
00F2             * * GET HIGH ORDER ADDRESS BYTE INTO "A" REGISTER
00F2 CD 7F 01    PRINT CALL CON
00F5 67                MOV    H,A               SAVE IN "H" REGISTER
00F6 3E 08             MVI    A,PRDEV           SET
00F8 32 BB 01          STA    DEVSW              PRINTER SWITCH
00FB C3 97 00          JMP    DSPLY
00FE             * *
00FE             * * ROUTINE TO START TESTING FROM A SPECIFIC ADDRESS
00FE             * * GET ADDRESS TO GO TO INTO "H" AND "L" REGISTERS
00FE CD AA 01    JUMP  CALL HADDR
0101 E9                PCHL                     GO TO IT
0102             * *
0102             * * ROUTINE TO SET QUIT POINTS
0102             * * GET ADDRESS WHERE QUIT POINT TO BE SET
0102 CD AA 01    QUIT  CALL HADDR
0105 CD 4A 01      CALL SPACE
0108             * * GET BYTE TO BE REPLACED
0108 7E                MOV    A,M
0109             * * REPLACE BYTE WITH X"FF"
0109 36 FF             MVI    M,0FFH
010B             * * DISPLAY BYTE DISPLACED IN HEXADECIMAL
010B CD 8F 01      CALL HXOUT
010E C3 49 00          JMP    BEGIN
0111             * *
0111             * * THIS ROUTINE IS ENTERED WHENEVER A QUIT
0111             * * POINT IS ENCOUNTERED
0111             * * THE RESTART 7 ADDRESS MUST BE SET TO POINT TO
0111             * * THIS ADDRESS
0111             * * OUTPUT A SPACE THEN THE "A" REGISTER
0111 CD B3 01    REGS  CALL DSP
0114 78                MOV    A,B          OUTPUT A SPACE THEN
0115 CD B3 01      CALL DSP
0118 79                MOV    A,C          OUTPUT A SPACE THE
0119 CD B3 01      CALL DSP
011C 7A                MOV    A,D          OUTPUT A SPACE THEN
011D CD B3 01      CALL DSP
0120 7B                MOV    A,E          OUTPUT A SPACE THEN
0121 CD B3 01      CALL DSP
0124 7C                MOV    A,H          OUTPUT A SPACE THEN
0125 CD B3 01      CALL DSP
0128 7D                MOV    A,L          OUTPUT A SPACE THEN
0129 CD B3 01      CALL DSP
012C             * * GET ADDRESS OF BYTE AFTER THE QUIT POINT
012C E1                POP    H            INTO "H" AND "L" REGS
012D 2B                DCX    H            GET ADDRESS QUIT POINT
012E 7C                MOV    A,H          OUTPUT THE
012F CD B3 01      CALL DSP
0132 7D                MOV    A,L            THE
0133 CD B3 01      CALL DSP
0136 C3 49 00          JMP    BEGIN
0139             * ***********************************************
0139             * *     SUBROUTINES                             *
0139             * ***********************************************
0139             * * TYPICAL CHARACTER INPUT ROUTINE
0139             * * BIT 4 MUST GO HIGH
0139             * * WHEN A KEYBOARD CHARACTER HAS BEEN DEPRESSED
0139             * * RETURNS THE ASCII CHARACTER IN "A" REGISTER
0139 DB 81       IN1   IN     STIND             READ DEVICE STATUS
013B             * *
013B E6 10             ANI    10H               TEST READY BIT
013D CA 39 01          JZ     IN1               LOOP IF NOT READY
0140 CD 5C 01      CALL WAIT
0143 DB 01             IN     INDEV             GET THE CHARACTER
0145 E6 7F             ANI    07FH              SET OFF PARITY BIT
0147             * *    JMP    OUTT               ECHO CHARACTER BACK
0147 00                NOP
0148 00                NOP
0149 C9            RET
014A             * * TYPICAL CHARACTER OUTPUT ROUTINE  BIT 4 MUST GO
014A             * * LOW WHEN THE OUTPUT DEVICE IS READY FOR A CHAR
014A             * * EXPECTS AN ASCII CHARACTER TO OUTPUT IN "A"
014A             * * REGISTER ON ENTRY
014A 3E 20       SPACE MVI    A,20H             OUTPUT A SPACE
014C F5          OUTT  PUSH   M                 SAVE "A" REGISTER
014D             * * GET BYTE TO SEE WHETHER DEVICE TO BE OUTPUT TO
014D             * * IS A VDU OR PRINTER
014D 3A BB 01          LDA    DEVSW
0150 FE 08             CPI    PRDEV             IS IT A PRINTER ?
0152 CA 64 01          JZ     OUTT1             YES
0155 CD 5C 01      CALL WAIT
0158             * * GET CHARACTER TO BE DISPLAYED
0158 F1                POP    M
0159 D3 01             OUT    INDEV
015B 00                NOP
015C             * * ROUTINE TO WAIT FOR DEVICE TO BECOME AVAILABLE
015C DB 81       WAIT  IN     STIND             READ DEVICE STATUS
015E E6 10             ANI    10H              TEST READY BIT
0160 C2 5C 01          JNZ    WAIT             ;LOOP UNTIL READY
0163 C9            RET
0164             *,* OUTPUT TO THE PRINTER
0164 DB 88       OUTT1 IN     PRST              READ PRINTER STATUS
0166 E6 10             ANI     10H               TEST READY BIT
0168 C2 64 01          JNZ    OUTT1             LOOP UNTIL READY
016B             * * GET CHARACTER TO BE PRINTED
016B F1                POP     M
016C D3 08             OUT    PRDEV            OUTPUT TO PRINTER
016E C9            RET
016F             * *
016F             * * ROUTINE TO GET A CHARACTER FROM THE SCREEN
016F             * * AND THEN CHECK IT FOR A VALID HEXADECIMAL
016F             * * NUMBER - IF AN INVALID CHARACTER IS FOUND
016F             * * THEN THE PROMPT CHARACTER IS REDISPLAYED AND
016F             * * DEBUG WAITS FOR ANOTHER VALID COMMAND
016F CD 39 01    INN   CALL IN1
0172             * * CHECK FOR A VALID HEXADECIMAL VALUE
0172 D6 30       HEX   SUI    30H
0174 FE 0A             CPI    0AH
0176 D8                RC
0177 D6 07             SUI    07H
0179 FE 11             CPI    11H
017B D8                RC
017C             * * INVALID - SO REDISPLAY PROMPT
017C C3 49 00          JMP    BEGIN
017F             * *
017F             * * THIS ROUTINE GETS 2 CHARACTERS FROM
017F             * * THE SCREEN AND CONVERTS THEM INTO ONE BYTE
017F CD 4A 01    CON   CALL SPACE
0182             * * GET A CHARACTER FROM THE SCREEN
0182 CD 6F 01      CALL INN
0185 07                RLC                     MOVE
0186 07                RLC                     BOTTOM 4 BITS
0187 07                RLC                      TO TOP 4 BITS
0188 07                RLC
0189 4F                MOV    C,A          SAVE IN "C" REGISTER
018A             * * GET NEXT CHARACTER
018A CD 6F 01      CALL INN
018D B1                ORA    C            MAKE INTO 1 BYTE
018E C9            RET
018F             * *
018F             * * ROUTINE TO DISPLAY ONE BYTE ON THE SCREEN
018F             * * AS A HEXADECIMAL VALUE
018F F5          HXOUT PUSH   M                 SAVE CHARACTER
0190 0F                RRC                    MOVE
0191 0F                RRC                     TOP 4
0192 0F                RRC                        BITS TO
0193 0F                RRC                     BOTTOM 4 BITS
0194             * *CONVERT TO AN ASC11 REPRESENTATION
0194 CD A1 01      CALL CONVT
0197 CD 4C 01      CALL OUTT
019A F1                POP    M                 RESTORE CHARACTER
019B             * * CONVERT TO AN ASC11 REPRESENTATION
019B CD A1 01      CALL CONVT
019E             * * OUTPUT CHARACTER AND RETURN FROM THERE
019E C3 4C 01          JMP    OUTT
01A1             * *
01A1             * * ROUTINE TO CONVERT A HEXADECIMAL VALUE TO
01A1             * * ASC11
01A1 E6 0F       CONVT ANI    0FH               SET OFF TOP 4 BITS
01A3             * * MAKE A TO F CAUSE A CARRY
01A3 C6 90             ADI    90H
01A5             * * DECIMAL ADJUST
01A5 27                DAA
01A6             * * ADD IN CARRY AND ADJUST TOP 4 BITS
01A6 CE 40             ACI    40H
01A8 27                DAA
01A9             * * RETURN WITH ASCII CHARACTER IN "A" REGISTER
01A9 C9            RET
01AA             * *
01AA             * * ROUTINE TO GET THE HEXADECIMAL ADDRESS
01AA             * * INPUT FROM THE SCREEN INTO "H" AND "L" REGISTERS
01AA             * *
01AA CD 7F 01    HADDR CALL CON
01AD 67                MOV    H,A    INTO "H" REG
01AE CD 7F 01      CALL CON
01B1 6F                MOV    L,A     INTO "L" REGISTER
01B2 C9            RET
01B3             * * ROUTINE TO OUTPUT A SPACE FOLLOWED BY THE "A"
01B3             * * REGISTER AS A HEXADECIMAL VALUE
01B3 F5          DSP   PUSH   M                 SAVE  "A" REG
01B4 CD 4A 01      CALL SPACE
01B7 F1                POP    M                 RESTORE "A" REG
01B8             * * OUTPUT THE "A" REGISTER IN HEXADECIMAL
01B8             * *  AND RETURN FROM THERE
01B8 C3 8F 01          JMP    HXOUT
01BB             * * SWITCH TO SEE IF DISPLAY OR PRINT REQUIRED
01BB             * * SET = 0D FOR DISPLAY
01BB             DEVSW DS   1
01BC             * *
01BC             STIND EQU  81H
01BC             INDEV EQU  01H
01BC             PRST  EQU  88H
01BC             PRDEV EQU  08H
01BC             * *
01BC             STACK EQU  03FFH
01BC             * *
01BC             * *
```

NEXT ISSUE

Full Evaluation of the Apple by John Coll & Charles Sweeten

PRINTED CIRCUITS and HARDWARE

Comprehensive range Constructors' Hardware and accessories.
Selected range of popular components.
Full range of ETI printed circuit boards.
normally ex-stock, same day despatch at competitive prices.
P.C. Boards to individual designs.
Resist-coated epoxy glass laminate for the d.i.y. man with full processing instructions (no unusual chemicals required).

Send 15p for catalogue.

RAMAR CONSTRUCTOR SERVICES
Masons Road, Stratford-on-Avon, Warwicks.
Tel: 4879

PERSONAL COMPUTING MAGAZINES FOR HOBBYIST AND BUSINESS: Dr. Dobbs, Personal Computing, Creative Computing, ROM — £1.50 (some back issues — £1.25). Byte, Interface Age, Kilobaud — £2.00 (some back issues — £1.50). P&P on any one magazine — £0.45. Microcomputing books also in stock. INTERAM Computer Systems Ltd., 59 Moreton Street, London S.W.1. (01-834 0261/2733).

ADVERTISEMENT

Come to one of our free introductory courses every Monday in June and July between 2p.m. and 4p.m. at our new showroom in the heart of London.

Entitled "An Introduction To The Use of Microcomputer Systems" they demonstrate the use of systems costing less than £4,000 in applications such as order entry, invoicing, and word processing, and show the beginner how easy it is to program in BASIC.

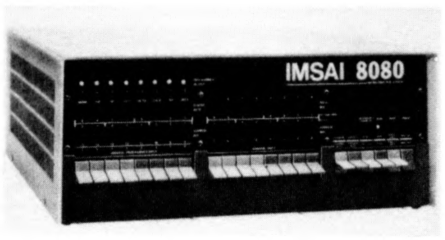
Details from: COMPUTER WORKSHOP, 38 Dover St., London W.1. Tel: 01-491 7507

BASIC PROGRAMS — FULLY TESTED

NOUGHTS-AND-CROSSES	£4.50	GLADIATOR	£4.50
GOLF	£4.00	CRICKET	£4.50
ROULETTE	£4.00	BATTLESHIPS	£3.50
LUNAR-LANDER	£2.00	DECAY	£2.00
SLOT MACHINE	£2.00	GUESS	£2.00

COMING SOON . . .
MASTERMIND, AMERICAN FOOTBALL, GOLD RUSH, HOT ROD, ROBIN HOOD, BIO-RHYTHMS NIM, SPACE FLIGHT, MONEY
FOR MORE DETAILS AND ORDERS, SEND S.A.E.+P.O. OR CHEQUE TO C. W. THOMAS, 15 CAEMAWR ROAD, MORRISTON, SWANSEA, SA6 7DX.
n.b. NO S.A.E. NO PROGRAM

Advertisement
Late Flash . . . *Lowered Prices* announced by AIRAMCO . . . 1702A 256 x 8 UVEPROM £4.97 • 2708A 1024 x 8 UVEPROM £10.00 • 8080A CPU £10.00 (*see our ad. elsewhere in this issue*).
We are now U.K. Distributors for Jade Computer Products. Come and see us at the D.I.Y. Computer Show, West Centre Hotel, London S.W.6. June 22-24. **Stand 28. Jade** will be demonstrating their equipment and systems including KIM 1, 2 & 3.
Airamco Ltd., 30 Witches Linn, Ardrossan, Ayrshire KA22 7BR.
Tel: 0294-65530



COMPUTER MART

WE ARE ONLY A SHORT TIME AWAY!

As the jet flies, we may be your closest computer store, and we carry all the major products sold in the United States. When you come to New York (as everyone does, sooner or later) stop in and see our display of computers, peripherals, parts, software and books. Take advantage of our friendly advice, service and low prices.

We sell and service the best products: PROCESSOR TECHNOLOGY, ALPHA MICRO SYSTEMS, SOUTHWEST TECHNICAL PRODUCTS, APPLE II, TDL, CROMENCO, S.D. COMPUTER PRODUCTS, NORTH STAR, REX, MICROMATION, KIM, RCA COSMAC VIP, HAZELTINE, and CENTRONICS.

STAN VEIT, Storekeeper

VISA *MASTER CHARGE* **COMPUTER MART OF NEW YORK, INC.**

118 Madison Avenue • New York, New York 10016 • 212-686-7923

MK14-the only low-cost keyboard-addressable microcomputer!

The new Science of Cambridge MK14 Microcomputer kit

Go
Mem
Abort
a
7
8
9
b
4
5
6
c
1
2
3
d
Term
0
f
e
MK14

Just £39.95 (+ £3.20 VAT, and p&p)

The MK14 National Semiconductor Scamp based Microcomputer Kit gives you the power and performance of a professional keyboard-addressable unit – for less than half the normal price. It has a specification that makes it perfect for the engineer who needs to keep up to date with digital systems or for use in school science departments It's ideal for hobbyists and amateur electronics enthusiasts, too.

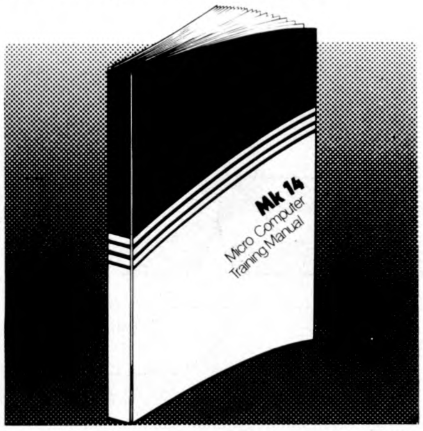
But the MK14 isn't just a training aid. It's been designed for practical performance, so you can use it as a working component of, even the heart of, larger electronic systems and equipment.

MK14 Specification

★ Hexadecimal keyboard
★ 8-digit, 7-segment LED display
★ 512 x 8 Prom, containing monitor program and interface instructions
★ 256 bytes of RAM
★ 4MHz crystal
★ 5V stabiliser
★ Single 6V power supply
★ Space available for extra 256 byte RAM and 16 port I/O
★ Edge connector access to all data lines and I/O ports

Free Manual

Every MK14 Microcomputer kit includes a free Training Manual. It contains operational instructions and examples for training applications, and numerous programs including math routines (square root, etc) digital alarm clock, single-step music box, mastermind and moon landing games, self-replication, general purpose sequencing, etc.



Designed for fast, easy assembly

Each 31-piece kit includes everything you need to make a full-scale working microprocessor, from 14 chips, a 4-part keyboard, display interface components, to PCB, switch and fixings. Further software packages, including serial interface to TTY and cassette, are available, and are regularly supplemented.

The MK14 can be assembled by anyone with a fine-tip soldering iron and a few hours' spare time, using the illustrated step-by-step instructions provided.

Tomorrow's technology – today!

"It is not unreasonable to assume that within the next five years . . . there will be hardly any companies engaged in electronics that are not using micro-processors in one area or another."

Phil Pittman, Wireless World, Nov. 1977.

The low-cost computing power of the microprocessor is already being used to replace other forms of digital, analogue, electro-mechanical, even purely mechanical forms of control systems.

The Science of Cambridge MK14 Standard Microcomputer Kit allows you to learn more about this exciting and rapidly advancing area of technology. It allows you to use your own microcomputer in practical applications of your own design. And it allows you to do it at a fraction of the price you'd have to pay elsewhere.

Getting your MK14 Kit is easy. Just fill in the coupon below, and post it to us today, with a cheque or PO made payable to Science of Cambridge. And, of course, it comes to you with a comprehensive guarantee. If for any reason, you're not completely satisfied with your MK14, return it to us within 14 days for a full cash refund.

**Science of Cambridge Ltd,
6 Kings Parade,
Cambridge,
Cambs., CB2 1SN.
Telephone: Cambridge (0223) 311488**

To: Science of Cambridge Ltd,
6 Kings Parade, Cambridge,
Cambs., CB2 1SN.

Please send me an MK14 Standard Microcomputer Kit. I enclose cheque/ Money order/PO for £43.55 (£39.95 + 8% VAT and 40p p&p).
Allow 21 days for delivery.

Name ______________________
Address (please print) ______________________
____
____
____
____

Science of Cambridge

PUTTING BITS IN THEIR PLACE

A simple 2708 Programmer

David Goadby

At the time of writing the 2708 is available for less than £11 in single units. The popularity of a chip is invariably in inverse proportion to its cost, so the 2708 is rapidly becoming an amateur standard, at least until something cheaper comes along!

Before describing the 2708 programmer an introduction to the 2708 is given for newcomers to personal computing. Experienced cybernuts jump a few paragraphs!

The 2708 is a non-volatile, 8K bit EPROM. Let's translate this: 1) It retains its contents even after power is removed — it's non-volatile. 2) It can hold 8192 bits of data organised as 1024 bits x 8, giving us 1K bytes, since 1024 is written 1K and 8 bits are called a byte. 3) EPROM (sometimes incorrectly shortened to PROM) stands for Eraseable Programmable Read Only Memory.

The package itself is a 24 pin ceramic and the memory array is visible through a quartz lid. The quartz lid is for *erasing* the chip using ultraviolet light. (More about erasure later).

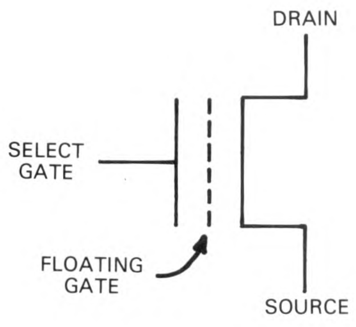
The 2708 is simply a large quantity of storage cells in an array. Figure 1 shows a storage cell and essentially it is a FET (field effect transistor) with a 'floating gate'. The gate floats because there are no electrical connections to it, and it acts like a small capacitor, retaining whatever charge it has for ten years (manufacturer's spec.) In the *erased* state this gate is discharged; in the *programmed* state it has a charge sufficient to inhibit the switching of the cell.

The array of the cells is *addressed* using rows and columns and each addressed cell will have electrical inputs to the drain and gate. If the floating gate is charged the output will be a ø. Logically, the 2708 will erase to all 1s, and in fact it does. Figure 2 shows the physical construction of the cell.

Programming the cell is by pumping up the charge on the floating gate to a value greater than the threshold in Figure 3. The specifications for the programming and erasure of the 2708 should be followed faithfully since, as Figure 3 clearly shows, there is the possibility of uncertain data close to the threshold band.

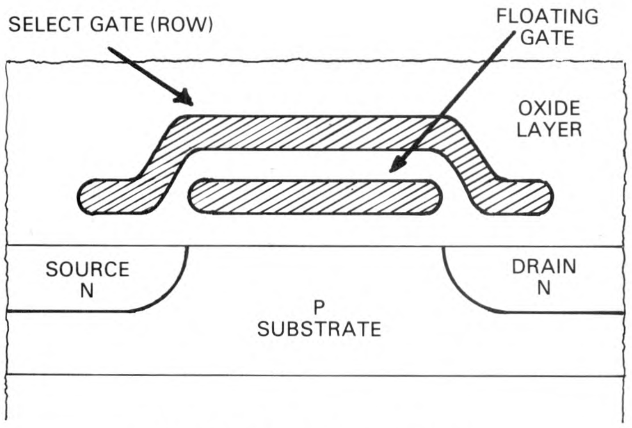
Erasure is achieved by using ultraviolet light of 2537 angstroms wavelength. The U.V. light overcomes the energy barrier and "short circuits" the gate to the substrate.

If you don't have a U.V. tube in the house don't worry. The normal sized health lamp will do the job effectively in about twenty minutes. Keep the chip 12" from the lamp. Should the idea of a suntan not appeal to you then a germicidal type U.V. tube is available from good electrical wholesalers. But erasure takes longer, about 30 minutes to 1 hour.



2708 STORAGE CELL

FIG. 1



PICTORIAL REPRESENTATION

FIG. 2

But it's essential to remember that ultraviolet light is a *dangerous* radiation and continued exposure to it may damage the eyes. So you must use the special goggles with the health lamp, and if you make your own erases put a safety switch in the lid to cut the supply when inserting or removing 2708s.

Nearly there! The programming sequence for 2708s is *not* like that for 1702s (another type of EPROM).

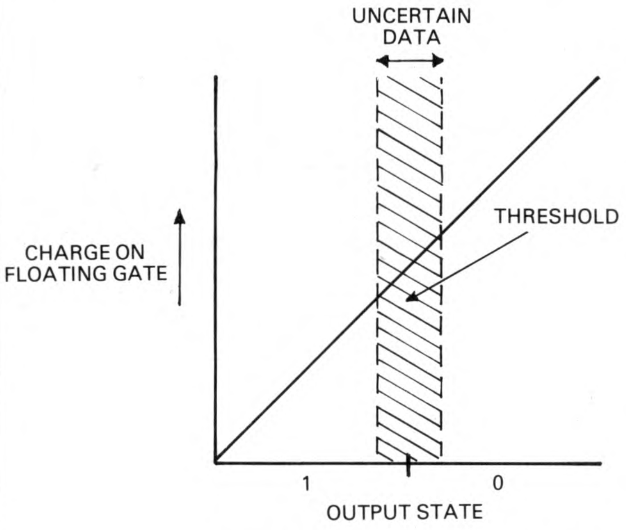
The sequence consists of programming each location with a single pulse and sequencing to the next location until all 1024 bytes have been programmed; this being one program loop. The loop is repeated a minimum of 100 times, using this simple formula:

$N \times T > 100$ ms; where $0.1 < T < 1.0$ milliseconds,
N = number of loops, T = programming pulse width.

A good choice for T is 0.4 ms, since then N=255 loops, a convenient number to count with 8 bits.

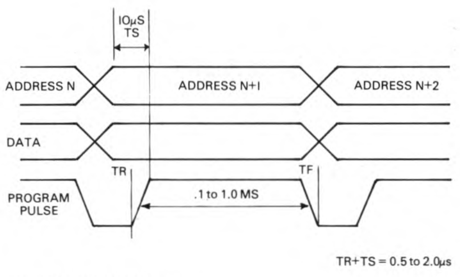
No attempt should be made to program single locations since transients in the array can alter cells already programmed and uncertain data may be generated.

For those interested, a simple timing chart is shown in Figure 4. The only critical value is the set up time TS. This should be a minimum 10 microseconds before the program pulse.



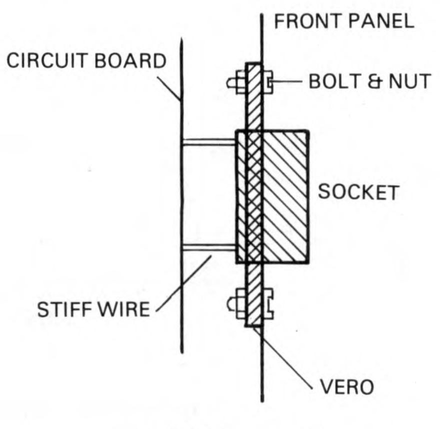
CHARGE VERSUS OUTPUT STATE

FIG. 3



TIMING DIAGRAM (WRITE)

FIG. 4



CONSTRUCTION IDEA

FIG. 5

THE PROGRAMMER

The design for the programmer is very simple and may be adapted in a number of ways to suit a number of different systems. In the form presented there is an 8 bit data bus input and output and handshaking lines. The prototype works with a MSI 6800 system and plugs directly into a PIA (peripheral interface adapter).

A printed circuit is under consideration and if demand warrants it will be made available through *PCW*. No power supplies are shown as the demand is moderate and they can be taken from the host microcomputer. The supply that may cause problems is the 26 volts. The current load is 3mA only and a voltage doubler or miniature transformer and rectifier will supply this easily. One note about the 26 volt supply: I found that 26 volts ±2% was very satisfactory. But the purists among my readers may prefer to make the supply variable and then using an oscilloscope adjust the programming pulse to 25 volts swing, which is the specified level; ie voltage high — voltage low = 25 volts.

To communicate with the host processor I have used a couple of 74125 buffer ICs (IC1 & 2). The output of these ICs goes tristate when a high level exists on the tristate pins. During programming the buffers are enabled and data in is applied to the 2708 data pins. This data is also available at "data out" at this time. So you can check all the wiring of the data lines by inputting a bit at a time and reading the output to see if it corresponds to the same bit.

Other than a power switch, to allow changing of 2708s without powering off the host system, the only other switch is a read/program switch. The switch in read mode will, as already stated, tristate the buffers, and will also disable the program pulse circuitry. The transistor TR1 is also switched on to make the 2708 pin 20 at ground level, this being the selected state.

The switch, when it is in program mode, enables the input buffers, allows program pulses via IC*6* and sets pin 20 of the 2708 to +12 volts, this being the level required for programming. If pin 20 is +5 volts the 2708 is deselected, ie tristate outputs, but this condition doesn't occur in the design, though possibly a multiple EPROM programmer could be made with only the device currently programmed being enabled.

No output buffering is used in the unit as 2708s have a single TTL drive capability and most sytem data inputs are MOS.

The address counters ICs 3, 4, 5 are binary counters and the outputs are fed directly to the 2708 as the loading is very light.

The line and reset is a line controlled by the host system. When a plus level exists on the line the address is reset to 0: normally a single pulse from the system is all that is required to reset. The line out of the counter chain marked +check is for the system and this line goes high after the 1024th pulse. The software should check that this line goes high *only* at this time. Any other time is an error in the counter chain, or maybe you forgot to reset the chain.

Now for the tricky bit. The line and step is a system line and a positive pulse is sent from the host to start the timer IC7. The negative edge of the pulse triggers IC7, a 74123, and the pulse width set by C1, R1 is 0.4 ms. This pulse is used for the program pulse. The negative edge of the pulse triggers the second half of IC7 and R2, C2 set the pulse width to 10 micro sec, this allowing the hardware to meet the requirements of TS in the timing specifications.

The address counters are stepped on the falling edge of the second pulse and this 10 microsec pulse is the handshake pulse for the host system, telling it that the cycle has completed.

The programming pulse is passed IC6 to TR2 which controls TR4 and TR3. TR4 passes the 26 volt pulse to the program pin and TR3 is a sink transistor and needed to sink 20mA from the program pin. This transistor cannot be omitted! The 100 ohm resistor and 0.002 mfd capacitor create the rise and fall delays TR and TF of about 0.7 microseconds. These components again cannot be omitted as the delay is necessary to prevent transient pulses creating incorrect data in the cell arrays.

Although not shown in the circuit a liberal sprinkling of 0.1 mfd capacitors are used, mainly around the 74123 between +5v and ground, this being a normal TTL noise precaution, although perfect operation will probably be obtained without them.

Construction:

The prototype was built on a Tandy© prototyping card. This has 1 matrix pads and voltage rails on it. Figure 5 shows the method I used to mount the unit. The socket, which should be a zero force insertion type, is soldered to a piece of veroboard. The circuit board is attached to the vero using stiff tinned copper wire, and the whole unit is mounted on the front panel using four screws and nuts.

Don't use cables with high capacity for the data in and out lines. I used 18" of Scotchflex. . .I had odd problems when I first tried three feet of surplus cable.

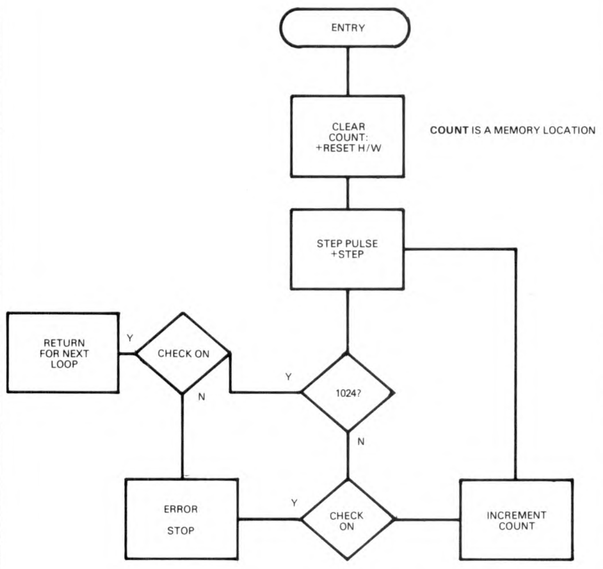
Testing:

You can apply +5 volts only and see if data applied to data inputs, in program mode, gets to the output pins and the 2708 data pins correctly. The system can then pulse the +step line and you should see a handshake pulse of approximately 10 microsec. and at the base of TR2 a 0.4 ms pulse. The +26 volts can then be applied to see if the program pulse appears correctly. Check all voltages on the 2708 socket and if you have an oscilloscope check the address counters. The timers are about right if the programming cycle takes about 1½ minutes.

You will need some software to use the device to its full capabilities. A copy of the software is available from me at a cost of £2.50 on cassette that you supply. (Please state 1) where the PIA address is. 2) where you want the program to reside; size is approximately 1K plus 1K data area).

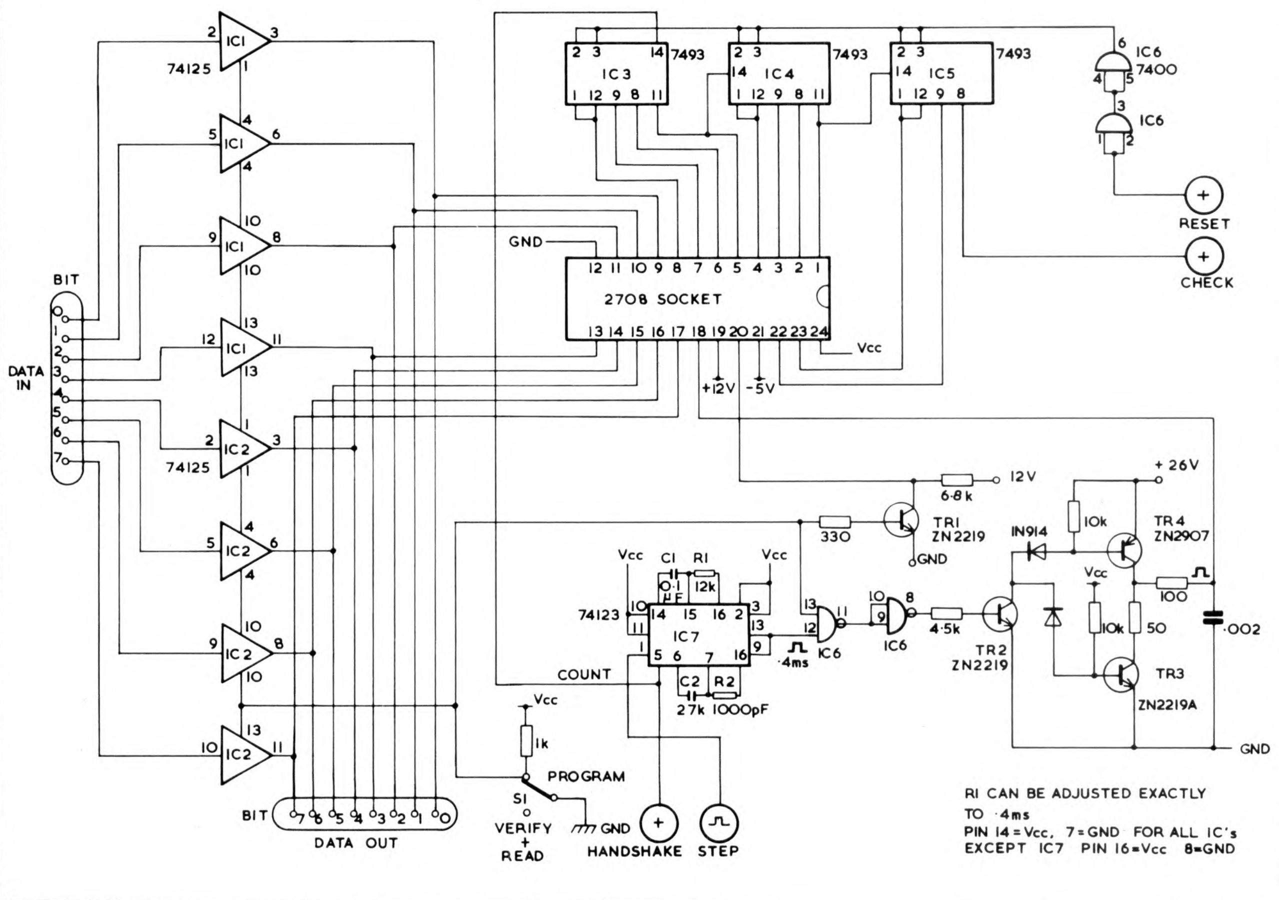
To help you write your own software, a simple flowchart for programming is drawn — Figure 6.

Finally, the programmer I have described is in regular use and during its life to date (seven months) has never once failed. I hope it serves you as well. The modifications for the 2758 and 2716 are in hand and may later be published; but at the moment 2716s are just a little bit too expensive!

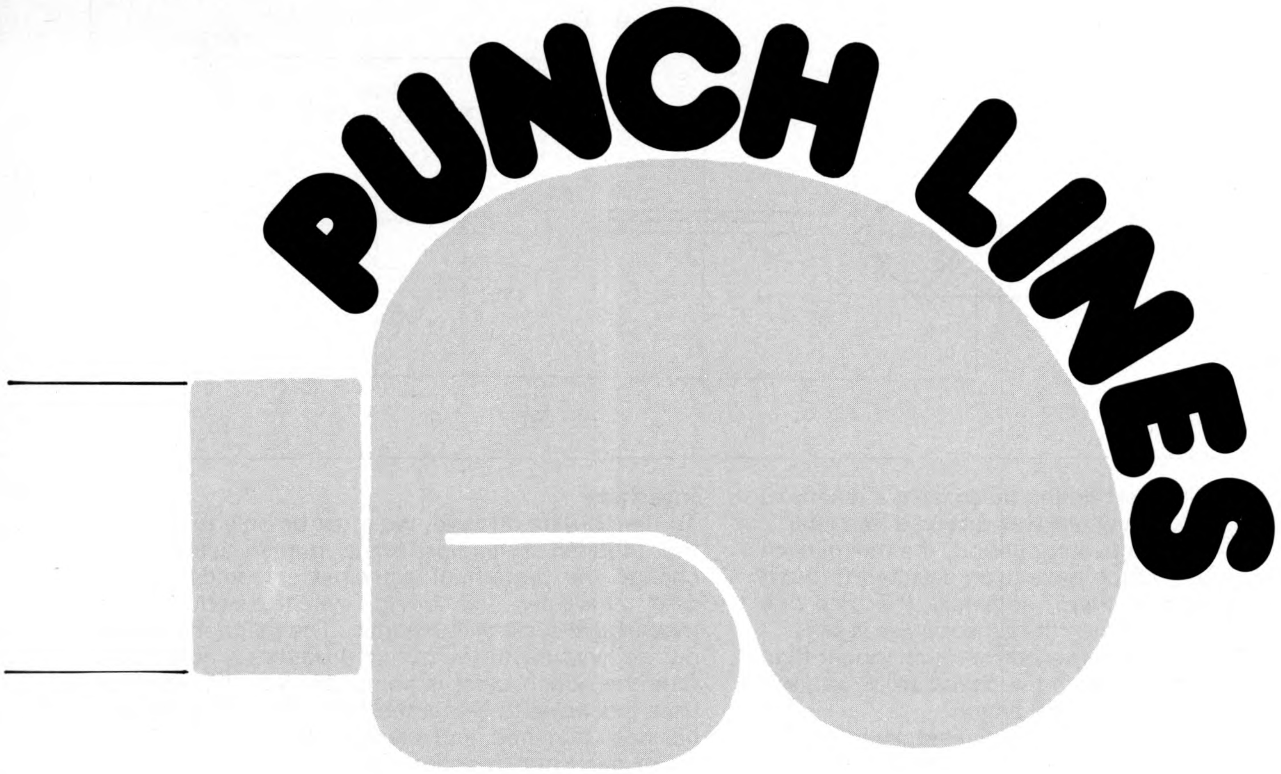


EXAMPLE OF SINGLE LOOP SOFTWARE FOR CONTROLLING HARDWARE

FIG. 6



RI CAN BE ADJUSTED EXACTLY TO ·4ms
PIN 14=Vcc, 7=GND FOR ALL IC's EXCEPT IC7 PIN 16=Vcc 8=GND



Interfacing a Westrex Punch to a 6800 mpu System

Mark Colton

A paper tape punch is a useful piece of equipment to have in an MPU system. Paper tape is a widely used medium, and provides a fairly indestructible copy of any saved programs and data. Teletype paper tape punches are necessarily slow (10 c.p.s.), and the saving of large programs is unfortunately tedious. A faster machine to do this job is therefore desirable.

The Westrex punch mechanism is the obvious choice when looking for a fast PTP, used by many large computer departments. With careful hunting used ones may be found cheaply . . .

This article describes the interfacing of a Westrex punch mechanism to a 6800 MPU system (although the interface is suitable for all CPU's currently available), with comprehensive software for running the punch to follow. The punch unit — distinguished from the mechanism — consists of the mechanism itself, power supply, solenoid drivers, and interface board. See Fig 1 for the unit block diagram.

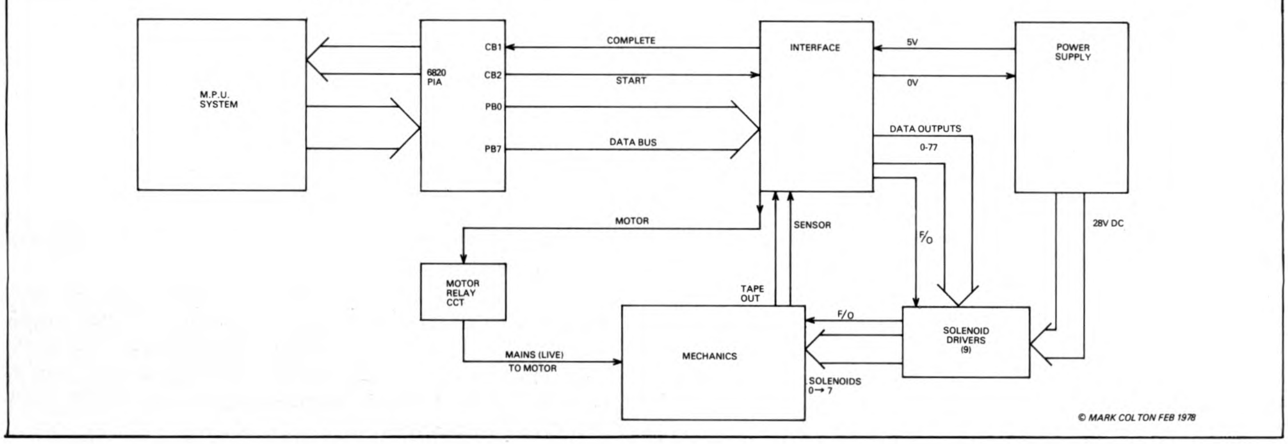
The power supply circuit is shown in Fig 2. The 20V 10A transformer produces 28V d.c. when rectified and smoothed by the diodes and capacitors shown. A 5V supply for the interface board is derived from the transistor-zener diode circuit.

The Punch Mechanism

The punch mechanism punches eight data channels plus a feed out hole on blank paper tape, at 110 c.p.s. The machine is driven by a motor geared down to drive a flywheel at 110 r.p.s. The period of revolution of the driving flywheel is thus 9ms. On each rotation any or all of nine knives are driven through the tape to produce a hole. The feed out hole is always punched, but any of the eight data channels may or may not have a hole punched at a particular point.

To each knife is attached a solenoid, and a particular knife may be selected to punch (be activated) by passing a current through its solenoid,

Fig. 1 *Unit clock diagram*



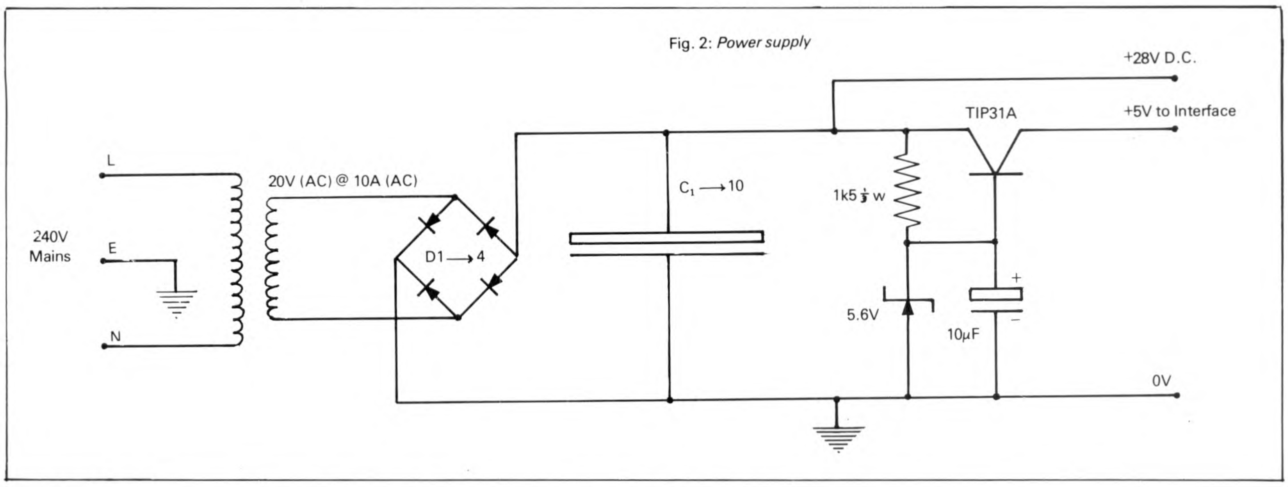


Fig. 2: *Power supply*

for 4.5ms after the last timing pulse from a magnetic sensor on the main flywheel was received (see later).

After this 4.5ms activating period, the mechanism drives the knives which have been selected through the tape, withdraws them, advances the tape one character length, and then the sensor pulse is sent.

If the solenoids are activated for much longer than 4.5ms, the mechanics latch the character in, and will duplicate it on the next punch action.

By changing the pattern of activated solenoids for each rotation of the flywheel, therefore, one changes the pattern of holes punched on the tape. The computer has access to each of the solenoids (via the PIA — interface — solenoid drivers) and the byte which it places on the data bus is punched in exactly the same format on tape:

Action No.	**Byte**	**Tape (H denotes a hole)**								
Bit No:	76543210	7	6	5	4	3	F/O	2	1	0
1	01010101		H		H		H		H	H
2	10101010	H		H		H	H		H	
3	11010011	H	H		H		H		H	H

(Action No. refers to rotation of flywheel)

The feed out hole (F/O) is used by paper tape readers as a strobe hole and is therefore always present with a character.

When the feed out solenoid on the punch unit is activated, it causes the tape to be fed through the unit at 110 c.p.s. (the tape is advanced one character length per punch action), and it also causes the feed out hole to be punched on tape.

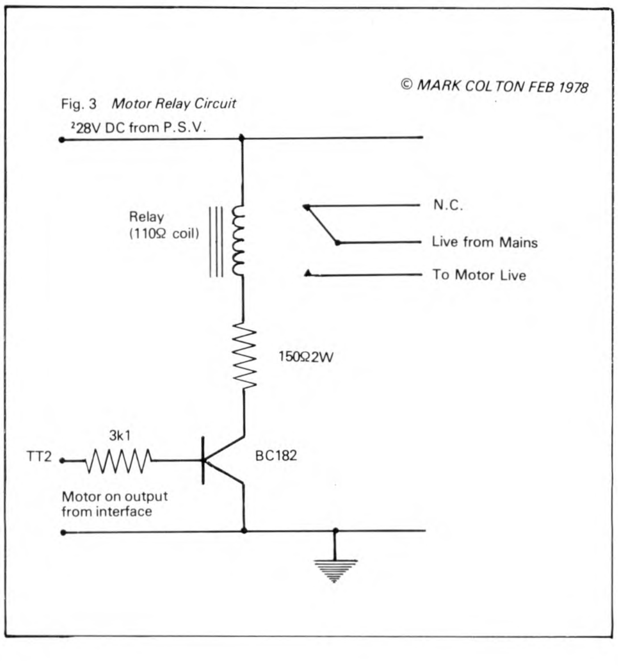


Fig. 3 *Motor Relay Circuit*

Interface

To punch data on tape, we must be able to activate the required solenoids for a punch action, and change the pattern of activated solenoids for the next character, all being synchronised to the mechanical cycle of the punch. The magnetic sensor on the flywheel of the punch produces a pulse each time the punch cycle is complete, and the interface uses this pulse to signal the MPU that the character has been punched, and it is ready for the next one.

For each punch action (i.e. every 9ms) the MPU must be able to write a character to the interface — via the PIA — at the correct instant. Similarly the punch must know when the next character from the MPU is present on the data bus. The START and COMPLETE lines enable this "handshake" to go on.

When the punch has finished punching a character, the interface sends a positive "complete" pulse to the PIA, to signal that it is ready for the next character. The MPU then sends the next character — on the data bus — together with a negative pulse on the START input — the latter telling the interface that the next character is ready on the bus.

On reception of the start pulse, the interface clocks the data present on the data bus to the solenoid drivers (via the data latches 0 to 7). The respective solenoids are then energised for 4.5ms (the activating time). The current is cut off, and the interface waits until the pulse from the magnetic sensor on the motor flywheel is detected.

The interface interprets this as a timing signal, and

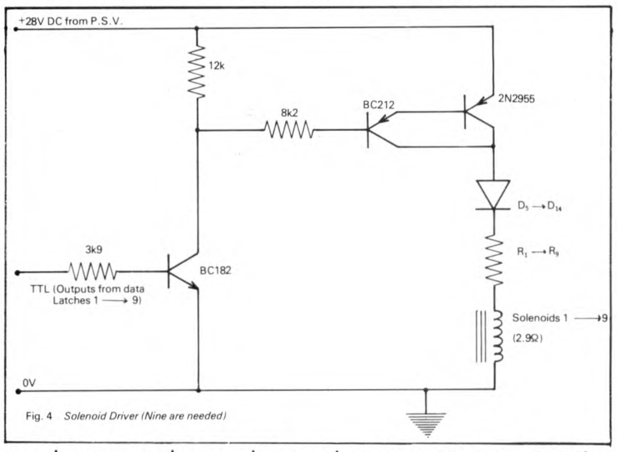
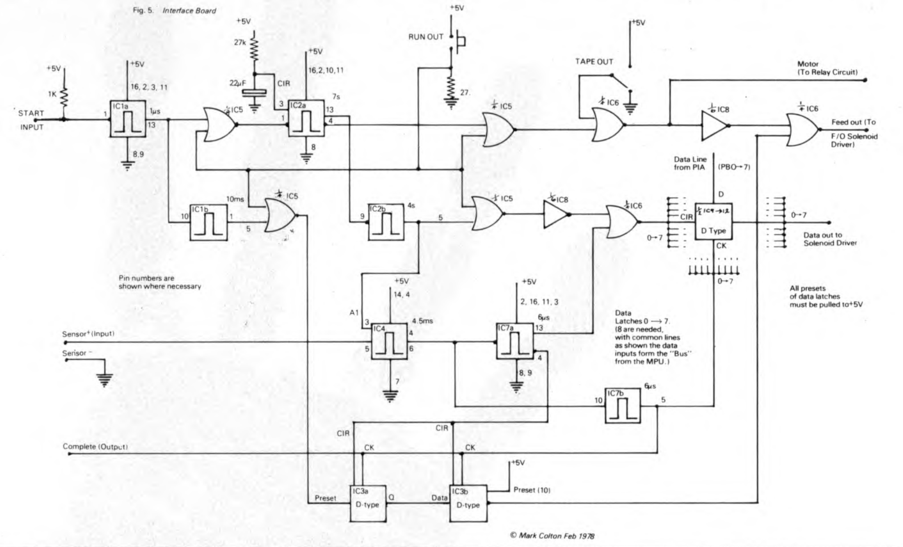


Fig. 4 *Solenoid Driver (Nine are needed)*

sends a complete pulse to the computer to get the next character.

The interface also controls the motor (Fig 3), and allows RUN OUT of a desired length of clear tape (except for F/O hole), simply by pushing the RUN OUT button. If the unit runs out of tape, the motor will not run until more is threaded — the TAPE OUT switch prevents this.

Fig. 5. *Interface Board*


START INPUT
RUN OUT
TAPE OUT
Motor (To Relay Circuit)
Feed out (To F/O Solenoid Driver)
Data Line from PIA (PB0→7)
Data out to Solenoid Driver
All presets of data latches must be pulled to +5V
Pin numbers are shown where necessary
Sensor+ (Input)
Sensor −
Data Latches 0 → 7 (8 are needed, with common lines as shown the data inputs form the "Bus" from the MPU.)
Complete (Output)
IC3a D-type
IC3b D-type
Preset (10)
© Mark Colton Feb 1978

The interface handles all the timing and logic, but the TTL output levels it produces do not supply enough current to activate the solenoids, so a Darlington pair (Fig 4) amplifies the current to activate the solenoids whenever instantaneously required.

If the motor is off, and a start pulse received, then the motor is started, and there is a 4 second run-up-to-speed delay before the character on the data bus (sent by the MPU with the start pulse) is punched, and a complete pulse sent. The motor remains on for 7 seconds after the reception of the last start pulse.

The circuit diagram of the interface is shown in Fig 5. It has the following I/O lines:

FROM BOARD	TO
COMPLETE	CB1 of PIA
START	CB2 of PIA
DATA INPUTS 0 to 7	PB0 to 7 of PIA
FEED OUT	Feed out solenoid driver cct.
MOTOR	Motor relay cct.
TAPE OUT	Tape out switch on punch mechanism
DATA OUTPUTS 0 to 7	Solenoid driver ccts. 0 to 7
SENSOR +/−	Sensor on punch mechanism

A wiring diagram colour coded for the socket on the back of our punch mechanism as it was received is shown in Fig 6.

There are nine solenoid driver circuits, one for feed out, eight for the data solenoids. These driver circuits are connected to the data outputs from the interface, likewise for the motor relay circuit.

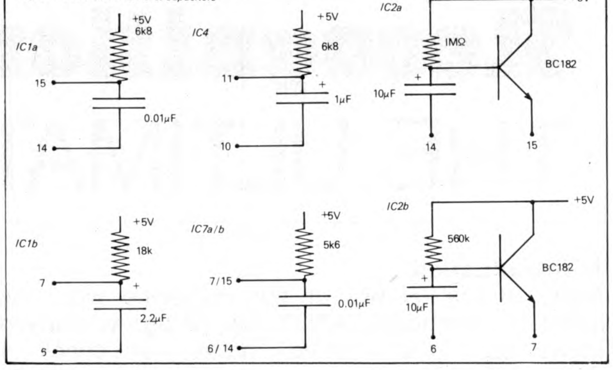
Fig.6 *Colour code for socket on back of mechanism*

		+ −
SOLENOID	*0*	*PURPLE*
	1	*GREEN*
	2	*ORANGE*
	3	*RED*
	4	*BROWN*
	5	*WHITE*
	6	*YELLOW*
	7	*PINK*
	F/O	*BLUE*
SENSOR		
TAPE OUT		*BLACK*
		GREEN
		RED

CONSTRUCTIONAL NOTES

1. The wire from the sensor to the interface board must be screened.
2. The unit is sensitive to noise, unless a 0.1uF disc ceramic capacitor per 5 chips is attached to the interface board, across the 5V power supply, along with one 100uF electrolytic capacitor.
3. If it is desired that the punch be operated at some distance from the PIA, it is advisable to attach 0.05uF capacitors to all the PIA outputs where they are attached to the interface board. This prevents spurious pulses arriving on the board due to the high output impedance of the PIA. The prototype was operated through a screened cable of 12 feet in this manner, from a 6820 PIA.
4. It is advisable to put a mains filter in the punch unit, as the high surge currents that the motor requires to start are liable to cause heavy mains transients.

Fig. 7 *Monostable Resistors/Capacitors*



COMPONENT LIST

The following is a list of unmarked components:

DIODES

D1 to D4	any 100V 5A diodes, or a 10A bridge rectifier.
D5 to D14	any 75V 3A diodes.

RESISTORS

R1 to R9	12.5 ohm 30W. (2 x 6.8 ohm @ 17W in series)

CAPACITORS

C1 to C10	any combination of capacitors, each at least 30V working, giving a total capacitance of 90 mFd (sic).

TRANSISTORS

Use types shown or equivalents.

INTEGRATED CIRCUITS

All are 74 series TTL. See FIG 7 for monostable timing resistors etc.

IC1	74123
IC2	74123
IC3	7474
IC4	74121
IC5	7402
IC6	7402
IC7	74123
IC8	7404
IC9 to IC12	4 x 7474

PCW Software for running the punch will be published in the next issue. **PCW**

Dr. David Hand

Pattern Recognition THE ULTIMATE INTERFACE

1. Introduction
Many of the readers of this magazine will have heard of computers which can recognise spoken commands, read printed matter, or check if signatures are genuine, and will be wondering if it is possible to program their own computer system to do similar things. The short answer is yes, it is possible, and this article explains how.

A natural place to begin is to ask what the common factor is which underlies these processes. The answer is simply: classification. We want to classify a spoken word as belonging to one of several possible commands, to classify a letter from a page into one of 26 classes, and to assign a signature to one of the classes *true or forgery*. The science of automating this process of classification is known as "pattern recognition". Apart from the examples given above it has also been applied to fingerprint classification, target recognition, chromosome analysis, speaker identification, engine fault detection, cancer cell identification, particle detection in nuclear physics, medical diagnosis, etc, etc. The list is virtually endless. Although it's unlikely that you will want to use it for chromosome analysis, how about for keeping an eye on the cooking food, or deciding when the vegetables in the garden are at their tastiest, or predicting when the neighbours will complain about the noise from your party?

Since all of these problems have a classification at their root I can talk in general terms and the results will be applicable to any of them.

2. An elementary pattern recognition system
The structure behind pattern recognition systems can be divided into two areas:

(a) Transducers
(b) Classifiers

The first area is the one nearest the outside world. *Transducers* are merely measuring instruments: the first step in classifying an object is to take a number of measurements on it. Clearly what measurements are taken will depend very much on the particular things being classified. For example, if you want to classify spoken words, you might take measurements which describe the waveform of the word. Or to classify an aerial photograph you might use a measure of texture — a ploughed field looks different from a field of grass. To decide what type a beetle is you might measure its length and colour.

The second part of the pattern recognition system, the classifier, then compares these measurements with stored measurements taken from objects of known classes. The new object is then assigned to the class which it most closely resembles, judged from the measurements.

You can see that the term "pattern" really means a set of numbers — the measured values. In mathematical terminology the pattern describing the object is a **vector**.

Usually, in fact, there is another process between (a) and (b). This process, termed "feature extraction", takes the crude measurements as input and then outputs another set of numbers (called "features") which lead to easier classification. I shall not dwell on this here but in order to stick to standard usage I shall talk of features rather than measurements — it makes no difference to the methods of designing the classifier since both are simply sets of numbers. In any case, the division into feature extraction and classification stages is a little artificial and is only made for reasons of practical convenience.

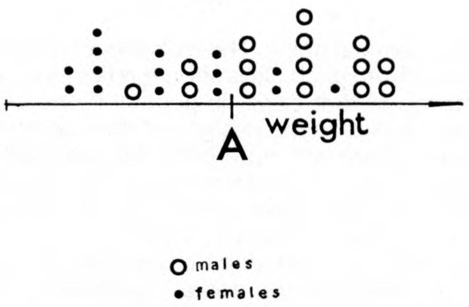
3. An example

One of the easiest ways to gain an understanding of pattern recognition techniques is to consider an example. The example I have chosen is trivial but demonstrates a number of points: suppose the aim is to classify people into sexes automatically. First it is necessary to take some measurements. Weight is an easy thing to determine automatically so I shall assume that the first feature (measurement) is weight. We begin by measuring this for a set of known men and a set of known women — let's say fifteen of each. This set of people of known classification is called the "design set". It's from them that the classifier will be designed. These thirty values might give something like figure 1 and from this a classification rule can be devised:

Classify a new person as male if his/her weight is greater than A.

Classify a new person as female if his/her weight is less than A.

Fig 1



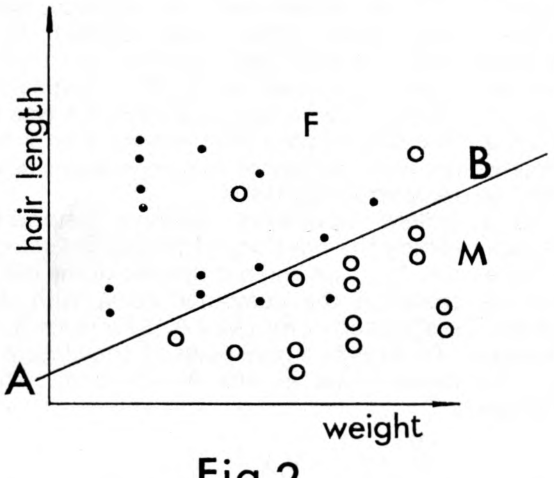
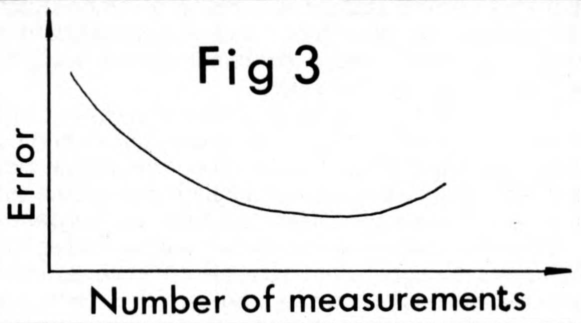


Fig 2

Although this rule will usually lead to a correct decision, the diagram shows that there will be some errors. There are three males and three females in the design set on the wrong side of A so it's not unreasonable to expect new people to fall on the wrong side occasionally. No matter where A is put in figure 1 the number of errors cannot be decreased.

However, there is another way this error can be reduced — by adding *further* features. Suppose you had a device which automatically measured hair length. If hair length was used by itself as a single feature for the classification then errors would result — some men have long hair and some women have short hair. But now try combining hair length with weight. This is shown in figure 2. Each of the thirty people in the design sample now corresponds to a point in a two-dimensional plane. This means that instead of simply seeing which side of a point (A in figure 1) they lie on it is necessary to see which side of a *line* they lie on. AB is such a line and it separates the classes quite well: a newcomer is classified as male if he/she lies on side M of AB, and female if on side F.

Fig 3



With only a single feature six of the thirty members of the design set were misclassified. With two features and a straight line classifier this is reduced to four.

This idea can be extended by introducing a third feature and so on. With three features the people in the design set correspond to points in a three dimensional space and you will need a *plane* to separate the two classes and then classify new comers by seeing which side of the plane they lie on. With four or more features the separating surface will be a "hyperplane".

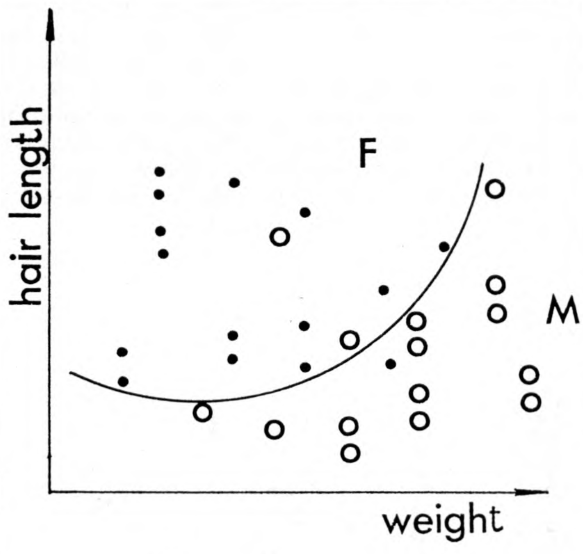


Fig 4

4. Choosing features

When choosing which features to use in a pattern recognition system you should generally try to choose features which are not closely related. This is because with two closely related features there will possibly be little advantage in using both over using either one separately. To go back to my example, suppose height was chosen as the second feature. Since tall people are likely to be heavy it's unlikely that this second feature will improve things much. Hair length, on the other hand, is independent of weight, so when used in combination with weight it will lead to a better classifier.

Theoretically you could go on adding features until the error was very low or even zero, but in practice there are limits. First there is the computation time problem. With 100 features calculating on which side of a hyperplane a point lies might take too long.

Second, there is a more subtle objection. Although it might seem sensible to have as many features as possible on the argument that "although each one separately only reduces the error a little, if there are lots of them the effect will add up," this isn't quite true. In fact it is known that the relationship between the number of features and the error is as in figure 3 for a fixed size of design set.

The reason for this is complicated, but it's basically due to the fact that the more features there are the larger is the design set required to represent the classes adequately. It's generally true that the design set should be as large as possible — and should certainly contain several times as many points as there are features. One of the primary aims of the feature extraction process referred to in section 2 is to reduce the number of features being used so that the classifier can be more reliable.

5. Other decision surfaces and assessing errors

The surface which separates the classes — a point when there is only one feature, a line when there are two, etc — is usually called a "decision surface". And it needn't always be linear (flat). Apart from increasing the number of features another way to reduce the amount of error is to increase the complexity of the decision surface between the two classes. To return to the sex classification example, still using weight and hair length, you could try using a quadratic decision surface — figure 4. This misclassifies only three points of the design set.

In practice whenever you assess the error of a classifier so that you can compare it with other classifiers you must use a *different* set of data from the design set. This second set is usually called the "test set". The reason for this is simply that an error assessment based on the design set will be optimistic — it will give too low a value. This is hardly surprising since the decision surface is chosen so that it correctly classifies as much of the design set as possible.

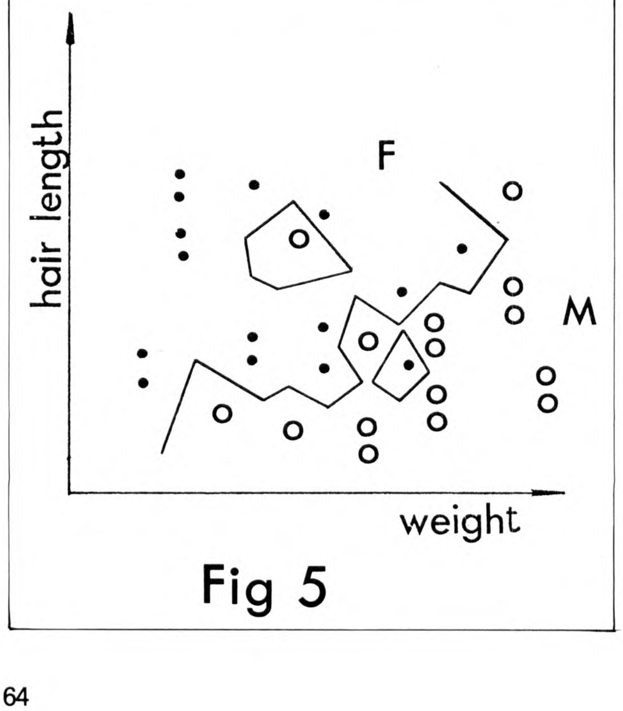


Fig 5

To illustrate this point consider another type of classifier, the "nearest neighbour" classifier. Here a new point is classified as belonging to the same class as its nearest neighbour in the design set. This is demonstrated for the sex example in figure 5. Now clearly this classifies *all* points of the design set correctly. But that doesn't mean it will have zero error when used on new people!

When more than two classes are involved you can easily extend these ideas. Figure 6 shows linear classifiers and nearest neighbour classifiers being used for a three class problem with two features. To classify a new point in figure 6(a) you must see on which side of each of the lines the point lies.

6. Writing classification programs

So far this has all been fairly theoretical. Now let's turn to the practical implementation of classifiers.

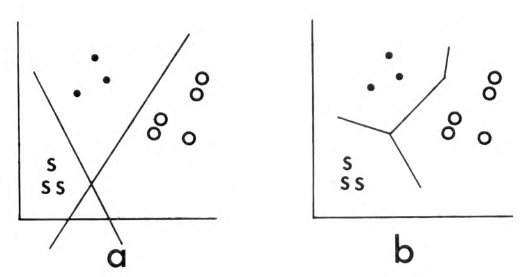


Fig 6

First, a general point. The numbers of points in the design set from each class should reflect the size and importance of that class. In general if one class has lots more points in the design set than another class then new points will more often be classified as the first class. Figure 7 illustrates this. In 7(a) class A and class B both have three points. In 7(b) class A has three and class B has ten. Clearly in 7(b) there is more chance of a new point being classified as class B by (for example) the nearest neighbour classifier shown. This is fine if class B is really much larger than class A (in printed text, for instance, some letters occur more often than others) or if *mis*classifying a point as class A is more serious than the other way round (switching off a hospital life support system if the patient would live if it was left on, is a more serious error than leaving it on if he'll die anyway). But if *neither* of these conditions is true then too many errors will result.

To implement the nearest neighbour method the vectors defining the positions of the design set (say X where i = 1, . . .,n and n is the size of the design set) are stored ié the computer along with their correct classification (1 for class A, 2 for class B, for example). To classify a new point Y, the closest X_i to Y is found. That is, the X_i is found which minimises

$$\sum_{j=1}^{d} (X_{ij} - Y_j)^2$$

where d is the number of features, X_{ij} is the jth component of X_j, and Y_j is the jth component of Y. Point Y is then assigned to the class of X_i.

In some cases these repeated distance calculations can be too time-consuming — especially if very quick classification is vital. Ways do exist to reduce the time but they are fairly complex.

Linear classifiers, on the other hand, lead to particularly simple and quick calculations. At this point it is worth commenting that problems which cannot be directly tackled by linear classifiers can be tackled if the raw measurements are suitably *transformed* first. Figure 8 illustrates this. Figure 8(a) shows the raw data, which has two measurements for each point. If the transformation $r = \sqrt{x_1^2 + x_2^2}$ is used figure 8(b) results — leading to classes which are easily separated by that ultimately simple decision surface, a point. This is another example of feature extraction and demonstrates the relationship between feature extraction and classification: you could use a circular decision surface on the raw data (8(a)), or you could transform the data (feature extraction) and then use a *simpler* decision surface (8(b)).

The general equation for a line in a plane is

$$w_0 + w_1 y_1 + w_2 y_2 = 0.$$

On one side of this line $w_0 + w_1 y_1 + w_2 y_2 > 0$ and on the other side $w_0 + w_1 y_1 + w_2 y_2 < 0$. Thus to

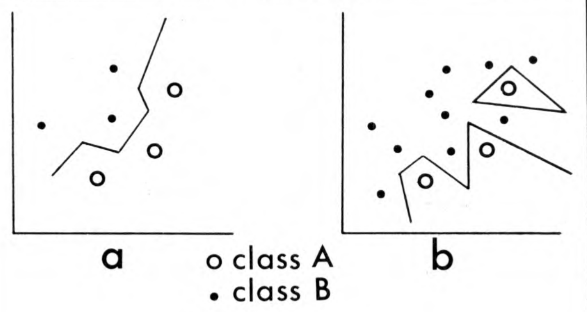


Fig 7

decide which side of this line a point with measured values x_1 and x_2 lies on you calculate $w_0 + w_1 x_1 + w_2 x_2$ and see if it's greater than or less than zero.

For the general case with d measurements you must calculate

$$w_0 + w_1 x_1 + \ldots + w_d x_d.$$

In **vector** notation this is **w . x** (for convenience we have defined a new component of **x**, $x_0 = 1$) and the classification rule is

If **w .x** <O assign to one class
If **w .x** >0 assign to the other class.

With two features you can easily plot the points and find a good decision surface by eye, but this is not possible for more than two features so *algorithms which can be programmed* have been devised. A particularly simple way to estimate the w_i is the following iterative technique where $\mathbf{w}^{(k)}$ is the estimate of **w** at the kth iteration:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \frac{1}{k} \sum_{i \varepsilon S} \mathbf{X}_i$$

where X_i is the ith point of the design set and the summation is over all points of the design set which are *mis*classified by $\mathbf{w}^{(k)}$

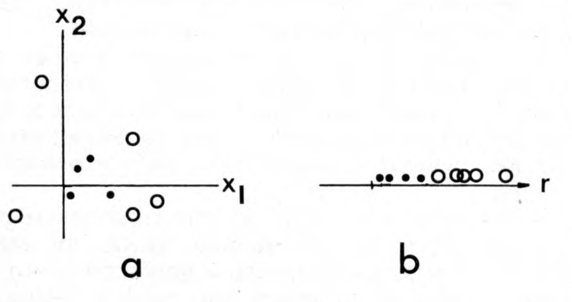


Fig 8

If the design set can be perfectly separated by a linear surface then this method is known to lead to a **w** vector which does it. Even if perfect linear separation is not possible (figure 2) this method should yield a good decision surface.

Finally, two more general points. The places where errors are most likely to occur are places near the decision surface. The linear classifier can easily be modified to reject points very near the decision surface as "unsafe to classify". To do this define a threshold t and classify

x as class A if **w . x** > t
X as class B if **w . x** <−t
x as unknown if $-t \leqslant$ **w.x** $\leqslant t$

This brings me naturally to the second point which is that one of the best ways to use a pattern recognition system is interactively. The difficult decisions ("x unknown") can be settled by the operator while the vast mass of easier ones are done by the computer. In fact baggage handling at some airports is done interactively: the operator reads the label aloud and the computer repeats aloud what its pattern recognition system thinks he said. If it's right the operator moves to the next label, otherwise he repeats his words.

I hope this introductory article has given you a taste of the possibilities of pattern recognition. It is a very young and fast growing science and already its techniques have been applied to a vast number of practical problems. Now you can begin to think about extending that range. The possibilities are limited only by the imagination.

APPLE 11 MICRO COMPUTER

Simply the best!

Apple II comes in a portable, 10 lbs. zip case and is the ultimate personal computer, with powerful colour graphics, extended Basic, 8 I/O ports, 1000 cps. display, 8K ROM, up to 48K RAM, 350ns memory access, mini assembler and much more!

From £995, inclusive of cables, manuals, software. Maintenance arranged.

Personal Computers Limited

18-19, Fish Street Hill, London EC3R 6BY.
Tel; 01-623 1434. Telex 88264.

MAKING A SPLASH

F. Heathman

COMPUTE YOUR FOOTBALL POOLS ENTRY

I am well aware of the verdict of mathematicians, that there is no advantage to be gained by the use of a computer to predict the results of football pools, but it is never-the-less fascinating to attempt to devise an algorithm that gives good correlation with the actual scores and draws.

Some years ago, I produced such a program and it gave sufficiently interesting results (yes I did win small amounts) that I intend repeating it now that I do not depend on 'borrowed' computer time.

The following outline may be of interest to readers if only to have my experience, contradicted.

In order to check the feasibility of the project, I took the previous years results for one Division which were in the form of a table (as in Fig. 1). By rearranging rows and columns I was able to obtain a table similar to that in Fig. 2. That convinced me that toward the end of a season I would be able to at least get more value from my 20p stake.

Although a program could be written to rearrange rows and columns until the best comparison with the ideal format was found, in fact I chose the mathematical approach described below, to establish Home and Away values for each team. This amounts to the same thing, but allows for modification and is useful earlier in the season.

Fig. 1

DIV 1 AWAY / HOME	Arsenal	Aston Villa	Birmingham	Bristol C.	Coventry	Derby	Everton	Ipswich	Leeds	Leicester	Liverpool	Man. City	Man. Utd.	Middlesbro	Newcastle	Norwich	Q.P.R.	Stoke	Sunderland	Tottenham	West Brom.	West Ham
Arsenal	–	3-0	4-0	0-1	2-0	0-0	3-1	1-4	1-1	3-0	1-1	0-0	3-1	1-1	5-3	1-0	3-2	2-0	0-0	1-0	1-2	2-3
Aston Villa	5-1	–	1-2	3-1	2-2	4-0	2-0	5-2	2-1	2-0	5-1	1-1	3-2	1-0	2-1	1-0	1-1	1-0	4-1	2-1	4-0	4-0
Birmingham	3-3	2-1	–	3-0	3-1	5-1	1-1	2-4	0-0	1-1	2-1	0-0	2-3	3-1	1-2	3-2	2-1	2-0	2-0	1-2	0-1	0-0
Bristol C.	2-0	0-0	0-1	–	0-0	2-2	1-2	1-2	1-0	0-1	2-1	1-0	1-1	1-2	1-1	3-1	1-0	1-1	4-1	1-0	1-2	1-1
Coventry	1-2	2-3	2-1	2-2	–	2-0	4-2	1-1	4-2	1-1	0-0	0-1	0-2	1-1	1-1	2-0	2-0	5-2	1-2	1-1	1-1	1-1
Derby	0-0	2-1	0-0	2-0	1-1	–	2-3	0-0	0-1	1-0	2-3	4-0	0-0	0-0	4-2	2-2	2-0	2-0	1-0	8-2	2-2	1-1
Everton	2-1	0-2	2-2	2-0	1-1	2-0	–	1-1	0-2	1-2	0-0	2-2	1-2	2-2	2-0	3-1	1-3	3-0	2-0	4-0	1-1	3-2
Ipswich	3-1	1-0	1-0	1-0	2-1	0-0	2-0	–	1-1	0-0	1-0	1-0	2-1	0-1	2-0	5-0	2-2	0-1	3-1	3-1	7-0	4-1
Leeds	2-1	1-3	1-0	2-0	1-2	2-0	0-0	2-1	–	2-2	1-1	0-2	0-2	2-1	2-2	3-2	0-1	1-1	1-1	2-1	2-2	1-1
Leicester	4-1	1-1	2-6	0-0	3-1	1-1	1-1	1-0	0-1	–	0-1	2-2	1-1	3-3	1-0	1-1	2-2	1-0	2-0	2-1	0-5	2-0
Liverpool	2-0	3-0	4-1	2-1	3-1	3-1	3-1	2-1	3-1	5-1	–	2-1	1-0	0-0	1-0	1-0	3-1	4-0	2-0	2-0	1-1	0-0
Man. City	1-0	2-0	2-1	2-1	2-0	3-2	1-1	2-1	2-1	5-0	1-1	–	1-3	1-0	0-0	2-0	0-0	0-0	1-0	5-0	1-0	4-2
[illegible]	3-2	2-0	2-2	2-1	2-0	3-1	4-0	0-1	1-0	1-1	0-0	3-1	–	2-0	3-1	2-2	1-0	3-0	3-3	2-3	2-2	0-2
[illegible]	3-0	3-2	2-2	0-0	1-0	2-0	2-2	0-2	1-0	0-1	0-1	0-0	3-0	–	1-0	1-0	0-2	0-0	2-1	2-0	1-0	1-1
[illegible]	0-2	3-2	3-2	0-0	1-0	2-2	4-1	1-1	3-0	0-0	1-0	2-2	2-2	1-0	–	5-1	2-0	1-0	2-0	2-0	2-0	3-0
[illegible]	1-3	1-1	1-0	2-1	3-0	0-0	2-1	0-1	1-2	3-2	2-1	0-2	2-1	1-0	3-2	–	2-0	1-1	2-2	1-3	1-0	1-0
[illegible]	2-1	2-1	2-2	0-1	1-1	1-1	0-4	1-0	0-0	3-2	1-1	0-0	4-0	3-0	1-2	2-3	–	2-0	2-0	2-1	1-0	1-1
[illegible]	1-1	1-0	1-0	2-2	2-0	1-0	0-1	2-1	2-1	0-1	0-0	0-2	3-3	3-1	0-0	0-0	1-0	–	0-0	0-0	0-2	2-1
[illegible]	2-2	0-1	1-0	1-0	0-1	1-1	0-1	1-0	0-1	0-0	0-1	0-2	2-1	4-0	2-2	0-1	1-0	0-0	–	2-1	6-1	6-0
[illegible]	2-2	3-1	1-0	0-1	0-1	0-0	3-3	1-0	1-0	2-0	1-0	2-2	1-3	0-0	0-2	1-1	3-0	2-0	1-1	–	0-2	2-1
[illegible]	0-2	1-1	2-1	1-1	1-1	1-0	3-0	4-0	1-2	2-2	0-1	0-2	4-0	2-1	1-1	2-0	1-1	3-1	2-3	4-2	–	3-0
West Ham	0-2	0-1	2-2	2-0	2-0	2-2	2-2	0-2	1-3	0-0	2-0	1-0	4-2	0-1	1-2	1-0	1-0	1-0	1-1	5-3	0-0	–

Fig 1a.

	A r	A s	B i	B r	C o	. . . .
Ar	1	1	1	2	1	
As	1	1	2	1	X	
Bi	X	1	1	1	1	
Br	1	X	2	1	X	
Co	2	2	1	2	1	

Fig 1b

	A r	B r	B i	C o	A s	. . .
Bi	X	1	1	1	1	
Co	2	X	1	1	1	
Ar	1	2	1	1	1	
As	1	1	2	X	1	
Br	1	1	2	X	X	

n.b. In a game versus itself a team is given a home win.

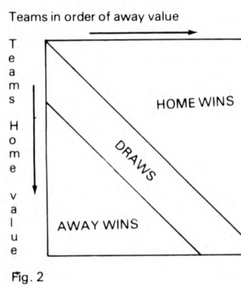
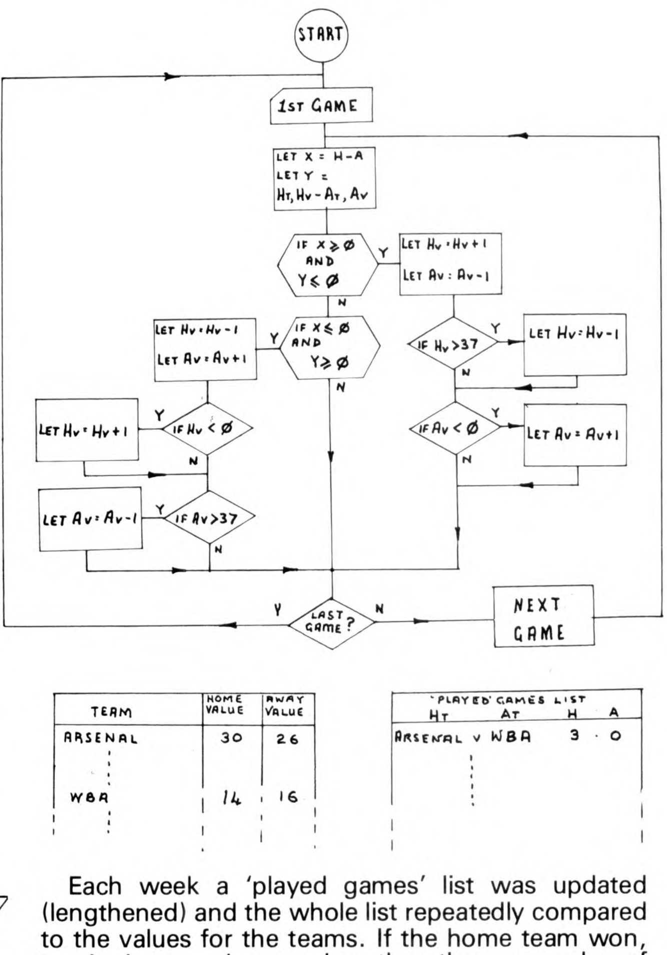


Fig. 2



TEAM	HOME VALUE	AWAY VALUE
ARSENAL	30	26
WBA	14	16

'PLAYED' GAMES LIST HT	AT	H	A
ARSENAL v	WBA	3	0

Each week a 'played games' list was updated (lengthened) and the whole list repeatedly compared to the values for the teams. If the home team won, but its home value was less than the away value of the other team, its home value was incremented and the other team's away value, decremented, similar adjustments being made for the other possibilities, the next game was then considered.

After repeated cycling these values converged to those needed to represent the results that were known. Arbitrary starting values were used as well as numbers related to the previous year's League positions, both converged to the same relative values.

In my program the cycle was allowed to continue for ten minutes, since this was convenient, but a more sophisticated technique could be used.

To predict results, a comparison between the home value of the home team and the corresponding away team value was made. I used the results to look for similar home and away values, to predict draws, but less avaricious readers may wish to predict all games.

A flowchart of the basic algorithm is shown *(Fig.3)*. The simplicity of the method allows for many enhancements, use of the actual goal score being the most obvious. Comparison with random selections would be an interesting measure of the algorithms success.

The Exhibition with the difference

PERSONAL COMPUTER WORLD SHOW

AT THE WEST CENTRE HOTEL LONDON S.W.6
ON 21, 22, 23 SEPTEMBER 1978

FEATURING	ATTRACTIONS	PLUS
Systems for small businesses	Competitions	Famous Personalities
Computers in Education	Best home brew system	Innovators corner
Computers in the home	Best software	Video games
Amateur computing	Best school applications	PCW Micro chess championship
	Best home applications	Seminars, PCW consultants in attendance

Three days to remember, Seminars:
Thursday 21st: Small Computers in Schools and Universities
Friday 22nd: Small Computers and Small Businesses
Saturday 23rd: Computing for the Beginner and Hobbyist

Readers interested in exhibiting, attending the seminars or visiting the Exhibition, please write for further details to: **Will Martin,** Interbuild Exhibitions Ltd., 11 Manchester Square, London W1M 5AB.

BUZZWORDS

Barry G. Woollard

ACIA — Asynchronous Communications Interface Adaptor.
Accumulator — One or more registers associated with the ALU which temporarily store sums and other arithmetical and logical results of the ALU.
Addressing Modes — an address is a coded instruction designating the location of data or program segments in storage. The address may refer to storage in registers or memories or both. The address code itself may be stored so that a location may contain the address of data rather than the data itself. This form of addressing is common in microprocessors. Addressing modes vary considerably because of efforts to reduce program execution time.
ALU (Arithmetic and Logic Unit) — the ALU is one of the three essential components of a microprocessor, the other two being the registers and the control block. The ALU performs various forms of addition and subtraction; the logic mode performs such logic operations as ANDing the contents of two registers, or masking the contents of a register.
ALGOL — **ALGO**rithmetic **L**anguage.
Architecture — Any design or orderly arrangement perceived by man; the architecture of the microprocessor.
ASCII — **A**merican **S**tandard **C**ode for **I**nformation **I**nterface.
Assembler Program — translates man-readable source statements (mnemonics) into machine understandable object code.
Assembly Language — a machine oriented language. Normally the program is written as a series of source statements using mnemonic symbols that sugest the definition of the instruction andis then translated into machine language.
Baud Rate — a measure of data flow. The number of signal elements per second based on the duration of the shortest element. When each element carries one bit, the Baud rate is numerically equal to bits per second (bps). The Baud rates on UART data sheets are interchangeable with bps.
BASIC — **B**eginners **A**ll purpose **S**ymbolic **I**nstruction **C**ode.
Branch — refers to the capability of a microprocessor to modify the function or program sequence. Such modification depends on the actual content of the data being processed at any given instant.
Breakpoint — a program point indicated by a breakpoint flag which invites interruption to give the user the opportunity to check his program before continuing to its completion.
Buffer — a circuit inserted between other circuit elements to prevent interactions, to match impedances, to supply additional drive capability, or to delay rate of information flow. Buffers may be inverting or non-inverting.
Bus Driver — an IC which is added to the data bus system to facilitate proper drive to the CPU when several memories are tied to the data bus line. These are necessary because of capacitive loading which slows down the data rate and prevents proper time sequencing of microprocessing operations.
Bus System — a network of paths inside the microprocessor which facilitate data flow. The important buses in a microprocessor are identified as Data Bus, Address Bus, and Control Bus.
Byte — indicates a pre-determined number of consecutive bits treated as an entity. E.g. 4-bit or 8-bit bytes. 'Word' and 'Byte' are used interchangeably.
COBOL — **CO**mmon **B**usiness **O**riented **L**anguage.
Compiler — translates high-level languages into machine code.
Control Bus — conveys a mixture of signals which regulate system operation. These 'traffic' signals are commands which may also originate in peripherals for transfer to the CPU or the reverse.
CORAL — **C**omputer **O**n-line **R**eal-time **A**pplications **L**anguage. This is a development of ALGOL. CORAL 66 was originally developed by the RRE for military projects, but has since been adapted for such applications as Airline Reservations and the Post Office Viewdata system.
CPU (Central Processing Unit) — the heart of any computer system. The CPU comprises storage elements called registers, computational circuits in the ALU, the Control Block, and I/O. The one-chip LSI microprocessors have limited storage space, so memory implementation is added in modular fashion.
Cross-Assembler — when the program is assembled by the microprocessor that it will run on, the program that performs the assembly is referred to simply as an assembler. If the program is assembled by some other microprocessor, the process is referred to as cross-assembly.
Data Bus — the microprocessor communicates internally and externally by means of the data bus. It is bidirectional and can transfer data to and from the CPU, memory storage, and peripheral devices.
Debug — a term used in connection with microprocessor software. Debugging involves searching for and eliminating sources of error in programming routines.
Dedicated — a microprocessor system that has been specifically programmed for a single application such as weight measurement by scale, traffic light control.
Direct Addressing — the standard addressing mode. It is characterised by an ability to reach any point in the main storage directly.
DMA (Direct Memory Access) — a method of gaining direct access to main storage to achieve data transfer *without involving* the CPU.
Execution Time — usually expressed in clock cycles necessary to carry out an instruction.
EAROM — **E**lectrically **A**lterable **R**ead **O**nly **M**emory.
EPROM — **E**rasable **P**rogrammable **R**ead **O**nly **M**emory.
Fields — a source statement is made up of a number of code fields which are acceptable to the assembler. These may be Label, Operator, Operand and Comment.
Firmware — software instructions which have been permanently frozen into a ROM.
Flag Bit — an information bit which indicates some form of demarcation has been reached such as overflow or carry. Also an indicator of special conditions such as interrupt.
Flow Chart (or Diagram) — a sequence of operations charted with the aid of symbols or other representations to indicate an executive program.
FORTRAN — **FORM**ula **TRAN**slator.
Handshaking — a colloquial term describing the method used by a Modem to establish contact with another Modem at the other end of a telephone line. Often used interchangeably with buffering and interfacing, but with a fine line of difference in which handshaking implies a direct package to package connection regardless of functional circuitry.
Hardware — the individual components of a circuit, and any piece of data processing equipment.
High-Level Language — this is a problem-oriented programming language, i.e. the instruction approach is closer to the needs of the problems to be solved than to the machine.
Immediate Addressing — in this mode, the operand contains the value to be operated on, and no address reference is required.

Indirect Addressing — addressing a memory location which contains the address of data rather than the data itself.
Instruction Set — constitutes the total list of instructions which can be executed by a given microprocessor.
Interface — indicates a common boundary between adjacent components, circuits or systems enabling the devices to yield and/or acquire information from one another.
Interrupt — an interrupt involves the suspension of the normal programming routine of a microprocessor in order to handle a sudden request for service. The importance of the interrupt capability of a microprocessor depends on the kind of applications to which it will be exposed. When a number of peripheral devices interface the microprocessor, one or several simultaneous interrupts may occur on a frequent basis.
Interrupt Mask Bit — prevents the CPU from responding to further interrupt requests until cleared by execution of programmed instructions.
Label — may correspond to a numerical value or a memory location in the programmable system.
Library — a collection of complete programs written for a particular computer, minicomputer or microprocessor.
LIFO — **L**ast — **I**n **F**irst — **O**ut.
Look Ahead — (1) a feature of the CPU which allows the machine to mask an interrupt request until the following instruction has been completed.
(2) a feature of adders and ALUs which allow them to look ahead to see that all carrys generated are available for addition.
Machine Language — the only language the microprocessor can understand is binary. All other programming languages must be translated into binary code before entering the processor and decoded into the original language before leaving. it.
Mnemonic Code — these are designed to assit the human memory. The machine code binary strings are assigned groups of letters (or mnemonic symbols) that suggest the definition of the instruction.
Multiplexing — describes a process of transmitting more than one signal at a time over a single link, route, or channel.
Object Program — the end result of the source language program after it has been translated into machine language.
Operand — a quantity on which a mathematical operation is performed.
PLA (Programmed Logic Arrays) — an orderly arrangement of logical AND and OR functions. Its application is very much like a glorified ROM. It is primarily a combinational logic device.
Polling — is the method used to identify the source of interrupt requests. When several interrupts occur at one time, the control program decides which one to service first.
Port — Device terminals which provide electrical access to a system or circuit. The point at which the I/O is in contact with the outside world.
Real-Time Operation — a data processing technique used to allow the machine to utilise information as it becomes available, as opposed to batch processing at a time unrelated to the time the information was generated.
Register — a memory on a smaller scale. The words stored may involve arithmetical, logical, or transfer operations. Storage in registers may be temporary, but even more important is their accessibility by the CPU. The number of registers in a microprocessor is one of the important features of its architecture.
Scratchpad — applies to information which the CPU stores or holds temporarily. It is a memory containing subtotals for various unknowns which are needed for final results.
Software — is the computer's instruction manual. It is the language used by a programmer to communicate with the computer.
Source Statement — a program written in other than machine language.
Simulator — a special program that stimulates the logical operation of the microprocessor. It is designed to execute object programs generated by a cross-assembler on a machine other than the one being worked on and is useful for checking and debugging programs prior to committing them to ROM firmware.
Stack — is a block of successive memory locations which is accessible from one end (LIFO). The stack is coordinated with the stack pointer which keeps track of storage and retrieval of each type of information in the stack.
Subroutine — part of the master routine which may be used at will in a variety of master routines. The object of a Branch or Jump Command.
UART (Universal Asynchronous Receiver Transmitter) — this device will interface a word parallel controller or data terminal to a bit serial communication network.
Vectored Interrupt — describes a microprocessor system in which each interrupt, both internal and external have their own uniquely recognisable address. This enables the microprocessor to perform a set of specified operations which are preprogrammed by the user to handler each interrupt in a distinctly different manner.
Word — a group of 'characters' treated as a unit and given a single location in computer memory.

MATROX
FROM
SHELTON

			PRICE (1 off)
1.	ALT-256**2E	256 x 256 S100 graphics card	£259.00
2.	ALT-2480E	24 lines of 80 characters S100	£199.00
3.	MTX-816	Big characters 8 rows 16 characters per line	£128.00
4.	MTX-1632	Very clear characters 32 characters 16 lines (SL version can be synchronised to TV picture)	£162.00
5.	MTX-A1/MTX-B1	Keyboard scanners and LED driver Single chips direct connection to any CPU bus	£28.00

SHELTON INSTRUMENTS LTD.,
22/24 Copenhagen Street, London N1 0JD Tel: 01-278 6273

How to find out through Libraries

Mark O'Connor

This article is based on the experience that most people are to a greater or lesser extent ignorant of the information resources available in libraries. When you enter the modern library you do not only enter a building which contains a collection of books, it is more; a terminal connected with every other library in the world. Through inter-library lending schemes, such as Laser for public libraries, a book or journal article can be obtained, if it is not in stock in that particular library. Access to libraries, at least for reference purposes, is much easier than might be imagined, for as well as the full public library service, to which every citizen is entitled, most Academic libraries must make their stock available to the general public for reference: often attention is not drawn to this because of the lack of space for readers in academic institutions. In London, the Science Reference Library, a branch of the British Library, is open for reference purposes: the branch of this library that would probably be most use to computer users, is the Holborn Branch, at 25 Southampton Buildings, Chancery Lane, WC2A 1AW, which was the Patent Office library, and the stock of which is orientated towards the physical sciences and the literature of technologies. The other branch is in Bayswater, at 10 Porchester Gardens, London, W2 4DE, and this contains materials on medicine, earth sciences, astronomy and pure mathematics. There is no formality of joining or obtaining a reader's pass. Help from the staff in the translation of foreign languages is also available.

The categories of materials in libraries may be broadly divided into bibiliographical materials such as abstracts, monographs (books) and serials (journals). General metabibliographical information can be obtained from such guides as Walford's *Guide to Reference Material* and Winchell's *Guide to Reference Books.* Also, libraries contain academic and business directories, such as *The World of Learning* and, most useful for those interested in computers, *The International Directory of Computer and Information Services 1974,* published by Europa, as well as directories of professional bodies which will give addresses of individuals and organizations.

Computer and Control Abstracts are produced in monthly parts by International Information Services (INSPEC) for the physics and engineering communities under the auspices of the Institute of Electrical Engineers (IEE) and other international bodies. This most useful tool covers all aspects of computers and control in abstracts and references derived from a wide range of sources including journals, reports, dissertations, patents and conference papers, published in all the countries of the world and all languages. The abstracts are numbered and classified by subject, with a subject guide, author index and subsidiary indexes. This means that materials can be sought from almost any point of view; for example, if you know that a paper was read at a certain congress, access can be obtained through the conference index. There are also annual cumulations of the author and subject indexes, as well as cumulations over a period of four years. Abstracts may seem rather daunting to those who are unfamiliar with them but when one knows how to use them they can provide access to exhaustive information on a particular subject. Another abstracting source is *Computer Abstracts,* published by the Technical Information Company, in Jersey C.I. Published in monthly parts, with an annual cumulated subject, author and patent indexes, it covers a wide range of computer and computer-related topics from hundreds of periodicals: it started publication in 1960.

Also produced by INSPEC is the monthly *Current Papers on Computers and Control* which is a current-awareness service on the subject. Current-awareness publications are not cumulated but cover the most recent publications, classified by subject, without indexes. One scans current-awareness publications in order to keep up to date within a particular field but there is little indication beyond the title as to the contents of the article, unlike an abstract which gives a resumé of the article. Titles however are usually instructive as they contain 'key words' to facilitate their withdrawal from computer-based information systems.

Computer-based information systems are a recent development in the dissemination of information whereby a literature search can be undertaken by giving a database a certain timespan and keywords. A printout of the titles of relevant articles is then obtained, sometimes with an abstract as well. Central stores of these bibliographical data bases have been set up, for example the Lockheed system at Palo Alto, California, and these may be interrogated via the satellite system from any suitable terminal in the country. Computer-based services often provide information in advance of the printed abstracts and also save time for the user.

Another useful publication is *Computing Reviews,* published monthly by the Association for Computing Machinery (ACM) to furnish 'critical information about all current publications in any area of the computing sciences': it contains fairly extensive reviews of recent publications in the field.

Because computing is such a fast developing field, the standard bibliography is of less use, as it cannot be updated quickly enough. Terminology and the basic concepts are more stable and computer dictionaries exist which give extensive lists of terms, acronyms, et cetera, one example being: Sippl, C.J., *Computer Dictionary and Handbook,* published by Foulsham in 1967.

The range of books in stock will depend on how closely the library is connected with a computer centre. In most colleges a computer is essential for research projects, as well as administrative tasks such as wages. Most academic institutions will have a central, major binary installation, as well as analogue computers in the departments, so it can be depended upon that there will be a fairly comprehensive collection of books and journals. Many local authorities also use computers but this is probably less well reflected in their public library stock.

In libraries which cover many subjects you may find that books on different aspects of computers are widely separated. For example, the books on cybernetics, languages, programming and manuals, may be in one section, and the books about construction and manufacture may be in the electronic engineering section; the former group being under the General class along with information and control theory and bibliographical works, while the latter is found under technology.

Finally, a useful source of information is the librarian. Libraries exist to provide information, and librarians are keen to help you find what you want, as this is the most interesting part of their job.

References:
1. World of Learning. Annually. *Europa.*
2. International Directory of Computer and Information Services, 1974, *Europa.*
3. Computer and Control Abstracts. V.1- , 1966. *Inspec.*
4. Computer Abstracts. 1960- . *Technical Information Co.*
5. Current Papers on Computers and Control. *Inspec.*
6. Computing Reviews. 1960. *ACM.*
7. Sippl, C.J.: *Computing Dictionary and Handbook, 1967. Foulsham.*

THE NEWBEAR COMPUTING STORE

7 Bone Lane, Newbury, Berks.

Software & Literature Section

Software (with paper tape) for the 6800	
Games	
The Bear Game	£2.00
The Well Tempered Microprocessor	£2.00
3K Basic	£5.00
8K Basic	£10.00
Coresident Editor and Assembler	£10.00
and full range for	£2.80
Books p&p 50p each unless otherwise stated	
Computer designs	
77-68 a 6800 Microcomputer	£7.50
Spare diagram set for 77-68	£1.50
WB-1 a TTL Microcomputer	£6.50
Spare Diagram set for WB-1	£1.00
Zilog	
Z-80 Technical Manual	£3.75
Z-80 P10 Technical Manual	£2.25
Motorola	
Understanding Microprocessors	£2.75 (30p p&p)
M6800 Microprocessor Programming Manual	£4.50
M6800 Microprocessor Applications Manual	£9.50 (£1.00 p&p)
M.O.S. Technology	
KIM 1 User Manual	£5.00
6500 Programming Manual	£5.00
6500 Hardware Manual	£5.00
Adam Osbourne	
Introduction to Microcomputers	£5.95
Vol. 0 Beginners Book	£5.95
Vol. 1 Basic Concepts	£5.95
Vol. 2 Some Real Products	£11.95
8080 Programming for Logic Design	£5.95
6800 Programming for Logic Design	£5.95
Z-80 Programming for Logic Design	£5.95
Some Common Basic Programs	£9.95
Payroll with Cost Accounting in Basic	£9.95
General Ledger System	£9.95
Accounts payable and Accounts receivable	£9.95
Sybex	
Microprocessors: from chips to systems. Rodnay Zaks	£8.00
Microprocessor Interfacing Techniques. C207	£8.00
Some Common Basic Programs P.10	£8.00
Scelbi	
'8080' Software Gourmet Guide Cookbook	£7.25
'6800' Software Gourmet Guide Cookbook	£7.25
What to Do After you Hit Return	£7.00
(PCC First book of Computer Games)	
PCC Reference on Home Computers	£4.95
Dr. Dobbs Volume 1	£10.00
Instant Basic	£4.95
Your Home Computer	£4.95
My Computer Likes Me	£1.65
Games with a Pocket calculator	£1.75
Games, Tricks & Puzzles for a hand calculator	£2.45
Best of Creative Computing Vol. 1	£6.95
Best of Creative Computing Vol. 2	£6.95
Best of Byte	£8.50
101 Basic Computer Games	£5.25
Hobby Computers are here.	£3.99

Our range of books is constantly expanding. Please write for latest list.

Goods are normally shipped within 24 hours subject to availability.
Barclay card & Access VAT at 8% for Hardware Components.
30p postage and packing unless otherwise stated.
Cheques to be made out to 'The Newbear Computing Store'

THE NEWBEAR COMPUTING STORE

Hardware Components Section

Memories

2102-1	£1.50
2102L-1	£1.60
2112	£3.04
4027 (250ns)	£6.00
2114 (450ns)	£10.00
4116 (250ns)	t.b.a.
2708 (450ns)	£10.50

Please enquire for 16K dynamic and 100 up prices.

Motorola Microcomputing IC's

MC6800P	£14.00
MC6820P	£6.20
MC6850P	£6.74
MC6810AP	£3.61
MC6830L7	£11.33
MC8602P	£2.88
MC14536P	£3.69
MC3459	£2.53

Zilog Microcomputing IC's

Z-80 CPU 2.5MHz	£15.50
Z-80 P10 PS	£10.00
Z-80 CTC PS	£10.00

Enquire for Z-80A series

Microprocessors

8080A	t.b.a.
SC/MP Mk II	t.b.a.
6502	£14.93
F-8	t.b.a.
2650	t.b.a.

Interfacing IC's

MC1488P	£1.40
MC1489P	£1.40
75150P	£1.30
75150N	£1.20
75154	£2.50
4N33	£1.95

For V.24, RS232C.

Buffers

81LS95	£1.43
81LS96	£1.43
81LS97	£1.43
81LS98	£1.43
8T26	£1.84
8T95	£1.60
8T97	£1.60
74367	£1.30

Standard TTL

7400	16
7402	16
7403	16
7404	24
7406	27
7410	18
7420	18
7430	18
7432	34
7437	36
7440	18
7454	18
7472	28
7474	34
7486	34
7493	57
7495	65
74107	34
74132	52
74151	70
74153	74
74155	78
74157	68
74163	88
74164	£1.00
74175	£1.30
74193	£1.30
74283	£1.10

74LS00	.18
74LS01	.21
74LS02	.21
74LS03	.21
74LS04	.26
74LS05	.26
74LS08	.21
74LS09	.21
74LS10	.21
74LS11	.26
74LS12	.21
74LS13	.55
74LS14	1.26
74LS15	.21
74LS20	.21
74LS21	.26
74LS22	.21
74LS26	.31
74LS27	.21
74LS28	.40
74LS30	.29
74LS32	.24
74LS33	.40
74LS37	.30
74LS38	.30
74LS40	.27
74LS42	.88
74LS47	.96
74LS48	.96
74LS51	.21
74LS54	.21
74LS55	.21
74LS73	.34
74LS74	.38
74LS75	.55
74LS76	.34
74LS78	.34
74LS83	1.05
74LS86	.38
74LS107	.38
74LS109	.38
74LS112	.38
74LS113	.38
74LS114	.38
74LS125	.56
74LS126	.56
74LS132	.90
74LS136	.38
74LS138	1.05
74LS139	1.05
74LS151	.96
74LS153	.96
74LS154	1.98
74LS155	1.05
74LS156	1.05
74LS157	.96
74LS158	.88
74LS160	1.22
74LS161	1.22
74LS162	1.22
74LS163	1.22
74LS164	1.18
74LS168	2.03
74LS169	2.03
74LS170	2.61
74LS173	2.28
74LS174	1.18
74LS175	1.10
74LS189	2.82
74LS190	2.02
74LS191	2.02
74LS193	1.86
74LS196	1.21
74LS197	1.21
74LS247	.96
74LS248	.96
74LS249	.96
74LS253	1.05
74LS257	1.09
74LS258	1.09
74LS266	.38
74LS283	.96
74LS289	1.05
74LS352	.96
74LS353	.51
74LS365	.51
74LS366	.51
74LS367	.51
74LS368	.51
74LS386	.38
74LS670	3.04

All IC's are 100% guaranteed and manufactured by Texas, Signetic, National etc.

Connectors & Sockets

Subminiature D Type

9 way plug	.58
9 way socket	.68
9 way cover	.74
15 way plug	.74
15 way socket	1.09
15 way cover	.83
25 way plug	.92
25 way socket	1.50
25 way cover	.85
37 way plug	2.10
37 way socket	3.11
37 way cover	1.19

Edge Connectors

80 way 0.1" single sided	1.80
45 way 0.1" single sided	1.30
43 way 0.156" double sided *(for Motorola exorcisor, D1, D2 kit bus)*	4.20
64 way plug for Eurocard to DIN 41612	1.65
64 way socket for Eurocard to DIN 41612	2.99
10 way plug Molex for S-50 bus	.30
10 way socket Molex for S-50 bus	.32

D-I-L Sockets (low profile gold plated)

14 pin	.24
16 pin	.25
24 pin	.36
40 pin	.66
24 pin low insertion force	£3.50

D-I-L plugs

16 pin	.60
24 pin	1.20

Scotchflex connectors

40 way plug	2.40
40 way socket	2.62

Printed Circuit Boards

For: 77-68 (8"x8" 0.1" edge 78 way)	
CPU board	10.00
4K RAM board	10.00
Bootstrap loader board	10.00
Prototyping p.c.b. (low pack density)	9.95
Prototyping p.c.b. (high pack density)	10.91
For: E-77 (double euro card)	
4K RAM board	6.25
Prototyping p.c.b. (low pack density)	7.52
For: 3U 43 way systems (4½"x8")	
2K RAM board	8.50
Back plane (Vero) board approx. 10"x4"	1.47
For: S-50 bus	
4K RAM board	12.00
For: S-100 bus	
4K RAM board	15.00

Miscellaneous

Subminiature switches

Single pole change over toggle	.69
Single pole change over (momentary action)	1.00
D-I-L 8 way SPST	1.32

LED's 0.2" High Luminance

Red	.15
Green	.18
Yellow	.18
Orange	.18

Crystals

1.0000 MHz	£3.50
5.000 MHz	£2.60
10.00 MHz	£2.60

Cable

Computing Stationery

C30 Cassettes Memorex MrX2	.75

Coding Form Pads

Lightweight Hexadecimal (for 6800 etc)	.80
Heavyweight Hexadecimal (for 6800 etc)	1.10
Lightweight Hexadecimal (for Z-80)	.80
Heavyweight Hexadecimal (for Z-80)	1.10
8 hole paper tape. 8" spool	1.00
8 hole paper tape. Fan fold.	2.00
Teletypewriter paper roll	1.00
8" floppy discs IBM compatible	4.00

*12½% VAT

Goods are normally shipped within 24 hours subject to availability.
Barclay card & Access VAT at 8% for Hardware Components.
30p postage and packing unless otherwise stated.
Cheques to be made out to 'The Newbear Computing Store'

Send for an up-to-date catalogue to:

The Newbear Computing Store

7 Bone Lane, Newbury.
Tel. 0635-46898.

Callers welcome Monday to Saturday 9.00a.m.-5.30p.m.
The Newbear Computing Store is a division of Newbury Laboratories Ltd.

can we inspire you?

Microprocessor technology is changing the face of the electronic and computer industry, allowing existing products to be more effective and revealing new application areas in industry and the home.

MICROPROCESSORS is a new bimonthly technical journal covering the hardware, software and applications of microprocessors and microcomputers.

For further details please complete and return the coupon.

Annual subscription (six issues) is £23 UK, £35 overseas.

TO: MICROPROCESSORS,
IPC Science and Technology Press Ltd,
IPC House, 32 High Street, Guildford,
Surrey, GU1 3EW, England.
Telephone: Guildford (0483) 71661

Please send me further details on
MICROPROCESSORS

Name ____________________

Organization and address ____________________

____

____

COMPUTING AND DEAFNESS

T. J. Allen

Deafness is perhaps the most neglected handicap there is and yet surprisingly the one where computing and electronics in general could have great significance. As a teacher in a secondary school for the deaf, I am especially interested in the impact computer techniques and microprocessors in particular could have in deaf education.

Blindness is said to cut people off from things, but that deafness cuts people off from people. Computer techniques could help restore the deaf child to the hearing world. Much research needs to be done but one or two obvious areas stand out. The first is the use of electronic devices to provide a visual means of displaying speech patterns. A relatively simple example is to store a perfect speech pattern (or voice print) which could then be displayed on a visual display unit as an example for the child to practise. A library of sounds, phonemes and complete words could be built up for children to practice. Some schools already use oscilloscopes and similar apparatus to display sound patterns but in all but the most expensive equipment, these displays are transitory and thus of little use as practice material for the child.

A microprocessor based system incorporating a memory could be used to store and display the teacher's speech pattern, perhaps at the top of the screen, labelled to show the sound displayed. The child could then practise the sound into a microphone. The child's speech pattern would then also be displayed, below the teacher's and would remain on the screen so the child could compare it with the perfect pattern and notice differences between the two. A further development of the system could be for the unit to compare the two patterns and print out a message to the child such as 'good', 'poor' or 'rubbish'! However, until computers can actually recognise voice patterns, such a refinement is more likely to produce confusion when the teacher disagrees with the machine than trust when he does agree with it.

Audiograms are a method of showing the degree of deafness of the child and are time consuming to obtain and require a trained teacher to be with the child while the audiogram is plotted. The pure tone audiogram presents the child with a pure tone at a precise frequency and varying loudnesses. The child is required to say when he can just hear the tone. The loudness of a particular tone that the child can just hear is plotted against the frequency of the tone. A graph showing the loudness that can just be heard is thus plotted against varying frequency. A microprocessor, with an appropriate program could not only generate the tones but also present them at varying loudnesses, the child responding directly to the machine which would itself plot the audiogram. This would free the teacher from the chore of plotting audiograms, enabling more time to be spent with individual children. Not only would this be of great use in schools for the deaf where audiograms of all children in the school are usually taken at least once a year, but would also be very useful in screening tests carried out in normal schools to identify the hearing impaired.

It takes on average about thirty minutes to plot an audiogram and then work out the average deafness of the child. Thus even in a small school, a large time-saving could be achieved and screening tests could be carried out more accurately, quickly and cheaply. Many children who are partially hearing slip through the screening procedures and these children are often considered to be of low intelligence in normal schools simply because they cannot hear what the teacher says. Their entire educational future is put at risk by inadequate screening procedures. Automated audiograms could identify these children at an early stage, a most important factor in the education of the hearing impaired, for if specialist educational treatment can be given early enough, the child will in all probability be able to attend a normal school rather than a special school. Since it now can cost nearly twice as much to send a child to a special school as to send one to a public school, this would be a great saving on the education budge of the country.

The telephone, which may cause us to curse at times, is perhaps the most widespread mass communication system in existence to-day. However, it is of no use at present to the deaf. The telephone system does provide a ready-made distance communication network which the deaf could use if some mechanical means of sending messages down the telephone lines were available. Such messages are already sent between teletypes and computers. It should surely be possible to manufacture at a reasonable cost a keyboard, V.D.U. and Modem for the deaf person to purchase or which could even be provided by the government, a gesture which might offset the enormous financial advantages which other handicaps enjoy over the deaf such as mobility allowances and blind persons allowances. The deaf, after all, are lucky if they get their hearing aid batteries for nothing.

Deaf children generally have retarded language development which could be assisted by computer techniques. A mini-computer system in a school could be used as a teaching aid to assist in language development. Most of the readers of Personal Computer World can recall the frustration of a carefully worked out program producing only error messages or rubbish because of the lack of a comma or zero. Deaf children have the same sorts of problems in their everyday conversations. The need for the 'and's', 'but's' and 'if's' of the English language is a total mystery to the deaf child. A teaching machine based on a microprocessor unit which would only respond if the correct grammatical form were entered could be a boon in the language development of the deaf.

Mathematical calculators incorporating educational programs are already on the market. A similar process using a microprocessor but dealing with language rather than number could be used with the deaf. For example, a sentence could be displayed but omitting one or more words. The pupil would then be required to enter the correct word in the correct position. If entered correctly, the program would display another sentence. If entered incorrectly, it would be possible to try again. If still wrong the unit would display the correct sentence and then display a further example. The appeal of this system to the teacher would be that a library of increasingly complex sentence structures could be built up, creating a computer assisted language development course. Built into the programs would be a method of recording the score of the pupil and, more importantly, a record of the types of mistakes made by the child. When the child entered the school, the level of language development reached could be ascertained by means of the machine and tests of language development could be carried out by the machine throughout the child's school career. Thus the programs would be a diagnostic tool for the teacher.

For the child, there would be the novelty of working with the machine instead of dull written exercises. However, there would also be the very important factor of receiving correction immediately after the problem was set by the machine, a very valuable factor in the education of any child but especially with the deaf, whose retention of linguistic information tends to be extremely short. Often they have forgotten the object of the exercise by the time it has been marked by the teacher and returned to them. A further use of the machine thus programmed is during the child's free time. Assuming he had been shown how to start the program, no teacher is required for learning to take place. Thus the machine could be used during the child's free time without a teacher being present.

Having got a computer into a school for the deaf, computer science would seem to be an obvious addition to the curriculum. Indeed, even without the language development possibilities of such a machine, computing could be a very useful skill for the deaf school leaver, as a possible area of employment. After all, one does not (yet!) have to actually speak to the machine to program it. Many deaf school leavers go into unskilled employment at the moment, simply because they are unable to cope with the public at large because of their handicap. Computer programming offers a reasonably quiet working environment, without the need to communicate with a large number of people and with the added advantage of the majority of the work being carried out with a teletype rather than a busy, unsympathetic human being.

And human beings are unsympathetic. It takes a great deal of patience, and perseverance to converse with a deaf person. We have all had experience of elderly hard of hearing friends or relatives and can remember the frustration of repeating everything, and of speaking slowly and distinctly. Talking to someone who has been born deaf is even more difficult since they can appear to be speaking a totally different language. But behind this barrier, there is a human being trying to get through to us. A human being with original thoughts, humour and love — all the attributes of the human race. However, deafness is the neglected handicap that mental handicap used to be. Much research needs to be done and modern electronics can help in the ways that have been indicated and in many others. Just to make people aware of the terrible handicap that deafness presents, is very difficult. Publicity regarding research into aids for the deaf assists in this process. Deafness suffers as a handicap by being invisible and therefore unnoticed.

There is the story of a leader of one of the New Commonwealth countries who, when asked how his country was dealing with the problem of deafness, replied that there was no problem. After all when he walked down the main street of his capital, he did not see any deaf people! Deafness cuts people off from people. Research and the use of modern electronics in particular can help the deaf to join us in the world of sound.

Book Review

Illustrating BASIC (A Simple Programming Language) by Donald Alcock. Published by Cambridge University Press. 1977.

Donald Alcock's dedication, to his sons, strikes the note that begins the tune of his book. The book is set out in what at first seems an eccentric way but later is refreshing to the eye and mind. The illustrations are, so to speak, in context.

Donald Alcock has not written a book for another expert to read but for students, engineers, managers and those who want to increase their knowledge of BASIC. He has taken the trouble to consult eleven manuals on different versions of the language. He goes step by step from the components of the language, to files of data, to the final chapter, which is on syntax. At the same time he does not serve up predigested mush. The reader has to do some chewing and digesting of his own. Chapters four and five on arrays and matrices will need careful reading. But Donald Alcock does everything to help, short of becoming the reader. Knowing the odour of sanctity Latin words have, he brings the word 'matrix' down to earth, giving both its definition and the meaning of the Latin — in plain English.

There are two complete example programs in chapter six. One is 'a program to convert Roman to more familiar numbers — this example demonstrates a symbol state table, a standard tool of programming.' The other is, 'Best Way Home. A problem common to industry and commerce.' The choice and treatment of these programs reinforces the content of previous chapters.

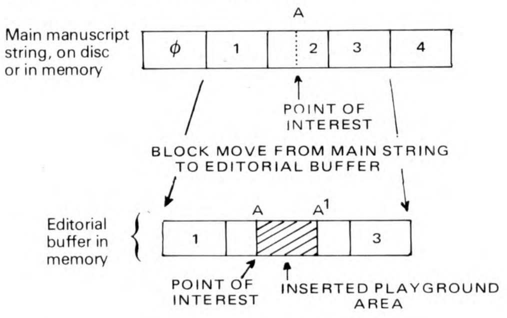
Chapters seven and eight deal with commands and files of data and are a neat introduction to a confusing aspect of BASIC. Chapter nine, on syntax, is difficult reading and Donald Alcock doesn't pretend it is not. All the same he has tried to present syntax to the reader as clearly as he can, which is very well.

Donald Alcock has written a book which should become a classic.

There are encouraging signs that British authors of educational books are getting the idea that learning is fun. Up to now this has seemed to be an American monopoly. But an uneasy thought comes to me. On page 49 of the book there is an illustration, a little flag. It looks like the Stars and Stripes.

PATRICK SUTTON'S ARTICLE VOLUME 1 No. 2. CORRECTED LINE DRAWINGS

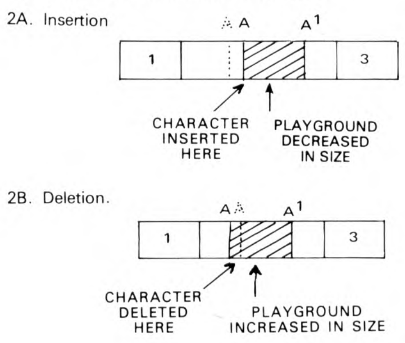
Fig. 1. Creating the playground area.



A
Main manuscript string, on disc or in memory
ϕ
1
2
3
4
POINT OF INTEREST
BLOCK MOVE FROM MAIN STRING TO EDITORIAL BUFFER
A
A1
Editorial buffer in memory
1
3
POINT OF INTEREST
INSERTED PLAYGROUND AREA

(A¹ is character initially adjacent to A)

Fig. 2. Insertion and deletion of characters



Error Messages

*This letter (from *Kit Spencer*, General Manager, CBM) is to correct certain factual errors in one of your readers' letters' in the last issue concerning the question of price of our PET computer.

Firstly, the price in the USA is not $595 but is $795 and the exchange rate referred to ($1.95) is in fact as of today's date only $1.81. The USA price is also exclusive of any taxes and shipping costs and therefore should be more closely compared with our U.K. VAT exclusive price of £643.52 and not £695.00. I am sure you will realise that the combination of all these errors makes the price comparison over 40% more favourable than it appears in your reader's letter. It is in fact lower than the normal $ for £ ratio experienced in the industry to allow for import duty, modification, servicing and other costs associated with bringing this product to the United Kingdom market.

I feel sure you will realise that the PET computer represents *outstanding value for money* compared with anything else on the British market.

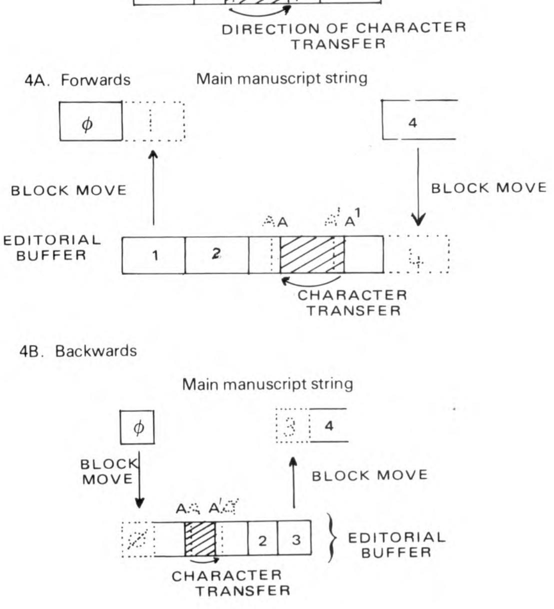
***E78 BUS SPECIFICATION (Vol 1, No 2)**

Owing to printing difficulties all the bars over signals, indicating active low signals or a logical not, were omitted from the text of this specification. The nomenclature in Table 1 is correct, and in most cases the meaning of the text should be obvious. The only case where the omissions may cause confusion is in the suggested logic for the 6800. As a result of a potential timing problem with 6800 peripheral devices this logic will, in any case, probably be modified, and anyone intending to use the 6800 on E78 is recommended to contact one of the authors. Logic has also been proposed for the 2650, watch these pages for further details.

*At £499, the TANDY TRS80 comes **complete** with video display and cassette recorder. The photograph of the TRS80 on page 37 of Vol 1 No 1 is the one on the right hand side.

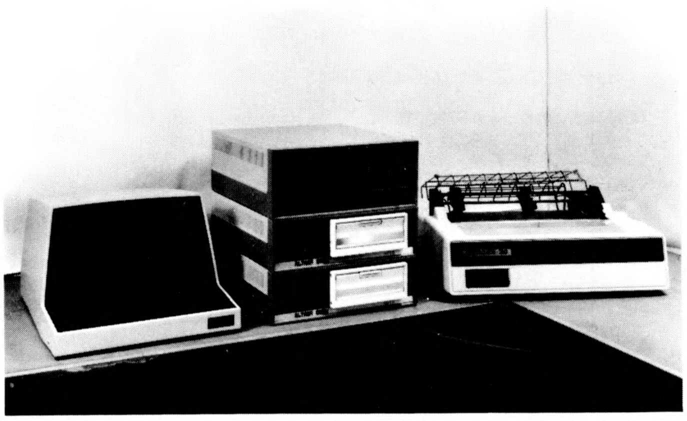
*Apologies to **Paul M. Jessop**, who wasn't included in our list of new contributors to the last issue. He is an experienced writer and potential contributors can only gain by noting the way he writes and lays out his articles.

Fig. 3. Moving the point of interest



compelec electronics

Welcome you to the Altair Age



THE MICROSYSTEM FOR THE PROFESSIONAL
(System 9V illustrated — £6,781 to £7,026)

We supply a complete range of low-cost Altair systems and software for educational, industrial and business use.

Main Frames
Choice of front panel or Turnkey
Memory capacity to 64K bytes

Bulk Storage Devices
10 megabyte disc system; standard floppy disc system; mini floppy disc system or cassette.

Interfaces
A variety of plug-in analogue, digital and industrial control interfaces supplied from stock.

Terminals
Soroc VDUs, Centronics and Qume line printers

Software
FORTRAN, BASIC, multi-user BASIC, disc operating systems, assemblers, editors, monitors, business packages, word processing packages, etc.
We supply fully commissioned Altair computer systems complete with terminals and software.

Telephone 01-328 1124 for a Demonstration

compelec electronics limited

Compelec Electronics Limited, 107 Kilburn Square, London NW6. Tel: 01-328 1124

COMPUTER WORKSHOP

BRITAIN'S LONGEST ESTABLISHED COMPUTER SHOP

HARDWARE

COMPUTER MP-68 with 4K Memory & Control Interface **Kit £275.00 Assembled £330.00**
MEMORY MP-4K Memory **Kit £70.00 Assembled £90.00** MP-8 8K Memory **Kit £170.00 Assembled £195.00**
DISC SYSTEM DMA1 1.2 megabyte twin system **Kit £1450.00 Assembled £1525.00**
MF-68 160K byte twin minifloppy **Kit £800.00 Assembled £860.00**
TERMINAL ASR43 Upper/Lower case 132 Char. 30 cps KSR **Assembled £915.00**
CT-64 Upper/Lower case visual display terminal **Kit £230.00 Assembled £315.00**
CT-VM Video Monitor for CT-64 **Assembled £140.00**
PRINTER RICOH RP-40 Daisy wheel printer 30 cps **Assembled £1800.00**
CENTRONICS 701 Dot Matrix printer 60 cps **Assembled £1400.00**
PR-40 40 column tally roll printer 55 cpm **Kit £200.00 Assembled £250.00**
CASSETTE AC-30 Cassette Interface for 2 audio cassettes **Kit £65.00 Assembled £100.00**
GRAPHICS GT-61 Graphics Terminal with P197 Power Supply **Kit £92.50 Assembled £120.00**
SUNDRIES MP-R EPROM Programmer for 2716's **Kit £36.00 Assembled £43.00**
MP-N Calculator Interface **Kit £37.00 Assembled £44.00**
MP-T Real Time Clock **Kit £32.00 Assembled £40.00**
MP-L Parallel Interface **Kit £30.00 Assembled £37.00**
MP-S Serial Interface **Kit £30.00 Assembled £37.00**
PPG-J Joystick **Kit £32.00 Assembled £40.00**
MOD1 UHF Modulator to link CT-64 to T.V. **Kit £4.50 Assembled £6.00**

PRICES DO NOT INCLUDE V.A.T. or carriage.

SOFTWARE
Software includes 3K†, 4K and 8K Basic*, Coresident Editor/Assembler, Text Editing System†*, Text Processor†*, TSC Assembler†*, Games and Scientific packages.
* Disk Based versions available.
† Full source listing included.
Prices vary between £5 and £20.

Both Disk Systems come complete with the comprehensive 'Flex' Disk Operating System. Commands include -

APPEND	COPY	LINK	RENAME
ASSIGN	DELETE	LIST	SAVE
BACKUP	EXECUTE	MONITOR	STARTUP
BUILD	GET	NEW DISK	VERIFY
CATALOG	JUMP	PRINTER	VERSION

Applications include: **• Travel bookings • Word Processing • Questionnaire analysis • Process control • Teaching • Accounts analysis**
Existing customers: **Schools — Universities — hospitals — M.O.D. — business — industry.**

Please write for full details to:-
COMPUTER WORKSHOP, 38 DOVER ST., LONDON W1
Tel 01-491 7507