PERSONAL COMPUTER WORLD

VOLUME 1, NUMBER 2   1978
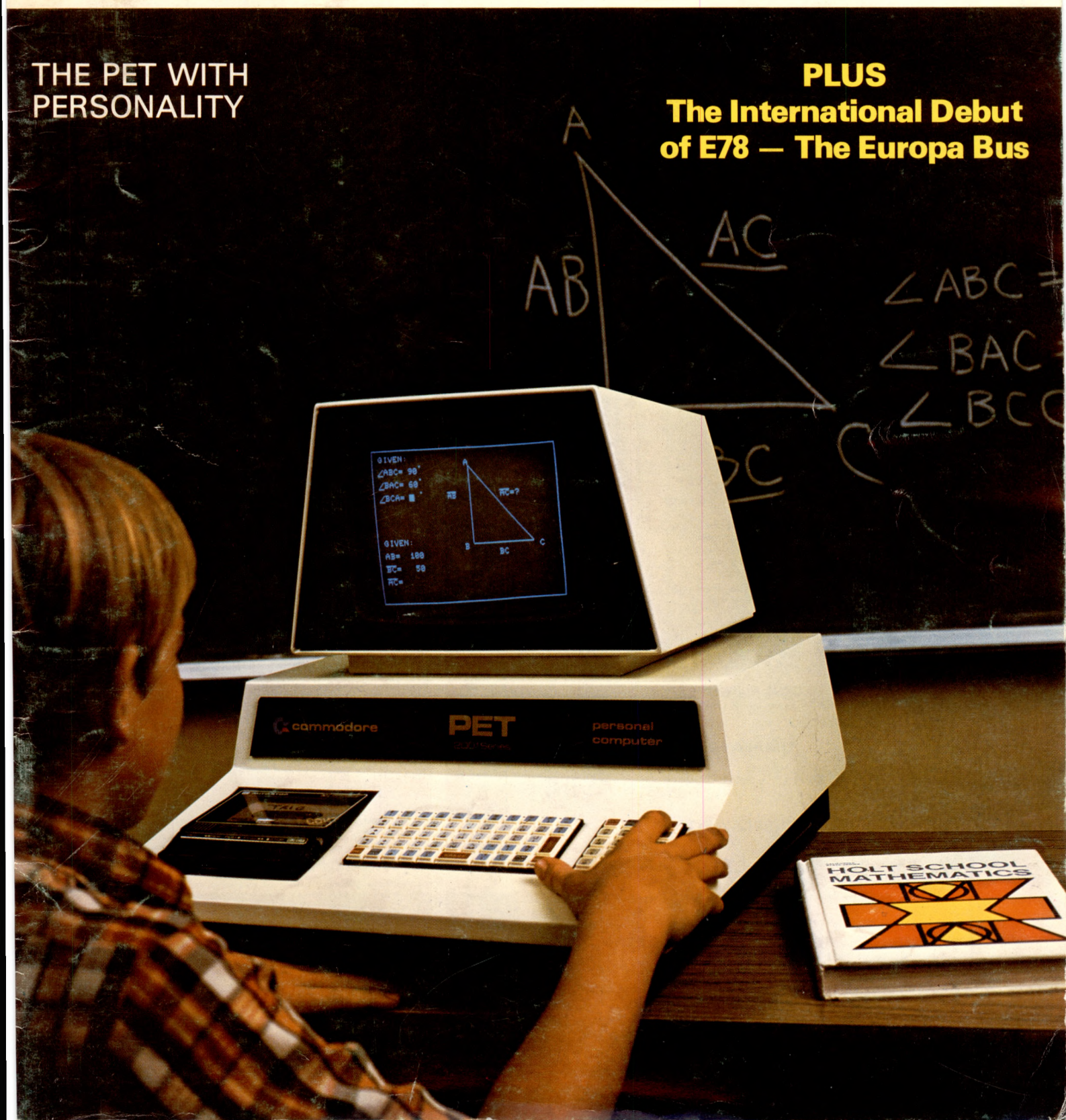
50p   U.S.$2.00

— **BASIC FOR BEGINNERS**
— **SCRAMBLE!**
— **FOUR EASY PIECES**
— **POWER TO THE PEOPLE**
— **COMPUTER IN THE CLASSROOM**
— **PCW REVIEW: RESEARCH MACHINES 380Z**

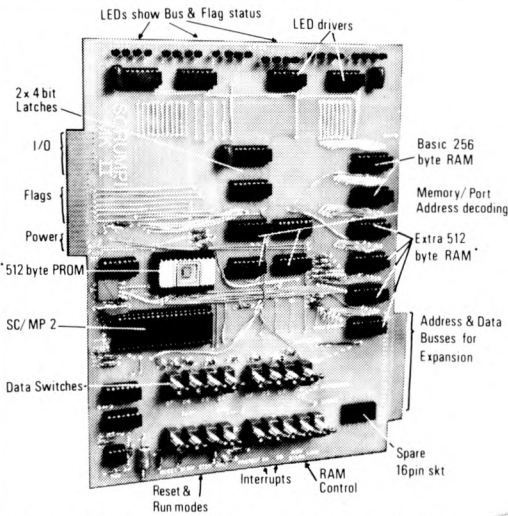**Europe's first magazine for personal computers for home and business use**

THE PET WITH
PERSONALITY

**PLUS
The International Debut
of E78 — The Europa Bus**

PERSONAL COMPUTER WORLD

UK 50p
US $2.00

Vol 1, No 2
1978

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

CONTENTS

DOTPIKS © K. HAMME
CARTOONS © GIGO & SAM

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.

Publisher's Letter

Dear Reader,

It is exciting to see that we were right in thinking that there is enough interest in Britain and the Continent for supporting a magazine devoted to personal computing. At many newsagents we were a "sell-out" within the first few days. The information we've had so far leads us to expect a sale of about 25,000 copies of the first issue. This includes the remarkable number of 3,000 subscriptions to date. Quite an impressive result for a small publication.

Your support enables us to make the magazine bigger and better.

The first result is more pages and more articles.

Second, we are going monthly: issue four will appear a month after issue three.

Third, we are at an advanced stage of organising our **own personal computer exhibition** — in September of this year. We hope that the exhibition will show that the personal computerist in Europe has come into his own. Many companies in this country have already pledged their support and some really important people involved in personal computing in America have promised to give lectures or exhibit.

Other ideas we have will be developed and put forward in future issues.

We take this opportunity to thank some of our American friends for the hospitality and valuable advice given to us during our visit to the United States. Nice people like **Wayne Green** and the staff of *Kilobaud;* **Leslie Solomon** of *Popular Electronics:* **Stanley Veight** of Computer Mart, New York; in particular **Virginia Peschke Londner**, publisher of *Byte:* and her senior editors **Chris** and **Blaise** and the circulation manager, **Greg**.

We were sorry not to have met **Carl T. Helmers**, the editor of *Byte,* who since its start in September 1975 has played a major role in making *Byte* the biggest and most influential magazine in the personal computing field — but we did manage to bring away a photograph, published below.
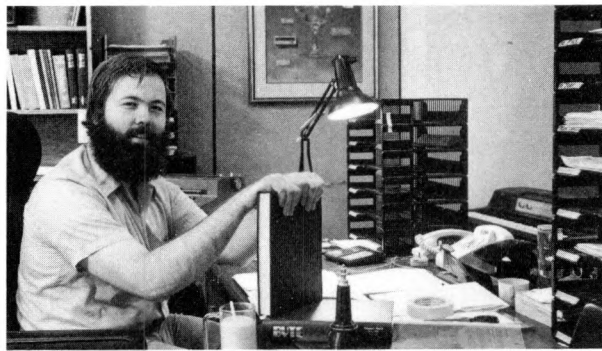
FOR PCW READERS IN NORTH AMERICA

From issue No 2 Personal Computer World will be available at over six hundred computer and electronics stores and news stands in the United States, Canada, Mexico and the Caribbean.

The world's leading publisher of small systems publications, **Byte Publications Inc. of Peterborough, New Hampshire**, are now our North American distributors.

We share their belief that with them we will be able to achieve a wider circulation in that major area and inform American readers of what is happening in Britain and the Continent.

All dealer enquiries should be addressed to:
**Byte Publications Inc., 70, Main Street, Peterborough, NH 03458
Phone: (603) 924-7217**
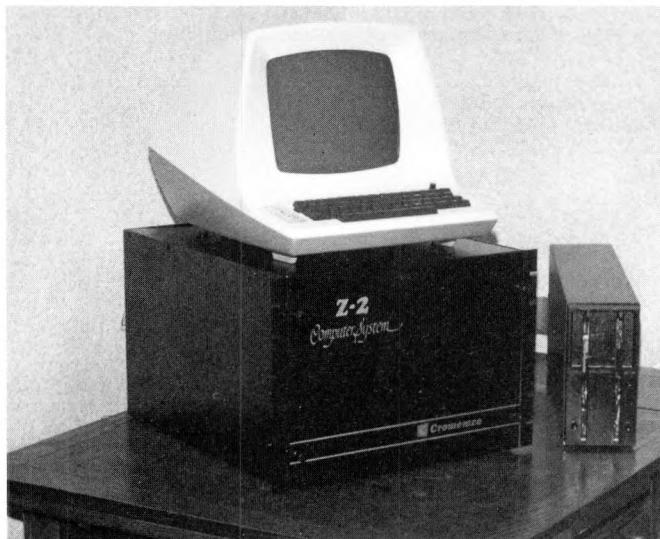
Editorial Page

Progress Report

Many readers will know that *PCW* was first on sale at the Microsystems '78 exhibition held in the West Centre Hotel, London. Ours was one of the busiest stands — a complete success. Then the pattern repeated itself at newsagents. Thanks to you, *PCW* is now on solid ground. An idea whose time has come is rare enough; an idea that makes its own time is even more of a rarity. *PCW* has not only been welcomed by people in computing but has also created a new awareness.

With plenty of help, of course. Which leads me to how some other magazines regard *PCW!* We asked some of them to take our advertising. Reasonably enough they nearly all refused, because *PCW* covers "some common ground; we'd be helping a competitor". An exception was *Electronics Today International,* which welcomed us in its pages. It may or may not be a coincidence that *ETI* is among the livelier publications around. As is *Computing,* from which we reprint in this issue Dr. Bob Johnson's article on the urgent need for the computer in general practice. We're pleased also that Guy Kewney, editor of *Computing*'s personal computing page, writes for us again; the man has style. Talking of magazines, I did not mention the American computing magazine *Interface Age* in the last issue. It now gives me genuine pleasure to recommend it to readers interested in the American scene.

Our Competition

Entries for the *PCW* competition range from the general to the technical and are all of them of a high standard. The first prize is, as most of you know, a NASCOM 1. It's coming to us in kit form. Whoever assembles and tests it will be asked to give an account of how he or she got on.

CASBA

We're getting the first signs that the idea of the Computer Association for Small Business Applications is turning out to be viable as more and more small businesses see the value of *PCW* as a suitable vehicle for their aspirations and objectives. The Association would be invaluable in all kinds of ways: it could put forward its views to manufacturers; share and so lower the cost of software originated by its members; members could give the benefit of their expeirence to those just starting and could even help out others if they have spare capacity; any more ideas?

PCW Consultants

Our consultants are pretty independent people — you've only to look at the articles by John Coll and Mike Dennis in this issue. If I wrote an article on the PET I would praise CBM mightily for its efforts to bring computing within reach of all of us. I would be similarly enthusiastic about any well designed personal computer (like Tandy's TRS80). Our consultants' role is to ensure that *PCW* is the result of a balance of viewpoints.

Take one spoonful every issue

Among the viewpoints are those of *PCW* readers. We take our own medicine: if we're criticised in a reasoned way, and if we've the space, we print. Intel have written us a letter — it's precise and fair.

We're thinking of having a newsletter incorporated in *PCW.* It would be called *Personal Confusion World.* It would list booboos by *PCW* (being a marvellous read is not the same as being perfect) and also amusing mistakes from other publications or pronouncements. Any contributors?

New Contributors and Consultants

Harry Harrison is a superlative writer of SF. His *Stainless Steel Rat* series has brought entertainment to many thousands of SF fans, and that includes me. A new book, *The Stainless Steel Rat Wants You,* is coming out this year. Philip Couzens teaches at Oundle.

Dr Robert Johnson is doing a PhD at UMIST on the real time retrieval of clinical data. He is a general practitioner.

Bill Ringer works for a large computer manufacturer in one of its product development divisions. Pat Crowe is a senior engineer in a large electronics company. His homebrew system is Z80 based. He's at present writing a Z80 assembler. Bob Cottis is a manager at the Fulmer Research Institute, doing contract research on corrosion and fatigue. He's currently building a Z80 based system using an IBM Selectric for output. Dave Howland is a senior engineer in a large British firm and is concerned with telecommunications. He's been using the 6800 for a year, starting with a D1 kit, and now has a homebrew system.

Martin Collis is a graduate of Imperial College, London. He's a principal of Applied Systems Techniques Ltd. which at present is mainly involved in general commercial applications in DEC and Hewlett Packard equipment. Graham J. Trott works for a large Government concern. He stays in late, much to the disgust of his wife, developing games and other programs. He's designed and built his own M6800 system, using Eurocards.

Patrick Sutton has joined us as a consultant. He's in the Department of Physiology, University College, London, doing a PhD in the applications of microprocessors and minicomputers to medical research. His article in this issue argues for the revival of elegance in computing.

Boris Sedacca is a research editor for *Which Computer?* magazine, a superb publication that's mailed direct to people in computing. Nick Toop is at Cambridge University. He's doing research into black holes. Kit Spencer is CBM's man in Britain. If he has anything to do with it the PET will become a household word.

Jeff Lynch of Xitan Systems has been in data processing for ten years, and in microprocessing for two. He's the Comart distributor on the South Coast, and specialises in marketing complete Z80 systems.

New *PCW* consultant **Michael James** trained initially as a mathematical physicist, and is just completing a PhD in pattern recognition, working with new parallel processing computers.

W.G. Marshall works for a large company. He shows, in this issue, how fundamentally simple and handy the SC/MP microprocessor is and implies that it has been rather unjustly neglected.

New *PCW* consultant **R.W. Davy** graduated from Cambridge and Aston where he is currently a research fellow in the Department of Engineering. He's written a cross-assembler, assembler and basic compiler for the Intel 8008 used initially by the microprocessor group in Aston. The group's current interest is the Texas Instruments TI99 series of processors. He was responsible for the manager for SDL's Coral 66 compiler for the 990/10.

Letters

Questions of Price

I first saw the PET computer at the 78 Micro Systems exhibition but I had read of it earlier in the September/October 1977 issue of *Personal Computing* an American magazine who made a feature of the new hardware as you are to do in number 2 of Personal Computer World.

However there is one significant difference which is the price quoted for this new device. In USA the price is $595 which at the current rate of exchange of $1.95 is about £305 whereas the price asked in this country is £695 which is more than twice the price in the USA where the disposal income is very much higher.

Of course there are many reasons why the price should be different in the two countries ranging from technical considerations like a different power pack for the mains in this country, different levels of taxes and transport costs. Never-the-less I would suggest that by no stretch of the imagination could this account for the discrepancy in prices. Perhaps Kit Spencer or someone else from Commodore could explain this apparent anti-British bias.

In the same issue of *Personal Computing* Radio Shack's announcement of TRS-80 is reported at $399.95 (£205) for the computer/keyboard or $599.95 (£308) for the full system including video monitor and cassette recorder. Will Tandy be giving a fair deal to British computer users and potential users?

Terence F. Hobson
44 Milton Avenue, Eaton Ford,
Huntingdon, Cambs, PE19 3LE

ps Why is the US price of *PERSONAL COMPUTER WORLD* not $1.00??

PCW Because of postage rates, handling and distribution costs.

Is there a Computing Club in Aberdeenshire?

Do you know if there are any clubs, organisations or similar minded people in my area (ABERDEENSHIRE) with whom I can get in touch for advice with my computing problems? I think not myself, but would appreciate any information you can find.

R. Duncan
2 Whitehills Cottages, Monymusk,
Inverurie AB3 7SS

A Brickbat surmounted by a Bouquet

Having only recently become interested in personal computers, I purchased the first issue of your magazine (inspired by its recommendation in ETI). I found, not surprisingly, that it was filled with ways for me to spend my money, but not many ways in which to save it.

I am not rich, but I do have a television set, a cassette recorder, and the ability to use my hands for construction. I also feel sure that I could find an old electric typewriter keyboard in Olivetti's rubbish bin or somewhere. So you see, all I need now is the most important item, the computer!

What I am saying is that whilst I don't expect you to recommend the construction of computers from rubber bands, paper clips and chewing gum to hold it together, I do feel that you should try to bring the computer to the most number of people, and this can obviously be done by reducing the cost as much as possible.

It was therefore disturbing to find that at least 5 of your 17 contributors had vested interests in the personal computer business. I don't dispute their knowledge of the subject or the facts they present, but I take their recommendations and opinions with a large pinch of salt. I think that if you must publish articles by these people, it should be made clear under the banner of the article what their interest is.

Finally, let me congratulate you on what looks like being an excellent publication (surprised you, didn't it?) which I shall continue to buy.

G. D. Cohen,
75 Munster Rd.,
London, S.W.6.
PCW *We* do *clearly state the "vested interests" of writers, in the magazine, when we give details of our contributors. And, Research Machines of Cowley Rd, Oxford, rang us just before we went to press to announce price reductions of up to 20% on their systems. They must have had Mr. Cohen in mind!*

Dear Sir,

I have purchased and read the first issue of your magazine PCW. I hope that you succeed in your aims. May I as a newcomer to the computer scene offer some comments, (if you say no, you still get 'em anyway)?

As a reader of *ETI, Practical Electronics, Practical Wireless, Elektor, Wireless World,* and *Radio Constructor* (among others), I find that the "buzz-words" of the computer fraternity confusing. As a 'frinstance, I read an article and interspersed are new words and their abbreviations i.e. UART, QWERTY, NADS, etc, etc, the explanation is in the text. So far so good. Further on in the article I come across the same abbreviation, I've forgotten what it means so I hunt for the explanation, by the time I've found it I've lost my place in the article, frustrating when a whole new vocabulary has to be learned. Why not put a glossary at the end of the article in which new words appear? Then if a series of articles is produced the reader knows that it's the glossary to look at and not have to search in the article.

When a new magazine that interests me appears on the market I always buy the first six issues, I think that it takes that time for a new magazine to get into its stride. I shall pay up to $1\frac{1}{2}$ hours pay for a good home computing magazine (that's roughly £1.30), but I always reckon in hours worked to buy anything.

One of my other interests is Aviation, and I get most of the magazines. If I write to *Flight* I always get a little card that says "Your communication of XXXX has been received, thank you", nothing ever published but they *do* known that I am alive and well. Electronic magazines readers are people who pay the costs and profits. The only reply I had from an Electronics mag. took four months, I had forgotten what I had said. Another journal that I have written to once, *Practical Photography* (another hobby) answered in five weeks which I think is a reasonable time; after all, there is a magazine to publish.

So now I don't write to Electronic magazines at all, I just buy them, but there are two which I am seriously thinking of cancelling.

J. Turner,
63 Millais Road,
London, E11 4HB
PCW *With a name like Turner and address at Millais Road, this could only be a graphical letter!* **PCW**

Dear Sir,

I enclose a booklet (A History of Manchester Computers, *S. H. Lavington,* NCC Publications) I feel you ought to have. You, and Guy Kewney whose article I enjoyed, may like to check this booklet against the article and the Summary of computer history on page 15, of PCW.

As a Mancunian I feel very strongly that the evidence is good supporting the claim that the World's first stored-program computer ran a program successfully in Manchester in June 1948. Edsac was not running until a year later in Cambridge and in its turn was a year ahead of corresponding American developments.

So, although we may not be very good at backing and developing commercially new ideas, the new ideas are there and are usually brought to a high stage of effectiveness by very few people, instead of large teams. In particular it is worth reading the last sentence of paragraph one on page 32 of the booklet.

G. D. Dawson, M.Sc., M.B., Ch.B.
Department of Physiology,
University College of London,
Gower Street,
London WC1E 6BT

PCW *The booklet is crammed full of interest.* **PCW**

Intel and PCW

Dear Sir,

I read Guy Kewney's article "Past Procession" in your first issue with interest and was very surprised to note a number of serious errors. I am writing, therefore, to draw your attention to these and to provide you with the correct information.

The story of the introduction of the first microcomputer, as described in the article, is totally inaccurate. In 1969, a calculator manufacturer approached Intel with a request to design a custom chip set for a range of electronic calculators, and the idea of producing a computer-like architecture on silicon chips was conceived. The result was the 4004, a four-bit word length microcomputer which did everything that was required of it. In fact, the 4004 is still in large-scale production for a diverse range of applications and, to date, very large quantities have been produced.

Intel went on to develop the first 8-bit word length microprocessor which was designated 8008. This was also a very successful device (contrary to your writer's unsupported claim), is still in production and, again, very large quantities have been produced.

Later, more powerful versions were introduced — the 4040 and the 8080. The latter device quickly became the industry standard microcomputer. It is difficulty to know what the author of the article means by "wretched instruction set", since he does not qualify or justify this statement in any way whatsoever. Perhaps Mr Kewney might like to present some reasoned engineering arguments to explain what he means. His views are obviously not shared by the thousands of professional engineers who have selected the 8080 for their design projects. The most successful hobbyist computer of all — the Altair 8800 — uses the 8080.

I am sure Motorola would be horrified to learn that they had developed their 6800 "by upgrading the 8080". The 6800 has an entirely different architecture to the 8080 and in no way is one processor a development of the other.

As a matter of interest, and to bring the story of microcomputer evolution up to date, a more powerful 8-bit processor has been added to the Intel range — this is the 8085. In addition, Intel have introduced a whole family of single-chip microprocessors with up to 2K bytes of program memory, 128 bytes of RAM and 27 I/O lines on one chip. They are the 8021, 8035, 8039, 8048, 8049 and the 8748. Very soon we will be announcing a 16-bit microcomputer which is similar in power to today's minicomputers.

Returning to the article; obviously, one of the reasons why humans employ computers for arithmetic is simply speed — and not precision as mentioned in the article — since humans and computers (even the 4-bit 4004) can work to any required degree of precision.

Any book on data base management will explain how, in a properly organized data base, to avoid the necessity of searching all records in file to find one at variance with the rest. Readers interested in this subject can do no better than obtain a copy of James Martin's book "Principles of Data Base Management" (Prentice Hall).

I would like to take this opportunity to wish *Personal Computer World* every success, since there is a need for a UK based publication of this type. It is, however, vital that the editorial staff should check the accuracy and validity of the material that is published. By all means criticize — this is an editorial prerogative — but make sure that such criticism is fair and reasoned in addition to being supported by sound engineering arguments. Rumours and hearsay have no place in any technical journal.

Keith Chapple,
Manager
Intel Corporation (UK) Ltd

Microprocessors and the disabled

Dear Sir,

I have enjoyed reading the first issue of your excellent magazine. I was interested to read a short item on page 5 referring to the possibility of microprocessors being used in aids for the disabled, and I thought you might be interested in a few of my comments on this subject.

I am a teacher at a school for the physically handicapped and one of the problems I often come across is that of the intelligent child trapped within a spastic body, unable to communicate other than by grunts, unable to move about independently, unable to control his or her environment, and unable to learn about the world by experimentation. Of course, aids are available for such children, but they all suffer in that they have evolved from equipment designed in the '60s to answer very particular needs.

An example is the POSM (Patient operated selector mechanism) which was a system which allowed patients paralysed below the neck, due to polio or accidental injury, to exercise some control over their environment. It incorporated a suck/blow type of switch and electromechanical devices to select and enable various mains-operated peripherals, e.g. light, radio, TV, alarm bell, etc. Later the selector mechanism was used in conjunction with an IBM typewriter with a solenoid bank strapped under the keyboard and with a variety of other switches suitable for the very fine or very gross movements produced by those afflicted by a large range of handicaps.

There are many marvellous stories of people whose lives have been transformed by the POSM typewriter and environmental control systems, and by the telephone equipment developed in conjunction with the Post Office. There are several successful computer programmers at work today who had been "written off" as subnormal when young children, because they could not communicate by themselves.

However, when we consider whether the design of such systems has kept pace with the fantastic rate of development of microelectronics we have to conclude that the 1978 models do little more than the 1968 models did, they are still designed around the solid-state equivalents of relays and uniselectors, and they actually cost more!

A remote controlled typewriter system now costs £1,800. The display board alone, consisting of fifty or so light bulbs fixed behind translucent perspex panels, can cost more than a NASCOM 1! How much more useful a versatile microprocessor-based system would be to the handicapped child. It is exciting to visualise a disabled child with a micro-computer like Apple II, with facilities for colour graphics, dumping onto and retrieval from cassette tape and a hard-copy print-out, all controlled by one or two "knock switches". The system would be the child's typewriter, note pad, reading book, picture book, dictionary, calculator, sketch pad, games compendium, music synthesiser and exam paper, all rolled into one. In the not too distant future it could literally become the speechless child's voice.

There are but a few groups in this country working on such systems, to help specific children or to equip particular schools, and they can only help a minute percentage of those who could benefit. Is it reasonable, however, to expect teachers, governors and administrators with no knowledge or experience of microcomputers to appreciate the opportunities that are being missed? Until the educational equip-ment firms jump on the microprocessor bandwagon it is up to the individual enthusiast to try to convince those who control the purse strings that a small box plugged into a TV set, costing perhaps £500, is far better value than a well-tried and well-known device, costing three or four times as much, but coming from a firm with a household name.

With hardware becoming cheaper and more readily available by the month, it is the need for specialised software routines which is becoming the major problem. Whilst many amateur electronics enthusiasts will feel confident enough to construct a microcomputer kit, thereby saving a considerable amount of money for a school, or the family of a handicapped child, few will be able to tackle the daunting task of writing a graphics program with the special editing features which may be required.

I am sure that it is here that the personal computer enthusiast will find an opportunity to make a unique contribution to help one of the sectors of society which might benefit the most from this exciting new technology. I feel that the bodies concerned with the physically handicapped will be slow to acknowledge the possible contributions which personal computing can make, and it is from the direction of the hobbyist that the most productive moves can be made. Head-teachers in schools for the handicapped cannot fail to be impressed when they see a system running demonstration programs and the contagious enthusiasm of the hobbyist is bound to be shared by all involved.

I hope that you find some of these ideas stimulating. I should be very pleased to receive any further comments on this subject and would be happy to correspond with anyone who has similar interests or problems to solve.

The eyes of the staff of our special schools must be opened to the opportunties afforded by this new technology and it is through the efforts of people like your readers that this can best be made to happen.

Paul J. Grant,
Valence School,
Westerham,
Kent, TN16 1QN

May I congratulate Mr. Zgorelec on his initiative in publishing Personal Computer World, and you, sir, on the excellence of the first issue.

P.B.E. Thompson,
17 The Grove,
Woking, Surrey.

Many congratulations on being the first in the field of personal computing this side of the Atlantic. Of the articles in the first issue, it goes without saying that Direct Addressing was the most valuable for its information, and the Benchmark programs were fascinating for comparison purposes — the speeds are about two orders of magnitude better than the best programmable calculators. (Although it would seem from BM4&5 that the PDP8E has negative sub-routine access time!)

Basic Pontoon was a particularly stimulating article, and I now have a version of this on the Texas Instruments' TI59/PC100A. The subroutine for shuffling the cards caused some concern however, for two reasons: 1) it was time-consuming having to
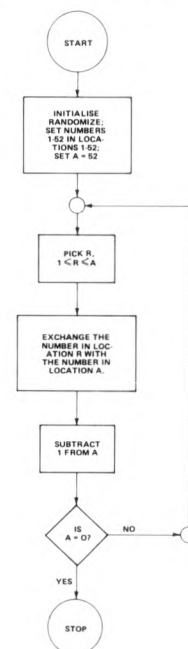
look at all the cards already ordered when attempting to choose the next card, which it would then often have to reject. 2) The shuffling was not properly random. For instance, if the first three cards to be chosen were aces, the last ace was still as likely to be the fourth card as all four kings taken together. The attached figure is my attempt to rectify these points.

The cards in the pack correspond to the numbers 1-52, so that the routine may also be used in games where the suit cannot be neglected.

John Murrell,
90 Pirniefield Place,
Edinburgh.

Olly's Originality

Dear Ed — I have been fascinated by the idea of building a computer and have no experience apart from a two year dabble in Fortran and Cobol as a youth. However, our immediate need is a hifi amp. Knowing nothing about electronics allows me to ask whether it would be possible to combine the two for the sake of rationalizing the expenditure? A more realistic question would probably be: are there any people interested in computers in the wilds of Norfolk?

Olly Cooper,
School House,
18 London Rd,
East Denham.

POWER TO THE PEOPLE

A view of personal computers and their possible effects on us

W. V. Ringer

How does a man boost his ego except through the things he controls and owns? Some would question the value of an inflated ego and suggest that the opposite might be more desirable in the end. However, we live in a world predominantly favouring aggression, pride, nationalism, acquisition, power and so on. The advent of the microprocessor and one of its many siblings, the personal computer, is going to have an unprecedented impact on all our lives and there is every indication that the personal computer is going to prove to be the most far reaching extension of the person yet devised. Psychologists will say that people demonstrate their earning and spending power through their houses, cars, holiday movies and so on in order to indicate to the world what their personal capabilities are. Their bookshelves and record collections are expressions of their tastes. Will the personal computer become the new status symbol and will the range of collected computer software and hardware be a new way of expressing personal capabilities and power? Whatever the consequences of the personal computer turn out to be, we can expect that they won't be trivial. This article will try to show that these consequences could lead to: changes in public attitudes to computers; the evolution beyond literacy and numeracy to include 'logicacy'; the possibility of safeguarding individual freedom from 'database tyranny'; the influencing of design from networks to keyboards; the unleashing of the creative potential of the masses; and finally the development of new methods of programming to satisfy the human need to understand complex programs.

Ownership is a very powerful motivator. The computer will become a personal thing, something to do with 'us' rather than something to do with 'them'. We will see computers as tools, available for all to use. The subject of computers will be demystified and the priests of computing, the operators, programmers and systems analysts, will simply be doing a job professionally that we all will be doing privately. What is it that will bring about these fundamental changes in attitude? In all significant turning points in history there is a sudden gelling of certain factors that have been gathering for some time. In the case of the personal computer, one of these factors is especially interesting, namely the subtle change in emphasis of computer ownership. Owners behave differently from users. Until recently, only organisations could own computers and individuals were merely users of the facilities. Now that individuals have the opportunity to own computers, a change can be predicted in the us-and-them relationship which has hitherto existed between the man in the street and the computer experts. Owning and using a computer will become no more unusual than owning and using a pocket calculator, a hi-fi set or a motor car.

A characteristic of Western civilisation is the almost universal literacy and numeracy of its peoples. The achievement of these skills has been helped by technological advances. As a consequence of the microprocessor technology, we are now about to witness another evolutionary step forward. Our future generations will not only have the ability to read, write and perform arithmetic but also to express relationships and solve problems through languages of logic. Most commonly, everyone will be familiar with at least one programming language. Beyond that, in the borderland of science-fiction, is the personal computer that will interactively discuss a problem and from the responses generate a program. Until that day arrives, however, we had better learn to be programmers ourselves. Only through the widespread popular understanding of the languages of logic can we protect our personal values and rights from the encroachments on those freedoms resulting from programmed implementations of laws and procedures by narrow-minded technologists. An individual's freedom is determined, to some extent, by his ability to access information. Libraries contain predominantly generalised information. There is no difficulty in accessing this information if we wish to. Government departments, employers, banks, building societies, department stores etc., all hold personal information about us. The personal computer provides the physical capability of accessing these various databases. Whether or not we can persuade or require the database owners to allow access for examination purposes is a matter which has yet to be resolved. The whole area of rights concerning interrogation and exchange of information is being tentatively approached by the government, the British Computer Society and other bodies under the heading of Computer Privacy. There is still a long way to go and the complexities of the problem are growing daily. The widespread use of personal computers will, no doubt, add a further degree of urgency to its resolution.

The personal computer without access to networks is like a horse without a carriage. Microprocessors will facilitate the construction of 'intelligent' communication networks at a variety of levels, from local to international. Microprocessors

will also facilitate the interfacing of numerous devices to the networks by their abilities to handle the appropriate protocols required on the networks, independently of host computers. The interaction of private, public, high-speed, low-speed, digital, voice, packet-switching and other network types is already causing an urgent search for national and international agreements on interfaces, speeds, protocols etc. The entry of personal computers into the networks will no doubt push the need for standards even more.

Significant advances

The market requirements of millions of personal computer users are bound to have an effect on the design and development of the most heavily used input-output devices. The most practical man-machine interfaces are of the screen-to-eye and finger-to-keyboard kind. Perhaps we'll all end up with square eyes and hammerhead fingertips. Already cathode-ray tube radiations are causing health concerns. This, coupled with the relatively high unit cost of CRTs will force the need to develop cheap, flat-screen, lower power, non-radiating displays. A satisfactory alternative to the eye and finger problem would be the use of the ear and voice faculties instead. Significant advances have been made with voice recognition devices and speech synthesizers. It is only a matter of time before they are freely available. For the moment the keyboard is here to stay, and this brings with it the problem of keyboard layout. We all wish to use keyboards efficiently. There is nothing more irritating and time consumer than keying-in our creations with one finger at the rate of one character per second after scanning the keyboard for each character and then scanning the source document to find the lost place. Many of us, with a bit of practice, are better than that, but we still envy those top-flight data preparation operators who can key at sustained rates in excess of 18,000 key depressions per hour. One solution, of course, is to learn touch-typing. The organisers of such courses may need to expand their facilities considerably. The anti-QWERTY keyboard pressure group might begin to feel desperate at the possibility of QWERTY being set in concrete for ever once the public at large has got used to it. Or is there perhaps another way out? New developments in keyboard design might provide a solution to this old problem.

The variety of uses to which the personal computer can be put is limited to some extent by the availability of ancillary equipment. The personal computer will follow the path already pioneered by the mini-computers. Computer peripherals will be designed and manufactured to cater specifically for the personal computer market. The limiting factor in reducing costs is that production and assembly of electro-mechanical components is more expensive than the automatic mass production of large-scale integrated circuit chips. The technology limitations at the moment are in the electro-mechanical field rather than in the logic field. The multitude of possible applications for the microprocessor are being held back by the lack of equivalent advance in the development and free availability of suitable sensors, transducers etc., at the input end and electro-mechanical robotics at the output end. In the area of the home alone, the wish of personal computer users to control their heating, lighting, telephone, domestic appliances and so on will require a response from the manufacturers of electro-mechanical controls and robotic-like servo-mechanisms. A valuable spin-off will result from hobbyists designing and building their own control and servo systems, thereby contributing to the wealth of unfunded private research. The best of these developments can ultimately be exploited and produced in volume for the benefit of all.

Is it possible that one day the pastime of television watching will have to compete with the attractions of the personal computer? Some are concerned about the effects of increasing leisure time. While many of us are content to the passive spectators, there are many others who would like to do something creative but are not prepared to go too far from the armchair. The program for a microprocessor which will control the movements of an invalid wheelchair, for instance, might be developed initially on a personal computer. There will be scope for thousands of individuals to be *creative in an abstract way for practical ends.*

Human Comprehension

In the future, programs should be written like good essays, so that they can be read and understood in an effortless way. Current program languages do not allow this because they have evolved from earlier languages which were designed for the convenience of the computer. A method of writing programs is needed which will enable the reader of a program to understand its purpose, objectives, limitations, exceptions and so on at all levels within the program. A program should be written with human comprehension as its prime requirement. A program generator, rather than a compiler, would produce the appropriate executable machine code. The ability to comprehend, assess and criticise a program is going to become a very necessary skill. The proliferation of available application programs for personal computers will bring the need for this skill forcibly to our attention. If these new programming methods are adopted, then we will be able to interrogate a program structure and understand why it works in the way it does. If we cannot do this, there is again a danger to our personal freedom. There will come a day when almost every aspect of our lives is affected by a computer program of some kind. We will allow this to happen because it will be very efficient. For instance, it would be pointless to work out manually each time how many miles are left in the petrol tank when the on-board microprocessor presents the appropriate result continuously. We will be conditioned to trust our computers and therein lies the danger. If we observe a program displaying undesired characteristics, we shall wish to examine the situation in greater depth. We will require an efficient method of interrogating the very principles and assumptions behind the logic of any program at every level. We shall wish to do this for programs under our control as well as having the right to question the logic behind the programs used by government and other authorities. We are able to examine the laws of the land and question their implementation. Should we not have the same facilities regarding the programs of the land? By 1984 we will surely need them!

"I know you're a personal computer but this is ridiculous."
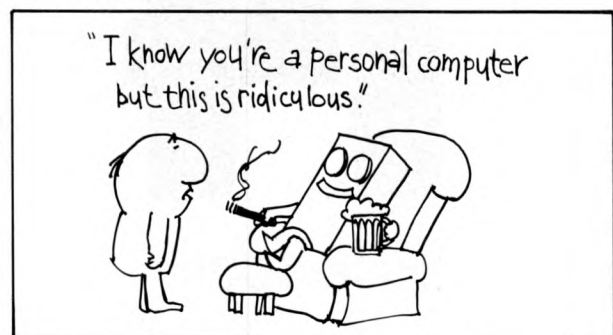
A MIGHTY MICROMITE THE MK 14

NICK TOOP

Introduction

I imagine that there are many people like myself who have an interest in home computing but are put off by the cost. The MK14 kit, produced by Science of Cambridge, makes up into the cheapest complete home computer I have yet seen. The cost is less than that of the ICs when bought individually. The kit enables anyone proficient with a soldering iron to build a complete home computer. Since a voltage stabiliser is included in the kit all that one then needs is a six volt power supply capable of delivering an amp. The basic kit enables one to enter programs written in machine code through a hexadecimal keyboard and to show the results on a calculator type display. There are also several channels for control applications.

Hardware and Software

The kit is based on the SC/MP2 microprocessor. This is an eight bit device made by National. The chip and all the necessary support devices are mounted on a single piece of printed circuit board which has an edge connector at each end. The entire unit is small, light, and robust enough to fit inside other equipment which makes it ideal for experiments with small mobile robots and other types of control devices.

The integral hex keyboard is of the type used in the cheaper type of calculators. It is my experience that this could eventually give some trouble but I do not think the manufacturers have any alternative here. A really reliable keyboard is a complex bit of engineering and is correspondingly expensive. The extra expense is probably uncalled for with the beginner user. The keyboard is operated in a matrix fashion. There are ten columns corresponding to ten consecutive addresses and four rows corresponding to the four upper bits of the data bus. The keys connect between rows and columns. When a key is depressed the bit corresponding to its row on the data bus will go high if the address corresponding to its column is on the address lines. Since there are contacts to both the columns and the rows on one of the edge connectors it is a trivial matter to add a more reliable keyboard with up to 40 keys at a later date. It would also be very easy to experiment with other types of keyboard using touch sensitive ICs or fibre optics.

The eight digit seven segment display includes decimal points. The eight digits correspond to eight special addresses and the corresponding data determines which segments are lit. It is thus straightforward to display a wide range of characters.

The 512 byte PROM contains a monitor program which enables one to examine the contents of the memory on the display and to enter new data through the hex keyboard. It also enables one to start execution of a program located anywhere in memory. While the monitor program does not support single stepping, which is very useful for debugging programs, a routine for doing this is included in the training manual. The monitor does not include routines to store and reload programs using magnetic tape although a later version probably will. As it stands the only way to load and run programs is to enter them through the keyboard each time one switches on but it is fairly straightforward to make a simple cassette interface which employs the SC/MP serial input and output and under a pound's worth of electronics. The details are explained in the manual.

The basic kit has 256 bytes of RAM, and this is more than enough for any of the programs in the

```
0000
       ↑
0100 ROM
       ↑
0200   ↕
       ↓
0300 ROM
       ↕
0400   ↕
       ↑
0500 ROM
       ↑
0600   ↕
       ↕
0700 ROM          4 K
0800   ↕
                  bytes
     RAM/IO
0900
     DISPLAY
0A00
     RAM/IO
0B00
     EXTRA RAM
0C00
     RAM/IO
0D00
     DISPLAY
0E00
     RAM/IO
0F00
     BASIC RAM
0FFF
```

Fig 1.   Addressing for the MK14

---

increments the appropriate register before or after use). The auto-indexed mode is particularly useful for reading date from internal tables and for implementing stacks to hold the arguments and return points for subroutine calls. There are three user settable flags, available through the edge connector, which can be used as control outputs. SC/MP has an extension register which can be used for the serial input and output of data or as a second accumulator. The edge connector also enables one to use the two inputs. One of these is an interrupt input and can be used to make control pass to a user written program. It is simple to use the four 'rows' of the keyboard described above as four more inputs. Finally there is a delay instruction which is useful for timing. The load from register time is about 3.5 microseconds which is about average in speed. Two disadvantages of the SC/MP chip are the somewhat restricted instruction set and the fact that there are only one or two byte instructions which mean that some operations require several steps. This results in programs which are slightly longer than those for comparable machines. An advantage of the restricted instruction set is that it is quickly memorised.

Extensions

Sockets are provided for three more chips. One is for a RAM/IO device which adds 16 input/output lines and 128 bytes of RAM. The others are a further 256 bytes of RAM. The RAM/IO device is a versatile chip which enables a program to define the sixteen IO lines as inputs or outputs. These sixteen lines can be addressed as two eight bit words or set, reset, and tested as individual bits.

With some extra gating (there are a few unused gates on the board) it would be possible to address the full 4K of memory. Only 1K of addressability is required for a TV type display and this would make a very desirable peripheral in conjunction with an ASCII keyboard. Addressing the full 64K capability of the SC/MP would require rather more extensive modifications.

Applications

It is not of course possible to run BASIC or other such languages on a machine with this amount of RAM. One might just manage it if one extended the memory to 4K and added a TV display and ASCII keyboard. Instead one has to write in machine code or rather write in the assembler which is its mnemonic equivalent and then do the assembly oneself. The hex keyboard makes typing in machine code quite quick and a program written in machine code should run much more efficiently than its BASIC equivalent. My personal interest in computing has led me from writing in high level languages like FORTRAN to assembler and machine code as these represent the link between the program as an abstract structure and the reality of the electronics. I would therefore argue that writing programs in assembler has certain educational advantages.

I have met several people who have bought and developed home computer systems but seemed at a loss for applications and end up just extending them. The MK14 training manual contains several amusing and interesting programs which are well worth trying out.

If one fits the RAM/IO device a wide range of control applications arise. There will be six input channels, four output channels, and sixteen input/output channels. Toys, robots, and laboratory applications would find the small size and power

training manual. Figure 1 shows how the memory is allocated. The memory wraps round on itself after 4K so that the RAM is directly above the PROM. This allows the monitor to use program relative addressing to save the contents of the SC/MP registers when the user program stops execution. It uses the locations (0FF9-0FFF). The monitor program has four copies in the first 2K of memory, (0000-08FF), and the basic RAM occupies the last ¼K, (0F00-0FFF). The addresses for the display have sixteen copies in (0900-09FF) and sixteen more in (0D00-0DFF). If the RAM/IO chip is fitted it has four copies in the ranges (0800-08FF), (0A00-0AFF), (0C00-0CFF), (0E00-0EFF). The first 128 bytes of each copy being concerned with IO and the last 128 bytes being RAM. If the 256 byte extra RAM chip is used it is addressed in the range (0B00-0BFF). The degeneracy of the addressing can be removed by extra gating.

The kit uses the SC/MP2 microprocessor. There are four addressing modes: Immediate, Program Counter relative (which enables one to write short programs which can be located anywhere in memory without modification). Indexed (using the address stored in any of the three 16 bit index registers), and Autoindexed (like indexed but decrements or

consumption an attractive possibility. An interesting domestic application would be the control of heating and lighting since the large number of inputs make it possible to act on the outside lighting, the room temperatures, the number of people in each room, etc. One could make sophisticated fire alarms or security systems which operate various emergency procedures including phone calls. The simple way in which one can extend the keyboard make the kit very suitable for games involving several players. One could make sophisticated alarm clocks capable of controlling many independent devices and of conditional actions. The large number of control channels make the kit suitable for electronic circuit testers and telephone exchanges. One could use it in multiplexed radio control or as a PROM programmer. The list is endless.

Conclusion

As an independent consumer I was very impressed by this kit. Its low cost and versatile but economic design are very pleasing. I am sure that it will have great success.

Supply 12V ac rms at 1 amps ac rms
Display NSA1198   8 digit LED

R1	4K7
R2	2K2
R3	100K
R4	1K
R5	2K2
R6	1K
R7	1K
R8	1K

R9	1K
R10	1K
R11	5K1
C1	27pF 160v Polystyrene
C3	0.1µF 100v
C4	0.1µF 100v
C5	0.1µF 100v
C6	22 F 10v
C2	1500 F 16v
D1	1N4002
D2	'' } not supplied
D3	''
D4	''
IC19	5v regulator
IC1	ISP-8A/600 (SCAMP-11)
IC2	DM745571 512 x 4 ROM
IC3	DM745571 512 x 4 ROM
IC4	MM2111-A 256 x 4 RAM (8111-2)
IC5	MM2111-A '' ''
IC6	MM2111-A '' ''
IC7	MM2111-A '' ''
IC8	ISP-8A/650 128 x 8 RAM I/0
IC9	DM74LS157 Quad 2 to 1 line selector
IC10	DM74LS157 ''
IC11	DM80L95 Hex TriState Buffer
IC12	DM74LS 193 Quad Tri-State Latch
IC13	DM7445 BCD to Decimal Decoder, 0/C outputs
IC14	DM7408 Quad 2 input AND
IC15	DM7408 ''
IC16	DM74LS08 ''
IC17	DM74LS00 Quad 2 input NAND
IC18	DM74LS04 Hex Inverter

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	For V.24, RS232C.
75150N	£1.20	
75154	£2.50	
4N33	£1.95	

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	4.20
(for Motorola exorcisor, D1, D2 kit bus)	
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.

THE COMPLEAT KIT BUILDER

Jeff Lynch

There is nothing quite like the feeling generated on that first power up when xxx Monitor Version 1.6,? appears on your terminal device, resulting from patience, attention to detail and good preparation.

It's not difficult, but beware of your overwhelming urge to "get it working".

Kits vary in difficulty — often in proportion to the quality of documentation supplied. However, indifferent documentation is no bar to a good result — it just takes longer!

1. Tools

"A poor workman blames his tools", and a good one has the correct tools to start with. Good quality tools are an investment and the following being the minimum required: —

    Long Nose Pliers
    A Multimeter
    Wire Strippers
    Magnifying Glass
    Diagonal Cutters
    Conductive Worksurface (earthed!)
    Set of Nut Drivers
    1 Megohm $\frac{1}{4}$ watt resistor & about 6' wire!

    Solder Removal Braid
    Regular Pliers
    Stanley Knife

    Transistor Soldering Heat Sink
    Soldering Iron Stand with Sponges etc.
    PC Board Stand to hold 6" x 10½" Boards.
    15-25 Watt Soldering Iron — Small and
    Medium Bits 60/40 PCB Solder.
    Screwdrivers — Various sizes including

Philips Head.
PCB Cleaning Solvent, (RadioSpares Cat No. 555-134), Cloth, Brush.

2. General Hints and Tips

(a) Read *ALL* the documentation provided. Read it *ALL* again. Find the errata sheets and if there aren't *any* ring your supplier and check. Amend instructions as required by the errata sheets ensuring that the changed instructions still make sense.

(b) Work in a clean, well lit area, keeping all the components together. Check the components against the modified parts list (using the errata sheet) before assembling anything. Try to avoid working in a distracting environment.

(c) Be patient. Work only when you can complete the section you start in one sitting without getting tired. You will need all your concentration to sort out those capacitors etc., so don't start installing them at midnight. Check every move as you make it. If you have *any* doubts, check it again before soldering. Not only will you have to correct a mistake (which takes time) but you will have to *find* that mistake. A 16K memory board has over 2,000 solder joints, and three days after starting you'll be at your wits end trying to establish which one has got the wrong component in it!

(d) Check voltages supplied to IC sockets before inserting ICs are correct, and within the tolerances noted by the manufacturer.

2. (e) *Never* install or remove ICs in sockets with power on the board, nor insert or remove

boards from the chassis/mother board with power on the machine. Basically ensure power is OFF before doing anything to the interior of the machine

(f) *Always* install CMOS or MOS IC/Devices last following the procedures outlined in part 4.

3. P.C. Board Assembly

(a) Examine the bare board for shorts between traces and IC socket pin positions etc., on both the component and solder (front and back!) side of the board. On S100 boards, check for zero resistance (shorts) between —

Pin 1 (+ 8v) & 50 (Ground)
Pin 2 (+ 16v) & 50 (Ground)
Pin 51 (+ 8v) & 50 (Ground)
Pin 52 (− 16v) & 50 (Ground)

& Voltage Regulator Output pins & Pin 50

If a short exists use a magnifying glass to examine the board. Look in areas of high pin/trace density such as IC socket positions, and also search especially carefully under silk screen markings. Take your time examining the board.

(b) **Component Device Identification**

Follow instructions carefully. Use a colour code chart to identify resistors etc., find a friend to help you with the capacitors. I find these the most difficult, but normally they can be identified by a process of elimination by sorting them into piles of different values/shapes and comparing the number in each pile to the parts list. Remember that 100N is 0-1 Mfd, 10N is 0-01 Mfd. With capacitors, take extra care to get the polarity correct where necessary. (The +ve lead is identified by: - i) the lead with a single small dot nearest it, ii) the lead nearest a + marked on the component, or iii) if the component has a single large dot on it, the +ve lead is to the right when the dot is facing you, and the leads are at the bottom).

(c) **Installation**

Follow the manufacturers instructions in the order of assembly. Bend leads on diodes and small capacitors carefully, using the long nosed pliers to hold the lead near the component, thus avoiding undue stress on the component. When it comes to installing IC sockets, it is best to either, i) put a strip of stiff card across a row of sockets, and tape it tightly so the sockets won't lift off the board when it is turned over for soldering, or ii) on each socket blend diagonally opposing pins to hold the socket tight against the board.

Make sure that sockets are seated firmly against the board before soldering. Install a section of components at a time, bending the leads to keep them in place. Tick them off the parts list as you do so on completion of a set (e.g. all the resistors) make sure you have none left.

Do not install switches until after you have cleaned the board (see Part 5.).

(d) **Soldering**

i) Use a low wattage iron (15-25 watts), and 60/40 Rosin core solder (Not acid core).
ii) Keep the bit clean using a damp sponge.
iii) Solder only on the non-component side unless told to do otherwise.
iv) Check before soldering that it is the right component in the right place.
v) Apply the bit to both the component lead AND the board foil, and then apply the

solder to the bit.

vi) It should melt immediately, and flow smoothly into and around the hole/lead. Less than ⅛" solder is normally sufficient.

vii) If the solder does not flow — remove the bit immediately — the junction you are soldering is probably dirty or corroded. Clean it with a little solvent (away from the bit! or flames etc) before retrying.

viii) Remove the bit as soon as the solder has flowed into the hole — leave to cool before going on to the next joint. Disturbing the joint before it cools will cause "cold" solder joints. Do not keep the heat applied for more than a second or so as overheating can i) cause the copper foil to lift and ii) cause "heat shock" to the components you are soldering. Both of these faults will cause long term reliability problems.

ix) The joint should be bright and shiny with a smooth surface. A dull or rough surface is an indication of a "cold" joint. Re-apply heat in these cases.

x) Once all component have been soldered (except any switches) all unfilled holes should be solder filled. Although they are "through plated" this is no guarantee there is any electrical or mechanical contact between one side and the other. Solder filling is the only safe method. (The through plating is there to provide a surface for the solder, according to military specifications!)

xi) Cut off excess leads using diagonal cutters.

(e) **Desoldering**

If you've been extra careful, you'll probably never need to desolder, but you might have to remove a duff component.

i) Never just heat and pull. You will do irreparable damage this way, with the copper foil lifting off, the throughplating pulling out etc.

ii) Apply desoldering braid to the joint, and then apply the bit to the braid. Don't push too hard — as the braid acts through capillary action and needs the gaps between individual wires to work. When the braid has sucked up some solder, remove bit and braid simultaneously. Cut off used braid, clean the bit and repeat until you have removed as much solder as possible.

iii) Be careful not to overheat!

iv) When 95-100% of the solder has been removed, heat the lead and simultaneously use a screwdriver to centre the lead. Gently pull the lead from the hole while applying heat.

Do all this in short stages, allowing the components to cool between attempts.

4. Handling MOS/CMOS Devices

MOS/CMOS devices are sensitive to static voltage discharges, and are readily damaged by static discharged from handling during assembly. However, following the rules below will make this highly unlikely — and most manufacturers insist that these sort of rules are followed to comply with warranty requirements.

(a) Using the 1 Megohm resistor in the "Tools" list and the 6' wire, connect yourself, via the resistor, to a good ground (cold water pipe, radiators etc). The wire can be attached to a

metal watch strap, or just noosed round your wrist. The resistor is a safety precaution to limit the extent of any accidental shocks you may get by touching "live" wires!

(b) Wear cotton clothes — synthetics are bad news!

(c) Spread out a conductive work surface and earth that as well. A sheet of metal is best, but a strip of Aluminium cooking foil is a good alternative. Do not use any plastic or glass. They are not acceptable.

(d) Try to work in a medium humidity atmosphere. Dry and/or cold air assists the building up of static and should be avoided if possible.

(e) If you have to solder a MOS/CMOS device ensure that the soldering iron tip is grounded.

(f) If you use any tools, ground them before touching the device — insulated pliers for instance may have picked up a charge when you last used them.

(g) Basically, try to keep everything in contact with everything else — and all of it grounded.

5. PC Board Cleaning and Inspection

When all the components except switches have been soldered — the Board should be cleaned. This step is vital. Multicore solders tend to spit rosin and minute drops of solder around the board. This has all got to be removed. A good quality solvent should be obtained (RadioSpares No. 555-134 is good as well as being non flammable) but industrial alcohol will do at a pinch. You will also need a stiff $\frac{1}{2}''$ brush.

(a) Liberally pour the solvent onto the solder side of the board. "Scrub" the solvent off. Pour on some more solvent and repeat. Make sure you cover the whole board. When satisfied you've covered the whole board thoroughly, blot the remaining solvent up with a cloth. Finish off off by wiping the contact fingers clean with a solvent soaked cloth.

(b) Now, using a magnifying glass, inspect the entire board for solder splashes, shorts, cold joints, unsoldered pins.

(c) When you're satisfied — carefully solder any switches into position, trying to avoid solder spits etc. Switches and solvent don't go together well — the switches tend to dissolve! That's why you should leave them to the last.

6. Inserting IC's into Sockets

This is not a difficult job, but there are two things to consider. Use your commonsense and check IC orientation before insertion.

(a) Line the pins of the IC's up as accurately as possible before you try to insert the IC.

(b) Make sure you push hard to get the IC to seat correctly. In some cases it will appear to require too much effort. Don't worry. The sockets are meant to grip tight.

(c) Once in, check that no pins have been bent up under the IC. It should be obvious if you look end on at the IC.

7. SUMMARY

If you haven't built a kit before, don't worry overmuch. Just consider the value trebled and think professionally! Be patient. Read these tips again. Read the instructions with the kit until you understand them. It's not as difficult as I may have made it appear, but if you follow these rules, you'll probably end up with a professional result which will work first time and give long useful service.

YES, BUT WHAT DOES THE LITTLE BEAST DO?

W. G. Marshall

Introduction

The last couple of years or so have seen the arrival of a large range of microprocessor 'chips' and their introductory kits. Most people have probably seen the adverts for them by now and wondered just what it is they do and how they do it. In this article I intend to describe the basic functional blocks which make up a microprocessor system, with particular reference to the National SC/MP. In a small set-up consisting of processor chip, read-write memory (RAM) read-only memory (ROM) and a few control logic gates, simple program execution should be easy to follow. Assuming that the reader knows something about the binary number notation and that a decimal number is represented by a string of '1's and '0's, it can be seen that in an electrical system a 'high' voltage ($\approx$4v) is a '1' and a 'low' ($\approx$0v) is a '0'. Theoretical binary can thus be processed in the machine as a series of high and low voltage levels.

Inside the Microprocessor

Fig. (1) illustrates the essential working blocks of the processor chip, greatly simplified, as it is not necessary to understand all the inner workings in order to use these little beasts. The *timing and control logic* determine the movement of data and the operations that are carried out upon it. The processor operates in a series of *cycles,* each of which consists of two parts: instruction fetch followed by instruction execution. This cyclic operation is common to virtually all digital computers including all microprocessors. The working registers in the processor can be thought of as groups of bistable flip-flops, each flip-flop being able to *store* a 1 or 0 logic state. In the case of the SC/MP, the Program Counter (PC) and Pointer registers each consist of a row of 16 flip-flops. The Accumulator, Extension and Status registers each contain eight. The term 'Bit' is given to a single unit of data, and 8
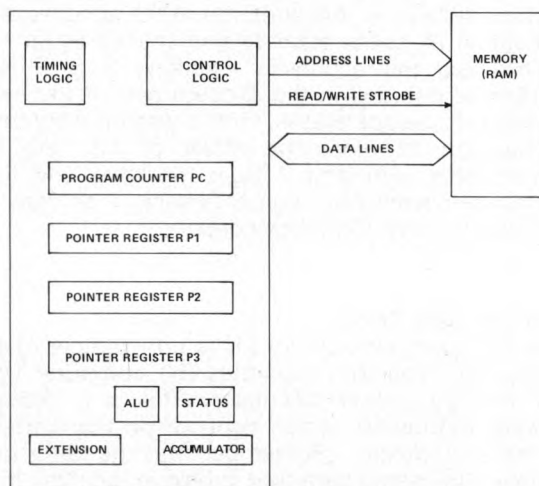
bits are usually called a 'Byte'. Hence an example of a byte is: 01010010, which has the decimal equivalent of +82. Now we can say that the accumulator is 8 bits long and the program counter is 16 bits in length.

Taking the program counter first, its outputs go out on the *address* lines of the microprocessor (MPU) at instruction fetch time, and so its contents determine *which* instruction is read from the memory. The instruction is read into the MPU from the 8 data lines. Having received its instruction, the computer now proceeds to obey it. It may for example, have to read into the accumulator register the contents of some memory location. When execution is complete, the program counter is *incremented by one* and a new computer cycle begins with another instruction fetch.

Most computers have one or more registers called *accumulators.* The SC/MP has two (the extension register is virtually an accumulator), the 6800 has two and the 8080 one. Most arithmetic and logical operations are carried out on data contained in the accumulator, and its tends to be the register that sees the most use.

Next come the *pointer* or *index* registers. These may be loaded by program and their contents may be selected, also by program, to go out on the address lines at instruction execute time. Hence they address the store location where program *data* will be stored or taken from.

The function of the various bits in the *status* register varies from MPU to MPU, but some bit functions are fairly standard. One bit is normally defined as the Carry/Link bit, and is set high when the result of an arithmetic operation causes a carry from the most significant end of the accumulator. This is useful when handling numbers of greater length than 8 bits. For instance if two 16 bit numbers need to be added, the least significant (rightmost) 8 bits of each are added first. The result is stored, and the most significant (leftmost) 8 bits of each added together, the carry/link being added in as well to propagate the carry.

Arithmetic in computers is normally carried out with binary numbers in what is called '2's complement' form, which results in the most significant bit in a byte being the sign bit. That is, if it is set to a 1, then the number is negative, if not set then the number is positive. It isn't quite that simple however! For example:—
00010110 is +22 in decimal

↖
sign bit

To get −22 invert (change 0s to 1s and vice versa) all the bits in the byte and add 1 to the result:—
11101001
+1
———
11101010   is −22 in decimal

Now it can be seen that with an 8 bit accumulator, numbers in the range −128 to +127 may be handled. But what happens when you add two numbers together when each lies close to the same end of the

FIG. 1. SC/MP SYSTEM BLOCK DIAGRAM

TIMING LOGIC — CONTROL LOGIC — ADDRESS LINES — MEMORY (RAM) — READ/WRITE STROBE — DATA LINES — PROGRAM COUNTER PC — POINTER REGISTER P1 — POINTER REGISTER P2 — POINTER REGISTER P3 — ALU — STATUS — EXTENSION — ACCUMULATOR

range? For instance add +120 and +112: —

01111000 = 120
+ 01110000 = 112

11101000 is −24 in decimal?!

The answer is −24 in 2's complement form and illustrates an overflow error. If the sign bit of the result is different from *either* of those in the two data bytes then an overflow error has occurred. A bit in the status register is normally set to indicate this condition, and the program may by written to check this bit after arithmetic operations.

That about covers the internal registers and we move on to the block in Fig. (1) marked ALU or Arithmetic Logic Unit. This is simply a piece of logic that performs the addition, complementing, etc. functions on two data bytes, usually leaving the result in the accumulator. The function it carries out is determined by the current instruction being obeyed.

The Computer Memory

The memory of a computer has two tasks, one to hold the program or list of instructions to be performed by the MPU, and the other to supply or receive data as required for the program execution. In an 8 bit system, each location in the memory addressed by the MPU contains 8 bits of data. Sometimes 8 bits are not enough to hold a complete instruction and so two or even three consecutive locations are used to hold one machine instruction. The SC/MP uses one and two byte instructions and the handling of these will be shown by detailed examination of a simple program. As well as the address and data lines going to the memory from the MPU, there is a *control signal* which determines whether data currently on the data lines from the MPU is *stored in* the addressed location, or whether the RAM is required to *supply* data. This is usually called the Read/Write Strobe. A read-only memory (ROM) does not of course require this signal as it is only possible to *read* data or program from such a device. At this point I should mention that the groups of data and address lines are generally called 'Buses'. Hence a **6800** MPU is said to use an 8 bit data bus and 16 bit address bus. If you examine the chip you will find 8 pins allocated for the data wires, and 16 for the address wires.

Examination of a Simple Program

One of the best ways of getting to know how a computer works is to play with one! However if that is not possible yet, consider the following small SC/MP program which adds one number to another several times and then halts.

Memory Address	Mnemonic Instruction Code or data		1st Byte Binary Code	2nd Byte or data
1, 2	LD	0(P1)	11000001	00000000
3	CCL		00000010	
4, 5	ADD	1(P1)	11110001	00000001
6, 7	ST	1(P1)	11001001	00000001
8, 9	DLD	2(P1)	10111001	00000010
10, 11	JNZ	−11(PC)	10011100	11110101*
12	HLT		00000000	
13	+2		00000010	
14	+3 } **		00000011	
15	+4		00000100	

The above is a small section of program. (Lightly tick off the rows in pencil as you read).

*Leftmost bit is 1, so this is a negative number stored in 2's complement form, = −11 in decimal.
**Initial values held in locations 13, 14, 15.

The program assumes that the pointer register P1 is set to 13 (i.e. data to be manipulated is stored from location 13 onward) and that the accumulator contains zero on entry into the program. The column on the left of the listing gives the address in decimal, of each instruction or piece of data. The middle column contains a mnemonic representation of the instruction or data. This is easier for the programmer to follow than the binary codes, in the final column, which the MPU understands. This binary code is the form in which instructions and data are stored in the *memory.* You can see that most instructions are of the two byte variety, requiring two consecutive memory locations to hold them.

On starting the program, the program counter register in the MPU is at 1 and so during the instruction fetch part of the cycle the processor reads the instruction contained in *location* 1 *of the memory.* This is achieved because 1 goes out on the MPU address bus and the memory puts the *contents* of location 1 onto the data bus. This instruction is latched *into* the MPU and it decodes the binary code of the instruction to find out what it has to do next. The most significant bit of the byte is set to 1, so the MPU knows that it only has *half* the instruction. The program counter is incremented to 2 and the byte in that location (2) read in. The MPU *now* proceeds to execute the double-byte instruction which says: "Load into the accumulator the data which is contained in the memory location addressed by pointer register P1 (ie 13), offset by 0."

Quite a mouthful! It should be easy to follow, except for the: "offset by 0". This is an additional addressing feature available on most MPUs whereby the address that goes out on the address bus at execute time is the arithmetic sum of the pointer register contents *and* the second byte of the instruction. In this instance the second byte or offset is *zero,* so the address 13 (in P1) goes out on the address bus. The data contained *in* location 13 *comes back* on the *data bus* and ends up in the accumulator.

The first computer cycle is now complete, the program counter is incremented to 3 and the next (single byte this time) instruction is taken from store. Bit 8 of the instruction is zero this time, so the MPU knows that it has the *complete* code and execution follows. This time all that is done is to set the carry/link bit in the status register to 0.

The next instruction (in 4,5) adds the contents of location 14 to the accumulator. The data is taken from location 14 this time because we *do have* an offset of +1, this time against P1 which is set to 13.

The instruction in locations 6 & 7 stores the result of this addition, ie. the contents of the accumulator, in location 14, as the offset relative to P1 is again +1.

The next instruction in locations 8 & 9 is a composite one and in the execute part of the cycle performs three operations. First it reads the contents of location 15 (note the offset of +2) into the accumulator, subtracts 1 from the latter and then stores the accumulator contents back in location 15. So now we have +3 in that location.

The Decision Time

The next operation (in 10,11) is an example of the instruction type that separates the computer from the average pocket calculator. This is a *decision making* instruction, which depends on the contents of the accumulator. Remember that the latter now contains the same data that is held in location 15. If the accumulator is zero then the program continues with the instruction in location 12. If it is not zero,

then the PC (which contains 11) is reset *back* by the amount (= −11) indicated by the offset. That is, to 0. The PC then increments to 1 and the LD 0(P1) code is fetched and executed again. The program thus contains a 'loop' with a 'loop counter' in location 15, the latter decremented by 1 every time the computer cycles round the loop. Eventually location 15 contains 0 and so does the accumulator — and the decision now is that the computer executes the instruction in location 12 instead of returning to 1. This is a Halt command and the computer stops any further operation.

Let us now examine the contents of the three data locations and see what they contain. Location 13 should still contain +2 as no operation other than reading (which doesn't alter the contents of memory) has been performed upon it. Location 14 contains the 'answer' or final result of the calculation. Using the initial data values given, this should be +11. Work through the program by 'hand' to check that this is so, and that the sequence of operations performed is: 2+3+2+2+2. Location 15 will be 0 as it is decremented for each trip round the loop of program steps.

Conclusion

In a short article such as this, the immense versatility of the computer or MPU cannot be fully explored. However I hope the reader's appetite has been whetted and that he will wish to investigate further the computer's power of decision making, by assembling one of the many MPU kits on the market. Even the simple ones may form the basis of more ambitious projects in the future.

BASIC

THE FIRST STEPS

Philip Couzens

Step after step the ladder is ascended
George Herbert, *Outlandish Proverbs* (1640)

BASIC, Beginners' All-purpose Symbolic Instruction Code, was originally designed at Dartmouth College, U.S.A. to be a simple introductory language from which students would progress to the more powerful high level languages such as FORTRAN, COBOL, ALGOL and PL/I. The grammatical rules of BASIC are so few and simple that the user can quickly program the computer to solve his problems.

Although BASIC is so simple, it is still a very useful and powerful high level language suited to both mathematical and business situations alike. Due to its simplicity, interpreters and compilers for the language are small, making BASIC particularly suitable for microprocessor based systems in which core storage may be limited.

If one has experience of one of the popular high level languages, particularly FORTRAN, then converting to BASIC should take only a few hours. The remainder of this article is intended for those with no previous programming experience, but might also be useful for those wishing to change from another language.

BEFORE PROGRAMMING

Before programming can begin, the system has to be switched on and the BASIC interpreter has to be loaded into memory. The method of doing this varies considerably from system to system, obviously instructions for doing this cannot be given here. It will be assumed that the system is ready and working in BASIC.

At least two of the control keys should then be identified. These are the 'carriage return' key which is pressed every time something is entered at the keyboard, and the 'escape' key which is used to stop the computer doing a program. If no particular key is marked 'escape', then the instruction is probably given by holding down the 'control' key while pressing another key — check in your manual for this.

One further key that is useful is the 'rubout' or 'delete' key (sometimes back-arrow or underline). This has the effect of erasing the last character which was typed. Pressing it twice will erase the last two characters, and so on.

WRITING A PROGRAM

Any line which is now typed in at the keyboard will be treated as a command, and this will be executed when the 'carriage return' key is pressed. If a line begins with a number (any integer between 1 and 10000), then that line will be stored as part of a program.

So, for example, the first line of a program is typed in:

    10 PRINT "HELLO"

This, of course, is followed by a carriage return. *Every program in BASIC should be terminated by the instruction END, so the last line is typed in:

    20 END

There now exists a complete program which will be typed out at the terminal if the command LIST is issued at the keyboard; the lines of the program are typed out in order of their line numbers, not in the order in which they were typed originally. To execute the program the command RUN is given. This program simply prints HELLO and then stops.

If it becomes necessary to insert further lines into the program, then this is done by choosing a suitable line number which has not already been used. For example, a line is inserted between the two existing lines:

    15 PRINT "HELLO AGAIN"

If the command LIST is now given, the whole program is displayed at the terminal:

    10 PRINT "HELLO"
    15 PRINT "HELLO AGAIN"
    20 END

*This is not necessary on all systems, but it is useful to conform to standards.

If it becomes necessary to change a line, then this is done by creating a new line with the same number as the one to be replaced and retyping the whole line. A line is deleted by typing in the line number followed by 'carriage return'.

So it is seen how a program is built up. It is usual to choose 10 for the first line number and to go up in tens, that is 10, 20, 30 . . . This leaves plenty of unused numbers so that lines can be inserted if required. The command RENUMBER, which is supported by most systems, changes the line numbers to fit this pattern.

When it is wished to lose the existing program and

start a new one, the instruction *NEW is given. The old program is lost completely and a new one may be commenced.

The important BASIC commands which we have met so far are:

(i) RUN — the current BASIC program is executed
(ii) LIST — the current BASIC program is displayed at the terminal
(iii) NEW — the current BASIC program is deleted so that a new one may be written
(iv) RENUMBER — the lines are renumbered to fit the pattern 10, 20, 30 . . .

ARITHMETIC IN BASIC

Numbers are expressed in BASIC in three ways:

(a) Integers — written without a decimal point, e.g. 4395

(b) Decimals — written with a decimal point, e.g. 31.416

(c) Numbers written in standard form. $1 \cdot 08 \times 10^{-4} = 0 \cdot 000108$ is written as 1.Ø8E−4

A negative number is preceded by a minus sign in the usual way, for example −4395, −31.416, −1.Ø8E−4.

These notations can be used as and when convenient.

*On some systems the command is SCRATCH

Numbers will be typed out by BASIC in the most suitable form, the user is normally not able to choose. Unless a number is particularly large or small the integer or decimal form will be chosen.

All arithmetic expressions have to be typed in at the keyboard, and therefore the notation is limited by the facilities of the keyboard. Addition and subtraction are denoted by the usual signs + and −. Multiplication is denoted by the sign * and division by the symbol /. The up-arrow ↟ is used for exponentiation (raising a number to a power), and brackets are used as in ordinary arithmetic.

As an example, the expression

$$\frac{(3 \cdot 14 + 2 \times \frac{1}{2})^4}{9^2 - 4 \cdot 6 \times 10^6}$$ could be evaluated

in BASIC by running the program:

1Ø PRINT (3.14 + 2*1/2) ↟4/ (9 ↟ 2 − 4.6E6)
2Ø END

BASIC uses standard algebraic hierarchy when performing arithmetic, the order of operations being given by BEDMAS (brackets, exponentiation, division = multiplication, addition = subtraction). Operations of equal priority are performed from left to right. So the answer to 1 + 2 * 3 is given as 7 since the multiplication is done before the addition. Notice that 12/3*4 is given as 16. The order of performing operations is in fact exactly the same as one would use in ordinary arithmetic.

LIBRARY FUNCTIONS

All BASIC systems (except those called Tiny Basic etc) should support at least the following set of library functions:

Function	Meaning
SIN (x)	the sine of x, where x is an angle in radians
COS (x)	the cosine of x, where x is an angle in radians
TAN (x)	the tangent of x, where x is an angle in radians
ATN (x)	the principal angle (measured in radians) whose tangent is x
SQR (x)	the square root of x
LOG (x)	the natural logarithm of x
EXP (x)	the value of e^x
ABS (x)	the absolute value of x, ABS (−1.9) = 1.9, ABS (1.9) = 1.9
INT (x)	the largest integer not greater than x, INT (6.3) = 6 INT (−6.3) = −7
SGN (x)	the sign of x, has value 1 if x is positive, 0 if x is 0 and −1 if x is negative x is positive, 0 if x is 0 and −1 if x is negative
RND (x)	a random number between 0 and x

In the above, x, called the argument of the function, represents an expression of any complexity, and may include other functions. Note that all angles are measured in radians so that degrees have to be converted (multiply by $\pi/180 = 0 \cdot 0174533$).

It should now be possible to use BASIC as a scientific calculator.

EXAMPLES.

(1) Evaluate $\sin 30° + \sqrt{2}$
    1Ø PRINT SIN (3Ø * Ø.Ø1745) + SQR (2)
    2Ø END

(2) Evaluate $\frac{2}{3}(\ln 3 + e^{\sqrt{2}})^3$ ↟
    1Ø PRINT 2/3 * (LOG (3) + EXP (SQR(2)))   3
    2Ø END

Note that several levels of brackets are allowed, but one should be careful that they are closed correctly.

VARIABLES IN BASIC

Variables are used in BASIC in a way not dissimilar to that of ordinary algebra. The equals sign, however, is used in a very different way and this is stressed by the use of the word *LET. For example, consider the BASIC statement

1Ø LET A = 4.2

This assigns a value of $4 \cdot 2$ to the variable A. The meaning is made clear by reading the instruction as 'let A become $4 \cdot 2$'. The expression on the right hand side of the 'becomes' sign can be of any complexity, and can contain any number of variables (including the variable on the left hand side). The left hand side must contain only one variable.

Consider the statement 3Ø LET A = A + 1. Although this would be nonsense in mathematics, it is allowed in BASIC, meaning 'let A become what it was plus one' and has the effect of increasing A by one. It is, indeed, a very useful and much used facility.

*In many versions of BASIC the word LET may be left out.

Ordinary variables which represent numbers in BASIC can be given any of the 286 names which

consist of a single letter: A, B, C, . . . Z, or any letter together with a single digit:

    A∅, A1,. . . A9
    B∅, B1,. . . B9
    :   :
    Z∅, Z1, . . . Z9

So to evaluate the expression given earlier,

$$\frac{(3\cdot14 + 2 \times \frac{1}{2})^4}{9^2 - 4\cdot6 \times 10^6},$$

the program given here would also be suitable:

    1∅ LET A = 3·14 + 2 * 1/2
    2∅ LET B = 9*9 − 4.6E6
    3∅ LET C = A ↑ 4/B
    4∅ PRINT C
    5∅ END

THE PRINT AND INPUT STATEMENTS

The PRINT statement has already been met, but is now considered in greater detail. One or more items can be printed by a single PRINT statement. The line 1∅∅ PRINT 2/3, X, 2.3 will type out the value of $\frac{2}{3}$, that is 0.666667, followed by the numerical value of X followed by 2·3. Since the items in the PRINT statement are separated by commas, they will be typed out in columns across the page. Alternatively, if they are separated by semi-colons then they will be squeezed together. Anything which appears between quotation marks in the PRINT statement will be printed out exactly as it appears in the program. Plain numbers and arithmetic expressions are allowed in the PRINT statement. A typical line in a program would be:

    5∅ PRINT "THE ANSWERS ARE: ", X; Y, 39/3; Z

This will cause the following to be typed when X = 0·333333, Y = 100, Z = 6

    THE ANSWERS ARE: 0.333333 100 13 6

The effect of 6∅ PRINT is to advance one line of paper. If a PRINT statement ends with a semi-colon then the printing head will remain in its last position until another PRINT statement is encountered.

The INPUT statement is a method of assigning values to variables from the keyboard *as the program is running.* When an INPUT statement is reached during the execution of a program, the program will stop and continue only *after* a number has been typed in at the keyboard. The variable mentioned in the INPUT statement will then have the value which was typed in. The input of a number is prompted by a question mark being typed.

So then, if it is wished to read in a value for X during the execution of a program, this is done by the statement:

    1∅ INPUT X

More than one variable may be assigned a value by a single INPUT statement:

    2∅ INPUT U, V

When this line is reached in a program, then it will stop and continue only after two numbers have been entered. U will have the value of the first one and V the value of the second one.

On many systems it is possible to change the prompt offered by the INPUT command by placing the required replacement in quotation marks in the statement thus:

    1∅ INPUT "TYPE IN THE VALUES OF X AND Y: ", X, Y

Note that the comma is required after the last quotation mark.

Example: This program will read in two numbers and type out their average.

    1∅ INPUT A, B
    2∅ PRINT (A+B)/2
    3∅ END

To execute the program, the command RUN is given. At line 10 values for A and B are required. The request for these values is indicated by a prompting '?'. The values must now be entered with a comma between them, followed by 'carriage return'.

    RUN

    ?4,6
    5

    END AT ∅∅3∅

Now to change the program some of the lines are retyped:

    1∅ INPUT "TYPE IN THE TWO NUMBERS: ", A, B
    2∅ PRINT "AVERAGE IS: "; (A+B)/2

If the command LIST is given again, a list of the amended program is obtained. A run of the new program will produce this:

    RUN

    TYPE IN THE TWO NUMBERS: −4.2,3.9
    AVERAGE IS: −∅.15

    END AT ∅∅3∅

THE GOTO STATEMENT

A program is normally executed in order of the line numbers. The GOTO instruction, however, causes this order to be changed. For example

    5∅ GOTO 1∅

This statement will cause the program to jump to line 10 rather than carry on to the next line after 50. Any existing line number can be used in the GOTO statement.

Example: This program will type out the natural numbers, together with their squares and square roots.

    1∅ PRINT "NUMBER", "SQUARE", "SQUARE ROOT"
    2∅ PRINT "_____"
    3∅ LET N = 1
    4∅ PRINT N, N*N, SQR (N)
    5∅ LET N = N + 1
    6∅ GOTO 4∅
    7∅ END

Work through this program and you will see that it contains an infinite loop (it will keep running for ever). The only way of stopping this program is to press the 'escape' key during execution. Naturally,

this is considered bad programming technique, but it has served a useful demonstration purpose here.

CONDITIONAL BRANCHING

To GOTO statement provided us with unconditional branching, but it is possible to ask a question before branching, and branch only if the answer to that question is yes. It would be useful in the example above to ask at line 45 the question 'Is N equal to 10?' — this would avoid the infinite loop. If the answer to this question is yes, then the program will stop. Otherwise the program continues to run as though the question had not been asked. The question is asked in BASIC by the statement.

45 IF N = 10 THEN STOP

The flow-chart for the program now looks like this:

```
          ┌─────────────┐
          │    START    │
          └──────┬──────┘
                 │
          ┌──────┴──────┐
          │    PRINT    │
          │   HEADING   │
          └──────┬──────┘
                 │
          ┌──────┴──────┐
          │  LET N = 10 │
          └──────┬──────┘
                 │
   ┌──────────────────────────┐
   │  PRINT N, N*N, SQR(N)     │
   └──────────┬───────────────┘
              │
            ◇ IS          YES
            N = 10  ──────────►  STOP
              ?
              │ NO
   ┌──────────┴───────┐
   │   LET N = N + 1   │
   └──────────────────┘
```
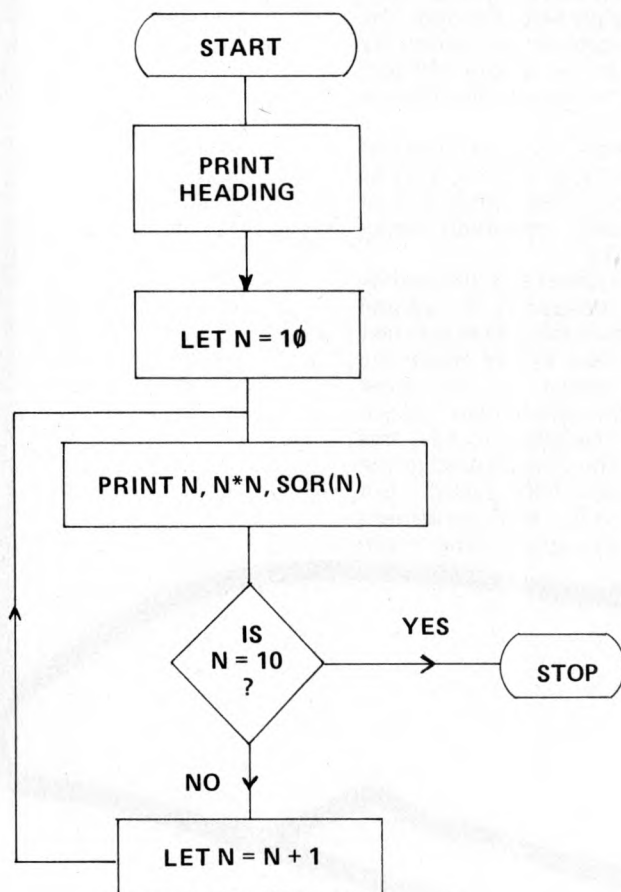
The program is:
```
10  PRINT "NUMBER", "SQUARE",
    "SQUARE ROOT"
20 PRINT " _____ "
30 LET N = 1
40 PRINT N, N*N, SQR(N)
45 IF N = 10 THEN STOP
50 LET N = N + 1
60 GOTO 40
70 END
```
In general, a question may be asked containing any of the relational operators:

=	equal to
>	greater than
<	less than
> = or = >	greater than or equal to
< = or = <	less than or equal to
>< or <>	not equal to

A test of this sort can result in any of the BASIC statements being executed,

e.g. 20 IF A> = B/C THEN PRINT A

This will result in A being printed if and only if A is greater than or equal to (B divided by C).

110 IF A*SIN(T)<=10 THEN GOTO 210

If the result of this test is true, then the program will jump to line 210, otherwise execution will continue as normal.

THE REM STATEMENT

When writing a long and complicated program, it is often useful to have explanations along the way actually written as part of the program. This is done in BASIC by the REM statement. If a line begins with the letter REM (after the line number), then that line will be ignored when the program is executed.

e.g. 10 REM THIS IS A REMARK WHICH
        WILL NOT INTERFERE WITH THE
        PROGRAM
     20 PRINT "THE SAME AS USUAL"
     30 END

FROM HERE?

Although only a few of the facilities available in BASIC have been described here, the reader should now be able to write programs to solve even quite complicated problems.

BASIC was designed for the user in the hope that the computer should be used as a tool to find solutions to what might otherwise be insoluble problems or at least problems which would involve large amounts of time when tackled by hand. But the user often becomes interested in computer programming for its own sake, finding it intellectually satisfying to have written a concise and effective program. By writing simple programs in BASIC one can begin to appreciate the pleasure which can be obtained from computer programming.

"I see Jacob's trying to program his way to heaven."

Typing Without Tears

Paul M. Jessop

To many people, typing is an integral part of their job, but to others the typewriter is merely a means of producing reports, manuscripts and so forth without the need for the recipient to decypher the handwriting of the author. To the new breed of computer hobbyists, typing is becoming more and more important as more time is spent at the teletype/VDU. Until machine readable media are cheaply and widely available, the only way of entering a program will be to type it manually, line by line. To type with one finger may be easy to learn and one soon becomes deceptively proficient so that one does not realise the advantages to be gained by 'touch-typing'. A keyboard can be a very efficient input device and it is sacrilege to negate this inherent efficiency by typing one fingered.

The routines presented here are not to be regarded as a substitute for proper teaching but for anyone who has to teach themself, they offer rather greater chances of success than typing repeatedly easily learned exercises from textbooks.

At this point, the text was to have read 'the author cannot touch-type and has despaired at several attempts to do so.' However, since this article was first drafted, many moons ago, I have in fact made the effort to learn and I can attest to the great improvement it has made to the satisfaction I derive from the writing of an article. Needless to say, this learning has encroached upon the time allotted to the commissioning of my home-brew 6800 system, but when I eventually get around to it, I shall implement these programs to improve my speed and, more importantly, my accuracy.

Figure 1. THE BASIC TYPING PRACTICE ROUTINE

There are two versions of this program, the second of which is a derivative of the first, and also rather more useful. However it is instructive to examine first the simple version and then to see how it can be extended. The practice in this case takes place from all the letters of the alphabet, which may or may not be a good idea for someone starting to learn to type, depending how well they respond to a challenge and whether they merely throw up their hands in despair. A flowchart for this program is shown in figure 1. Where letters are shown in quotes, this means that the ASCII value for that letter is being referred to. The program variables whch are very important are 'CURRENT' and 'NEXT'. These, as their names imply, are the variables which are used to store the characters which are currently under consideration and the character following it respectively. Why it is necessary to look ahead will become clear in due course. The computer outputs the current letter, initially an 'A' and then goes into an input loop until a key on the keyboard is pressed. On each pass through the loop, the variable 'NEXT' is incremented and the tested whether it has become greater than ASCII 'Z'. If so, it is reset to 'A' and the input loop continues as before. Since the time at which the user interrupts this loop by pressing a key is essentially random, what has been done is to create a random variable, 'NEXT'. As was mentioned before, 'NEXT' is the variable containing the character for the next to be considered so the result is that this character has been randomised. This is of course very useful since most so-called random number generators are essentially cyclic in nature and could present the user with easily recognised patterns. This completes all the hard work and all that has to be done now is to copy 'NEXT' into 'CURRENT' and to jump back to the beginning of the program.

This program has obvious appeal in its simplicity but it gives no indication of whether the user has typed the correct character or not and it will only work with the whole alphabet, or by changing the limits (in this case 'A' & 'Z') in the input loop to linear chunks of the alphabet. A revised version of the program is shown in figure 2. This embodies the two improvements implied above, namely a count of the number of correct and incorrect characters typed (positive reinforcement) and it allows the user to choose the specific letters of the alphabet which he desires to practice. The basic principle is exactly the same, that is that the length of time the user takes to respond to the prompt is used to select the next character used. However, instead of 'NEXT' containing an ASCII character, it is used as a pointer to a table which contains the wanted characters. Also, when the key on the keyboard is pressed, this is examined to determine whether it is correct or not and either 'PASS' or 'FAIL' is incremented accordingly. In the previous version the only way the user could escape from the program was by some physical reset button on the processor but now the program looks for a carriage return character and when it detects one, jumps out of the main program and prints the number of passes and fails.

Figure 2. THE MORE ADVANCED ROUTINE WHICH HAS ADDED FACILITIES.

(*: INSERT HERE THE No. OF CHARACTERS IN THE TABLE).

To pass now to the rather more esoteric subject of actually teaching oneself to type, the overridingly important factor is to achieve a rhythm which will become much more important later as faster speeds are achieved. In days of yore the recommended method used to be to type in time with a metronome but these items are rather rare and very expensive today, especially when a 555 timer IC does the same job at a fraction of the price. If one were to be really keen, the rhythm timing function could be written into the program itself but in this case, the hardware-software tradeoff is tilted firmly in the direction of hardware.

Figure 3 represents a typewriter keyboard and the keys outlined in bold lines are known as the home keys. These are the keys where the four fingers of each hand come to rest when the hand in question is not actually typing. The row of keys vertically (or obliquely) above and below each of the home keys is serviced by the finger associated with that home key, with the addition that the rows of keys between the two forefingers are also serviced by the forefingers. The space bar is of course operated by the thumb and when a shifted character is required, the little finger of the hand not in use operates the shift button.

This concludes the mechanics of the problem and all that remains now is a good deal of hard work on the part of the learner. However, one or two other applications of this type of program might be usefully dealt with at this point. A fairly obvious candidate for this treatment is morse code. Both sending and receiving can be taught, in the first case, the computer types (or displays a character) and the student has to echo it on a morse key attached to an input port, and in the second case the computer generates a morse code character and sends this to a loudspeaker and the student has to echo this on a keyboard, possibly improving his typing at the same time. In each case, the length of time taken to respond to the stimulus can be utilised to decide the following character. The same might also be applied to the teaching of semaphore (if anyone uses it any more), but the way in which this might be engineered is left to the imagination of the reader.
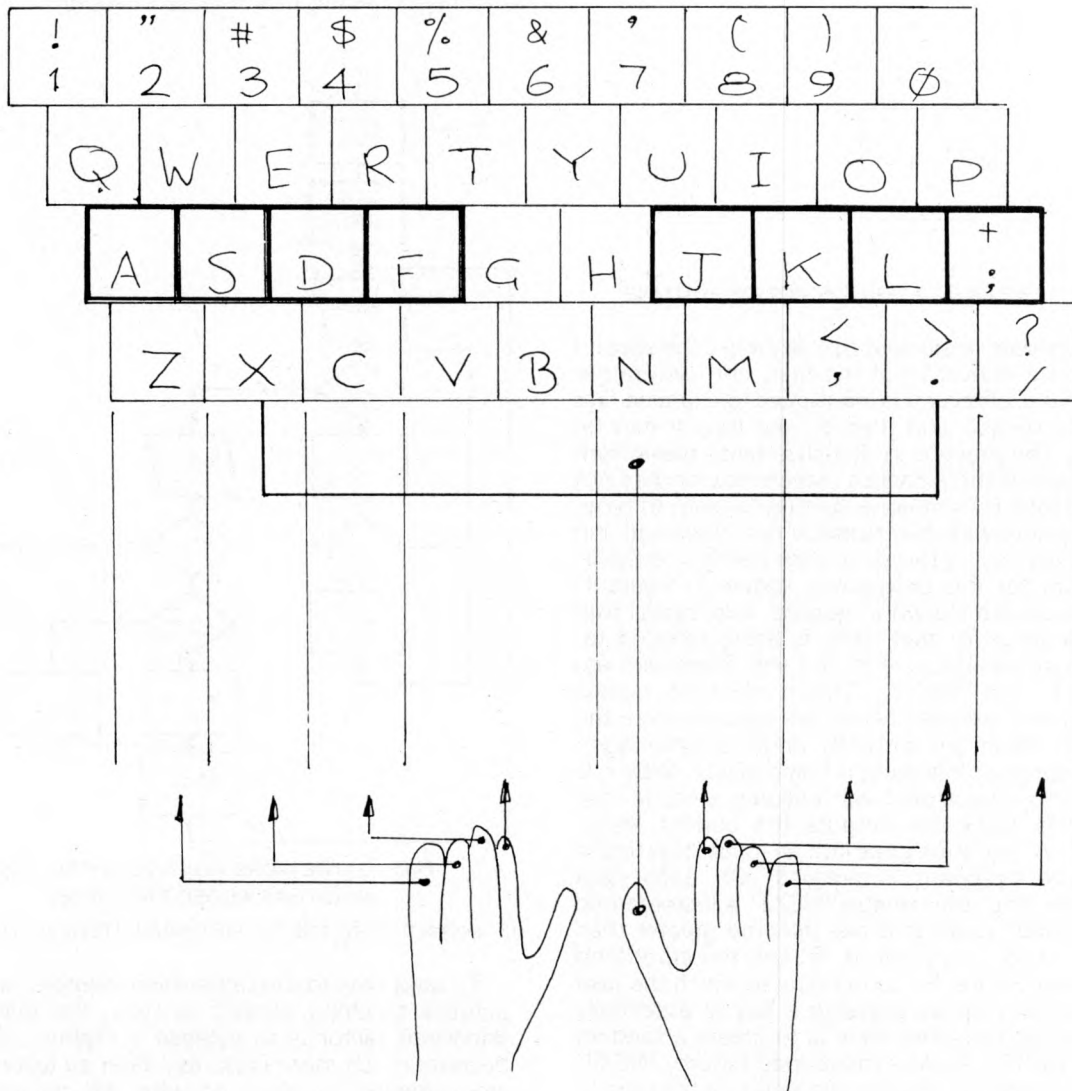
Figure 3. THE MAIN PORTION OF THE TYPEWRITER
KEYBOARD AND THE FINGERS USED ON THE
VARIOUS KEYS'

Figure 1. THE BASIC TYPING PRACTICE ROUTINE

There are two versions of this program, the second of which is a derivative of the first, and also rather more useful. However it is instructive to examine first the simple version and then to see how it can be extended. The practice in this case takes place from all the letters of the alphabet, which may or may not be a good idea for someone starting to learn to type, depending how well they respond to a challenge and whether they merely throw up their hands in despair. A flowchart for this program is shown in figure 1. Where letters are shown in quotes, this means that the ASCII value for that letter is being referred to. The program variables whch are very important are 'CURRENT' and 'NEXT'. These, as their names imply, are the variables which are used to store the characters which are currently under consideration and the character following it respectively. Why it is necessary to look ahead will become clear in due course. The computer outputs the current letter, initially an 'A' and then goes into an input loop until a key on the keyboard is pressed. On each pass through the loop, the variable 'NEXT' is incremented and the tested whether it has become greater than ASCII 'Z'. If so, it is reset to 'A' and the input loop continues as before. Since the time at which the user interrupts this loop by pressing a key is essentially random, what has been done is to create a rand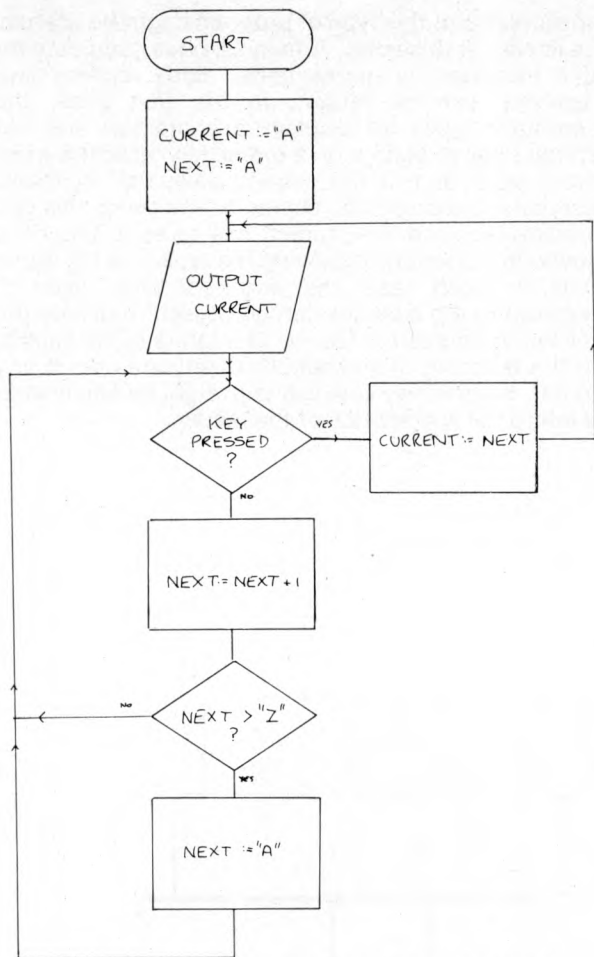om variable, 'NEXT'. As was mentioned before, 'NEXT' is the variable containing the character for the next to be considered so the result is that this character has been randomised. This is of course very useful since most so-called random number generators are essentially cyclic in nature and could present the user with easily recognised patterns. This completes all the hard work and all that has to be done now is to copy 'NEXT' into 'CURRENT' and to jump back to the beginning of the program.

This program has obvious appeal in its simplicity but it gives no indication of whether the user has typed the correct character or not and it will only work with the whole alphabet, or by changing the limits (in this case 'A' & 'Z') in the input loop to linear chunks of the alphabet. A revised version of the program is shown in figure 2. This embodies the two improvements implied above, namely a count of the number of correct and incorrect characters typed (positive reinforcement) and it allows the user to choose the specific letters of the alphabet which he desires to practice. The basic principle is exactly the same, that is that the length of time the user takes to respond to the prompt is used to select the next character used. However, instead of 'NEXT' containing an ASCII character, it is used as a pointer to a table which contains the wanted characters. Also, when the key on the keyboard is pressed, this is examined to determine whether it is correct or not and either 'PASS' or 'FAIL' is incremented accordingly. In the previous version the only way the user could escape from the program was by some physical reset button on the processor but now the program looks for a carriage return character and when it detects one, jumps out of the main program and prints the number of passes and fails.

Figure 2. THE MORE ADVANCED ROUTINE
WHICH HAS ADDED FACILITIES.

(*: INSERT HERE THE No. OF CHARACTERS IN THE TABLE).

To pass now to the rather more esoteric subject of actually teaching oneself to type, the overridingly important factor is to achieve a rhythm which will become much more important later as faster speeds are achieved. In days of yore the recommended method used to be to type in time with a metronome but these items are rather rare and very expensive today, especially when a 555 timer IC does the same job at a fraction of the price. If one were to be really keen, the rhythm timing function could be written into the program itself but in this case, the hardware-software tradeoff is tilted firmly in the direction of hardware.

The Mystery of the Lost Computer

Harry Harrison

I am beginning to feel that I once lived in another world, one of those parallel worlds so familiar to readers of science fiction. Perhaps because I write SF I am beginning to believe in one of my own unreal universes. Why even my good friend, the good Doctor Asimov, who knows everything about everything, doesn't seem to know about my world. In his book *Science Past — Science Future* he discusses the computer revolution. He jumps in one mighty narrative bound from Aiken's Mark I, 1937-1944, to ENIAC in 1946. With nothing in between.

Nothing? With quivering hands I picked up the glorious first issue of *Personal Computer World* and turned to "Past Procession" with its compact boxed history labelled COMPUTER CLOCK. The truth revealed at last. But unhappily for me the truth outlined there is the same one that Isaac Asimov revealed. Yet I put it to you that both of these learned gentlemen are wrong. There were computers in general use between those dates; hard-working, practical computers that did their job remarkably well.

I should know, since I was a computer mechanic during that period and serviced these machines.

What both Asimov and Kewney have ignored for the moment is the fact that there are more ways of running a computer than by electronics and digital means. What about Mr Babbage? With that clue in mind I challenge all you computer fanatics out there to figure out what kind of computer I am talking about. Cover the next lines with your hands and think — no cheating! Have you thought? Good, the answer is. . .

. . . a mechanical analogue computer of course. I have before me, stained and dog-eared, a United States War Department Hand Book on the operation and maintenance of *The Sperry K-3 Automatic Computing Sight.* Something that computes is a computer, right? And when I worked on these things my technical MO was Power Operated Turret and Computing Gunsight Specialist, Skilled.

Since I once saw one of these gunsights on sale after the war in a surplus shop, I hope I am breaking no laws by quoting from the hand book. (Dated Sept. 1943, I must add.)

"The sight computes the range and automatically compensates for ballistics and prediction. . .data input is transmitted to the computing mechanism. . .After the complete solution, the line of sight is offset the proper amounts from the line of

The enemy targeted neatly.

The proof that cleanliness is a major virtue, even in computing.

If the computer is excessively dirty it may be washed with alcohol, benzine, white gasoline, or carbon tetrachloride.
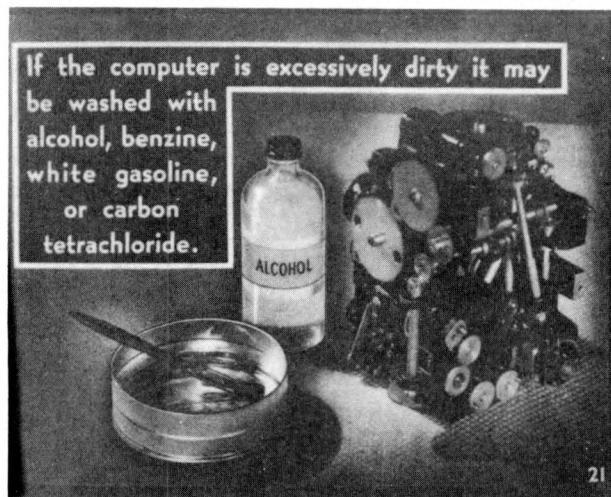
bore, in lateral and vertical planes, to compensate in all deflections."

That last bit, translated from the military technicalese, means the sight points one way, the guns the other. Like leading a bird with a shotgun. When the trigger is pulled the bullet rushes off to the spot where the plane is going to be, not where it is at the moment of firing as with a stationary target. (And let us not be too sure of that stationary bit, either. The Paris gun, with which the Germans shelled the city in World War I, fired a shell that travelled a high arc 92 miles long and took 3 minutes to reach its target. In order to hit the city the rotation of the Earth had to be considered in the computations.)

Heath-Robinson Collection

The only constant the K-3 worked with was the muzzle velocity of the guns and the resultant ballistic orbit. All of the other variables were either fed in by the navigator, the gunner or the instruments; altitude (therefore air pressure), wind velocity, distance to target, relative speed of target, and more. These variables were inserted by the strangest Heath-Robinson collection of gadgetry ever conceived by man. Gear trains, take-up springs, revolving plates with offset ball cages and — pride of the pack — three dimensional contour cams with cam followers. These were beautiful things, machined out of solid brass, and could be rotated about a central axis, or moved back and forth along that axis. They were the most educational device in the world because you could *see* the analogue actually at work.

But our computer problems weren't the ones you have today. No glitches or static electricity problems to bother us. Backlash in the geartrains was what we had to worry about! Not that we didn't have electricity in the K-3 — it was not powered by steam and leather belts as has been hinted — why, we had a constant speed 2000 RPM motor. And a light bulb for the reticle sight.

The computer worked. These same principles were applied in that other analogue computer, the Norden Bomb Sight. (Someday I must tell you about the *complicated* device it used to release the bombs.) The B-17's and B-24's that carried power turrets with computing gunsights flew in stepped formation so any attacking pursuit plane would have to face the fire from a number of guns. The German fighters, for all their strengths, were very unstable gun platforms. They had open sights and their wing guns were zeroed at a point 200 yards in front of the plane; that is, the cone of fire from the guns met at this point. The maximum range where they might hit what they aimed at.

The K-3 was accurate from 1000 yards on in.

So the aerial gunner had theoretically 800 yards to fire without being fired back at. Thirty shots a second from his two calibre .50 guns firing armor piercing ammunition. The sights worked and they did the job — we won the war, didn't we?

So don't let anyone knock mechanical analogue computers. If the airborne argument doesn't suit you, why, just think of the Senior Service. All that beautifully accurate fire and great ranges, many times over the horizon of the guns firing, was controlled by "fruit machines."

Read "Mechanical computer" for that last and you'll have a good idea of where the computer action was in the forties. It wasn't until 1945 that I saw my first electronic computer. This was the Central Fire Control system on the B-29, the plane that dropped the atomic bomb. All thermionic valves. I had no idea of how it worked, I would have had to re-enlist to find out and civvy street's lure was too strong. I must find out some day — because I notice that the histories of computers don't mention this one either.

Then oil ball carriages, discs, drums and felt wipers with Univis No. 48 oil.
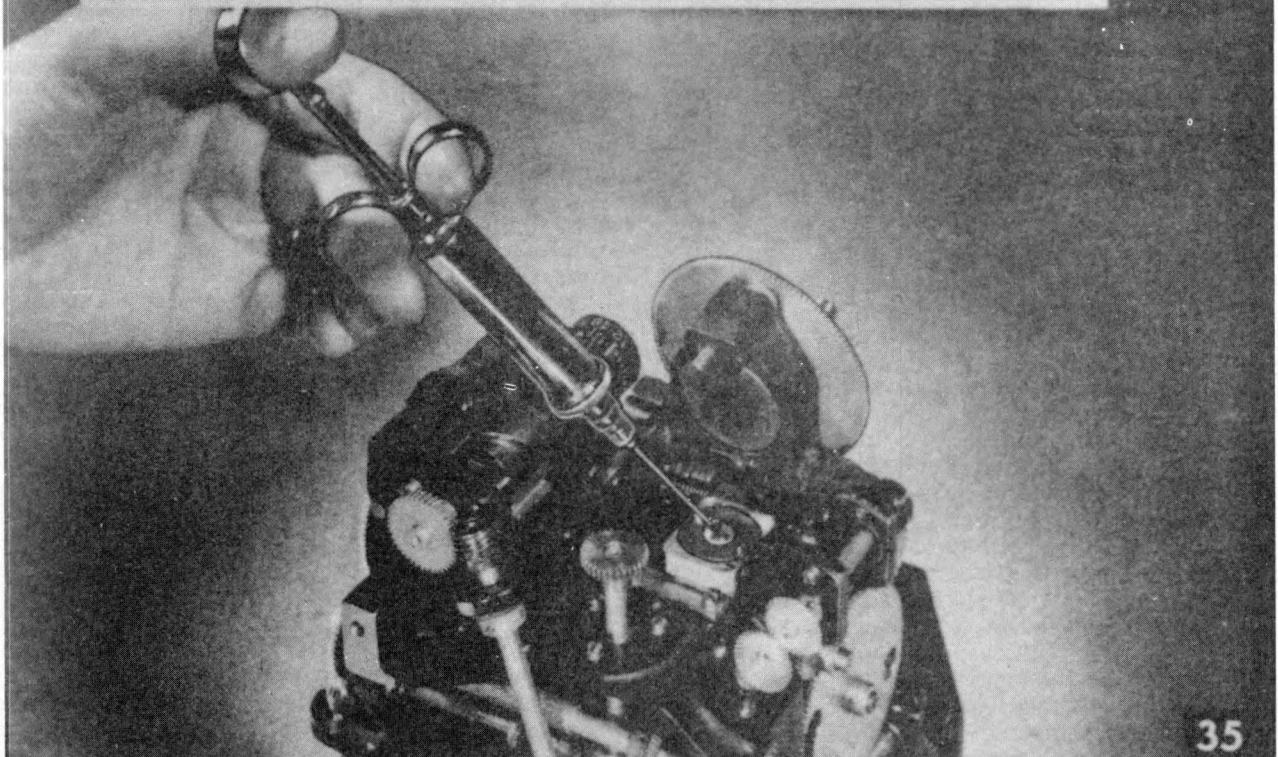
35

EXPLOIT OR CULTIVATE ?

COMMENT ON THE PERSONAL COMPUTER BUSINESS

Guy Kewney

The sad truth of the matter is that there is still no retail computer market in this country.

Retailers in most businesses expect to make between 35% and 45% profit on what they buy for resale — depending on the perishability of the goods.

Many of the new businesses just struggling to make ends meet by selling cheap computers in thie country are trying to make well over 100%.

If you don't believe this, just compare prices of American kits and systems as advertised in *Byte, Dr Dobbs, Kilobaud* and so on, with any price you may be quoted for *the same kit* from a 'retailer' in this country. Typically, the sterling price is nearly the same number as the dollar price — and remember, the dollar price *already includes* at least 25% profit, and importers here typically get a substantial discount. In one case I know of, the UK distributor gets a *bigger* discount than US agents, to cover costs of importing.

This is not retailing — or if it is, it is dishonest retailing — but the computer business in all its glory.

If you program computers yourself, or design circuits, you may not realise what 'all its glory' actually means. I will give you one example of the sort of price structure you get in the computer business.

Three years ago, working on the now-defunct journal Dataweek, I was shown a big disc drive being enhanced by a company which built systems around a famous make of minicomputers. With a special interface, this company sold the disc drive at around £8,000 — they built the interface themselves, and they bought the drive and the controller bog-standard from its supplier, for £3,000 or £4,000.

This drive and controller probably cost a maximum of £1,500 from the factory, estimated my friendly systems builder company. And, did I know, the *same* device formed the disc drive sold by company X? What did I think X sold it for?

X sold it for around £28,000.

I didn't believe it either, so I checked. First I checked with X themselves, who said yes, but I had to realise that the sale price also covered . . . and when they finished their list of excuses and I asked them what the *cost* price was, they refused to comment.

So I checked with the supplier and they admitted that the drive was basically the same machine as the one they sold, in small quantitites, for under £4,000.

But they said, although they sold it in quite large quantities to X, it wasn't quite standard — it had a different board of electronics components in it, to the company X's specification. No, they weren't prepared to say what it cost after the bulk discount was subtracted and the special board was added.

My editor decided not to print. this story — not because he didn't believe the figures, or thought there was some extra factor which ought to be added on the costs — but because he felt it would be unfair to publish the company X's name on this sort of margin if I couldn't also show that *another* company

Y did the same sort of thing — which he himself didn't doubt, by the way.

You don't have to look far for further examples of an industry which needs this sort of profit margin to cover its gross inefficiencies of design and the beyond the frontiers of technology, both hardware and soft. Why does a video display with fewer components cost four times the price of a colour television? Why does a matrix-needle printer with very few moving parts cost twice an ordinary motor-driven typewriter's price? Whatever your explanation, it better not be 'they cost more to make', They don't. They cost less.

Well, that's the computer business, and it works, on its own bizarre level.

It won't do for the computer retail business.

It's all very well for those who are just setting up their own businesses in this area, without a lot of capital behind them, selling ones and twos to start with, and having to cover the mortgage on a large family house and everything that goes with it. They can say that, selling to polytechnics, the Home Office, and wealthy central heating distributors, these margins are acceptable to the customer, and that without them, the business wouldn't get off the ground — but I wonder.

I tend to agree with Nen Norman (she won't tell you what Nen is short for and I promised not to) of *Computer Centre*, recently opened.

This lady started out her and her partner's business with an 8K RAM board for the Altair-type S100 bus, costing £85 in kit, £95 assembled — static, not dynamic. I announced my pleasure at finding a reasonably priced memory board, and incautiously told readers of my *Personal Computing* column that the Centre must be making only about £10 on it.

It just goes to show how you can get used to excess profit margins, I suppose, because the ink was still flowing over the pages when Nen's partner rang me up to explain that he was not making only £10 — nothing like it. 'A comfortable profit' was the phrase used, and we left it there.

But the Centre people then expressed themselves forcibly on the subject of other operators in the cheap computer business. The quote I have down in my notebook goes:

'There are far too many people guilty of suppressing the development of the amateur and private computer business, by overpricing.'

I agree.

The business is not going to take off until these operators realise they are in commerce, not in industry, selling to consumers, not clients — and, more important, that there is more money in it.

Perhaps I better say that again, for the slow and hard of hearing. There is more money in a big consumer market than in a small buying clique.

This isn't just a question of the economies of scale. It's a question of publishing, and software. On the face of it, there may be nothing to choose between

selling 50 computers and making £400 profit on each, or selling 500 and making £40 on each, except that the 500 take a lot more paperwork to shift.

But there is a very important difference. Each of the users of the 500 can pick 499 other brains and software libraries, and operate quite a lot quicker in achieving whatever they bought their system to achieve.

And after two years, when both sets realise that they can buy 100 times the computer power for half the price, and both the suppliers have to design Mark II or lose their customers, the 50-man will have only 50 customers already hooked on his software, while the 500-man will have a nice-looking user base — and probably a user base who are a lot more enthusiastic about the product. For one thing, they'll have more peripherals, more software, more general ideas . . . and for another, they can look forward to a cheaper replacement machine.

While the 50-set will have come to think of their supplier as something of a rip-off merchant.

The high-cost people are not rip-off merchants, of course. They are merely cautious — overcautious in my view — and don't yet really believe that the man-in-the-street wants to spend £300 on a computer.

They'll learn.

Book Review

Michael James

THE BEST OF BYTE VOLUME 1, *edited by D.H. Ahl & C.T. Helmers, 1977; 376 pages. (Creative Computing Press, paper £8.00)*

'Best of Byte Volume One' was published in March '77 and consists of a selection from Byte magazine from Sept '75 to Dec '76. Thus the usual glimpse of the future, expected by anyone used to reading up-to-date US publications, is sometimes missing. However, as a view of what *was* going on in the USA and hence what is (just about) going on *here* it is excellent. It is split into seven main sections, Opinion, Theory and Technology, Computer Kits, Hardware, Software, Applications and Resources, so readers have no excuse for not finding something of interest.

The longest section, with some 148 pages, is devoted to Hardware. Most of the articles do not give complete constructional details (i.e. printed circuit layouts are the exception rather than the rule) but the designs are generally well explained and should prove easy to interpret. My own vote for the most interesting item goes to the 'fast cassette interface'; working at 1100 baud, it would save a lot of time when used in place of a 300 baud Kansas City standard system.

The Software section at 114 pages is nearly as long, and if hardware is not your speciality then this, along with the remaining sections, should provide a good introduction to the non-soldering part of computing. If you would like to know how an assembler works, or how to calculate your biorhythm chart, or design an on-line debugger, then 'Best of Byte' will tell you. There are few complete programs listed but plenty of ideas and 'how to do it' notes. The description of how to compile algebraic expressions is not quite complete, but very easy to follow and succeeds in revealing a lot of the computer science inside a compiler.

The Computer Kit reviews are useful but are descriptions rather than evaluations, and a number of important recent machines are not included. The following *are* covered: the Altair 8800, SWTPC 6800, Altair 680, KIM, Z80, and the LSI 11. It is an interesting reflection of the differences between UK and US home computing that the LSI 11 at about £1000 is included as a computer kit! The most disappointing section is the one dealing with applications. With only two articles and nine pages it is, compared with the other sections, severely undernourished and leaves one thinking 'what *are* home computers being used for?'

At £8.00 'Best of Byte' is good value, and its large A4 sized pages hold a wealth of knowledge, opinion and ideas. It can be recommended to all but the absolute beginner.
ends tape 61

COMMODORE'S APPROACH TO COMPUTING

Kit Spencer

THE EARLY PET STORY

The story of PET begins with the development of the 6500 series of microprocessors by MOS Technology which then became the heart of the KIM I microcomputer system. Less than eighteen months ago Commodore acquired MOS Technology to give us our own in-house chip processing and design. Since then KIM I has become the best selling microcomputer system in the USA. (It is now available in the UK for £149 plus VAT).

PET effectively graduated out of the KIM I — the whole project taking less than a year from the first development "go ahead" given by Commodore after we realised the vast potential such a system could have and the fact tht we now had the necessary in-house technology, production and marketing capability for the fascinating PET project.

What Was The Design Philosophy?

Firstly, let me make it clear that this is not a technical paper but an insight into our overall philosophies. Several main concepts were used in designing our PET. These were that it should be AFFORDABLE, CAPABLE, EXPENDABLE, UNDERSTANDABLE, USABLE and SERVICEABLE. Let me explain each of these concepts more fully.

Capable

Although 'popular' media sometimes talk of PET in terms of game playing, I should make it very clear straight away that PET design parameters were *not* this but that PET should be just as capable as a traditional computer that up till now would have cost thousands of pounds. The PET was firstly designed to a demanding specification and then we set about deciding how we would get the price right for what we wanted to achieve.

Affordable

We felt that if we could offer a full computer capability at under £1,000 the market for computers would change from one of 'major company board room decisions on capital equipment' to a level of personal and departmental decisions that would open up dramatically increased market applications and potential. If this was so we could write off development costs over many more units and apply some of our mass production and marketing expertise that would make this an economic proposition for us as well as our customers — response so far appears to justify fully this decision, thank goodness!

Expandable

We designed our PET computer so that it could be

upgraded as the heart of an extremely powerful system. For this reason we built in an IEEE-488 output (often known as the H.P-I B) which allows for multiple intelligent peripherals — we have under development our own printer and floppy disc.

The 8K Ram of the machine is designed to be expandable in 8K increments up to 32K and these will in the future be offered as optional extras. There is also a parallel user port and a second cassette interface (the second cassette is another optional extra). I O is managed by built in operating system software.

We believe all this makes our computer system very expandable.

Understandable

We have tried to take some of the mystique out of computers in order to broaden people's understanding, appreciation and potential use. For this reason we chose an expanded, fast, comprehensive and powerful 8K *Basic* which has been built into the machine (part of our I4K Rom) as this is one of the easiest computer languages to learn, understand and use. There is also machine language accessibility for the professionals.

Our PET computer has been designed as a fully integrated and independently operable unit that can be used from the day it is purchased by simply plugging into a standard mains supply. For this reason a 9" video display, built in cassette and keyboard were designed into PET — PET can be used as an excellent teaching machine and, in its direct mode, is an excellent programmable calculator. (There is even a built in clock for timing applications — settable directly from the Basic via a simple keyboard entry.)

Personal

This is perhaps one of the slightly misrepresented aspects of our PET computer. It does apply to the price of the unit, to the fact that it is portable and to the fact that it can be operated even without any of the expansion capabilities being added — it certainly does not mean that it cannot be used other than as a personal unit. We know the unique graphics capability makes it a fascinating 'personal computer' for the hobbyist programmer but that's only one aspect of PET.

Serviceable

When you open up a PET to look inside you will immediately realise it has been designed for ease of servicing and economic mass production techniques.

The PET consists of only three p.c. boards — the main board which can be snapped out and easily replaced in the event of any defect and the other two boards which are for the CRT circuit and the cassette recorder (specially modified), both of which are removable and replaceable for ease of servicing — and assembly, of course!

The other components are a power supply, CRT and a keyboard. The PET has been designed for the person who wants to use its capabilities and not for the man who wants to poke his soldering iron around — at least inside the machine.

What Is The Future For Our Pet Computer?

As an International Corporation we have already made major investments in this market and our efforts will be continuing in three areas. peripherals, software and marketing. All of which are going to be extremely important.

What About Peripherals?

There are no specific details yet (I don't want to be misquoted) except to say that our first plans, already well under development, include a second cassette deck, a printer, memory expansion and floppy disc. But remember, if you don't like our units or can't wait there are others compatible with PET. For those
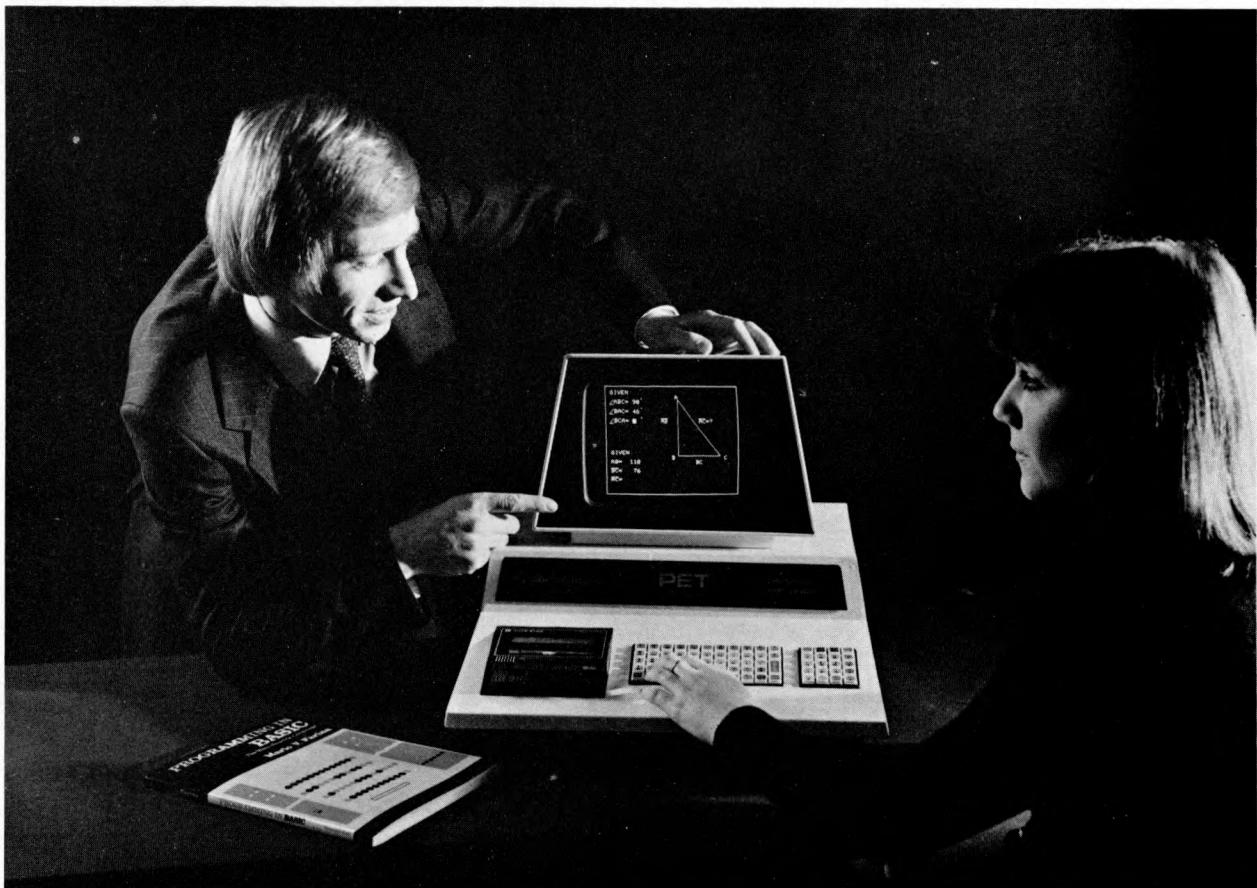
A VERY SMALL BUSINESS SYSTEM?

Martin Collis

In the past, most software houses in the U.K. have been unwilling to invest in hardware, the usual reason given being that installing a system would affect their independence. In practice in-house systems must greatly increase productivity and, as hardware prices continue to fall, it will soon become normal for most software houses to own a considerable amount of hardware. As we are a very small software house, we have started by installing a very small system, an SWTP 6800 with a teletype © and twin mini floppy disks.

Our reasons for installing a system were: —
1. To see if we could put a computer kit together and make it work — we have no hardware experience.
2. To see if a very small system could be used for commercial applications without the low hardware price being swamped by software and maintenance costs.
3. To have an in-house system available for training.
4. To get some experience, and hopefully some work, of micro software development.

We decided to base the system on a teletype © as, if all else failed, we would at least be left with a terminal which we could use for punching paper tape, time sharing development, etc. We were lucky to be able to buy a nearly new Data Dynamics ASR 33 with a V24 interface.

At the time when we were looking for a system, which was last summer, the only possibilities were: —
1. The SWTP 6800 system from Computer Workshop
2. Three "S100" systems from Compelec, the Altair, Cromemco, and Polymorphic

We chose the SWTP system because: —
1. It was the cheapest.
2. There were (and still are) plans to manufacture it in the U.K.
3. The Motorola 6800 seemed to be more widely used in the U.K. than the Intel 8080 on which the other systems were based.

The System
We ordered a basic system (processor, 4K of memory and one interface), an additional 12K of memory, a second interface and a cassette tape controller.

With the exception of an 8K memory board, and the cassette controller, the kits were delivered about ten weeks later. We bought some basic equipment, a low wattage soldering iron, wire strippers, multimeter, etc. and set to work.

With two of us working mainly in our spare time, the processor took about two weeks to assemble. The instructions were reasonably clear and our soldering certainly improved with practice. It is quite obvious from looking at the system, which parts we assembled first!

Testing & Debugging
As the SWTP 6800 includes a monitor in ROM and 256 words of RAM on the processor board, the first system tests can be performed without any memory. We assembled the interface board, connected the teletype and switched on. At this point the system should respond with "*" on the terminal. We got no response. We pressed RESET a number of times, this should have given "*" each time, still nothing. By poking around with the multimeter we established that RESET was not working. I also shorted +12V and −12V supplies on the mother board which blew the power supply (but not the fuse!).

We took the system to Computer Workshop, who fixed the power supply and then established that the RAM chip was duff. They replaced it and the system then worked. This took about 45 minutes, was done as soon as we arrived, and was free of charge.

We went home and assembled the two 4K memory boards. The diagnostics showed up faults on both of these which we were unable to fix. So we took them back to Computer Workshop. This time they were busier than when we went with the processor and we left them to be fixed. The faults, a

duff chip on one board and a solder bridge on the other, were again fixed free of charge, and the boards were ready within three days.

Now we could run the 4K BASIC, very basic, but there was still no sign of the 8K memory board or the cassette interface. This was very disappointing as four months had now elapsed since we had placed our order.

In a fit of extravagance we cancelled the cassette interface and bought twin mini floppy disks and controller as these were available from stock. The 8K memory board arrived soon after. We set to work again and the disks and the 8K board both worked first time. This I suspect was luck but it might have been because we were now more experienced.

Software

We now turned our attention to the software. The first release of the mini floppy operating system included BASIC but not data files. However as a new release of the operating system was promised soon (before Christmas!) we concentrated on getting a variety of games on the system. The BASIC is quite good although it seems to be rather slow and we didn't find any bugs. The system is kept in a room which can get very cold and this seems to give rise to quite a lot of disk errors.

The new operating system finally arrived at the end of January. To our dismay it proved to be not version 2 of the original system but version 0.9 (pre-release?) of a new system. The documentation was hand written and, more dismay, we had to re-locate 4K of memory to 28K-32K addresses to run the system.

This meant that only 12K was available for BASIC and the application program and we would have to buy more memory to run reasonable sized programs

(shades of IBM). The final blow was that although data files were supported they could only be read or written sequentially. We are currently waiting for the next release of the operating system — we have also found a number of bugs in 0.9. We are also looking at an alternative operating system which does offer random and sequential files.

To summarise our experience to date: —
a) The hardware does seem to be robust and reliable.
b) The SWTP system software still has some way to go before it could be used as the basis for a small commercial system.
c) If you can go and see them Computer Workshop are very helpful in fixing hardware faults. Their software support is less good.

Once we get a good basic operating system our first step will be to develop a set of standard programs for setting up, maintaining and printing data files, for generating indices and for selecting and processing records via the indices. This set of programs will then provide the basis for a number of standard applications such as invoicing, word processing, stock control etc.

How much?

A final word on costs. The processor, twin disk drives and 24K of memory cost about £1,700 in bit form. A vdu and 30 cps printer would cost about £1,500 giving a total hardware price of £3,200. Our aim is to provide a basic software package which the user can extend to meet his requirements. It is difficult to price this at the moment but it is unlikely to be less than £800, giving a total price of £4,000. Using full size floppy disks would increase this by about £900.

IN A FORTHCOMING ISSUE
PCW REVIEWS THE TANDY TRS-80

*If what happened so far is anything to go by, customer relations are high on Tandy's list of priorities. We have been offered a machine for review, for a reasonable length of time.
Barney Price of Tandy took delivery of the first two hundred TRS-80s in March. The TRS-80 is built around the Z80 microprocessor. In the photo above it's seen "in action". Tandy is offering or is developing pre-recorded cassette programs for a wide variety of applications. The TRS-80 microcomputer with regulated power supply, less video display and data cassette recorder, is priced at £499.

And . . . just some articles in THE NEXT ISSUE
*Interfacing a Westrex punch to your microcomputer.
*Win the pools! At least, have fun with a "pools" program.
*The Z80 instruction set.
*Are you a microcomputer widow?
*More hints on how to program.
*In praise of the PDP-11.

. . . It's the unique **PCW** mixture: articles for the beginner, hobbyist, recreationer, businessman and expert.
(Please note that for reasons of space *The Gates of Reason* and *Buzzwords* will appear in the next issue.)
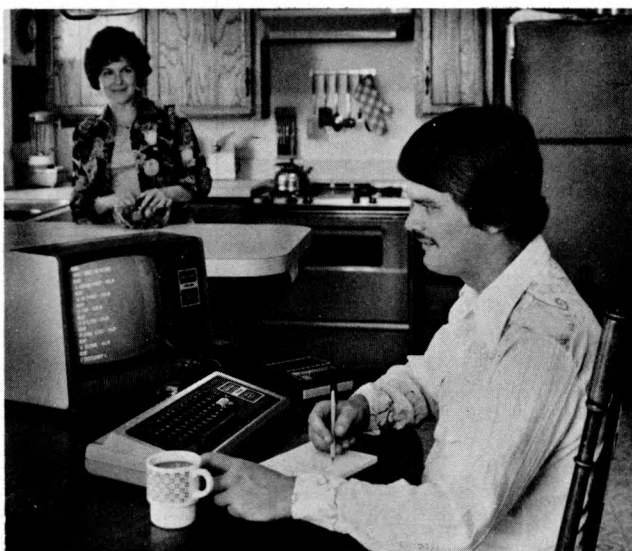
THE APPLE II — Full **PCW** Review
Mike Sterland of *Personal Computers* Ltd., who believes that "Apple will become a major force in the Personal Computer market", has generously loaned **PCW** a machine.

PCW CASE STUDY

Boris Sedacca

The following case study should be of interest to the hobbyist but is not written specifically with the hobbyist in mind. The intention here is to show that an average small business proprietor can configure and develop a fairly sophisticated small business system around a microcomputer.

Firstly, an outline of some parameters which were set in the selection of this particular case; the over-riding constraint was that the total hardware cost should not exceed £10,000. Software was excluded from this pricing exercise because, apart from the fact that one would be hard-pressed to find a workable system to include software under £10,000, the intention here is to show that one does not have to be a "computer technocrat" to develop one's own with applications tailored to suit the business's particular needs.

What makes this a lot easier is the use of BASIC; an acronym for Beginner's All-purpose Symbolic Instruction Code. Basic's concise English-like commands allow for quick mastery of programming concepts, and its interactive mode sends out prompts and error messages to the student programmer at his terminal.

The main crticisms of BASIC are: slow run time execution through its interpretive stages (the commands are basically sub-routines), the need for extensive documentation to append programs (COBOL, although much more verbose lends itself to automatic documentation), and for more sophisticated versions of the language, high memory overhead.

A counter-argument to the first criticism is that the time-related cost of developing a system using assembly code programming is much higher than the savings achieved. However, the economics are more complex if a program is to be copied a number of times as a standard package, where it would probably be more cost effective to have a software house write an assembler version of the BASIC programs.

As for memory overhead, the reader will no doubt have heard of falling costs of memory due to MOS/LSI technology advances *ad tedium*. However, cost is not the only factor to be considered; most 8-bit microprocessor CPUs have limited addressing capability.

Not that BASIC is doomed to being an interpretive language indefinitely; Jacquard for one has written a BASIC compiler to supplement its interpreter for debugged programs to be converted to code. The Jacquard CPU is made up of a National Semiconductor IMP-16 consisting of four 4-bit slice chips so the coded software is not cross-compilable.

The term "microcomputer" probably needs some qualification. The generally accepted definition for this is that the CPU is made up of a single microprocessor chip. Another definition is in the use of LSI (Large-Scale Integration) as opposed to the MSI (Medium-Scale Integration) technology used in most minicomputers.

Obviously there is a certain grey area as more and more minicomputer manufacturers use LSI chips on their boards. As a further refinement to the definition

one could regard the microcomputer as a design where the chip is used to handle all operations centrally on the classic von Neumann model, heavily dependent on the single processing resource. This has obvious implications in terms of instruction execution speeds and is one of the many criticisms of the microcomputer in its present form.

This definition should not be taken too literally however, as Fairchild's F8 would be automatically disqualified. At least three chips are required to make up a CPU capable of addressing 64K Bytes of memory: the 3850 CPU chip proper, the 3851 Program Storage Unit (PSU), and the 3852 or 3853 memory interface.

Another limitation of the microcomputer in its use in commercial DP is its difficulty in interfacing to a high-speed DMA advice such as hard disk. Most microcomputer systems in their present stage of development handle their mass storage I/O through a multiplexer bus to floppy disk devices.

In addition to its relatively slow speed the floppy disc has a limited storage of 256KB or 312KB, the former coming from IBM or IBM-compatible devices. IBM has recently announced dual density, double-sided floppies which brings the storage capacity up to just over 1MB per diskette. This allows for substantial expansion of file storage on their IBM 5110 up to 4MB.

A microcomputer however should not be seen as being perpetually upgradeable to any number of peripherals. If a business is going to require more than 4MB (megabytes) of on-line storage and a fast printer for a large volume of reports, it would be better off with a small business system offered by a mainframe maufacturer or by a turnkey systems house.

Limitations aside it is possible to configure a small business system around a microcomputer, and an increasing number of them are coming on the market. Certain software houses are offering standard packages for these systems, written mainly in assembler as they can afford the capital outlay on debugging and development kit.

The following case study, then, shows how a small business proprietor went about installing a computer system at a price less than £10,000.

THE COMPANY: Computech Systems.

THE BUSINESS: Computer staff recruitment agency.

The background: Computech was begun in 1966. The original intention was to set up more than just a recruitment agency and to write software for accounting machines and visible record computers. However the recruitment side became increasingly busy and the other side was not developed as a result.

Computech currently employs a staff of nine full-time employees and turns over in the region of *£120,000*p.a.

THE SYSTEM:

EXTEL Microfile 32K processor ⎫	£3,500
2 PerSci floppy disc drives 312K ⎭	
30 cps EXTEL intelligent buffered terminal	£1,950
Upper & lower case VDU	£750
16KB BASIC User licence fee	£300

L to R: Laurence Payne and Boris Sedacca

Operating utilities
Text editor & elementary word processing
File handler

Microfile is built by a Californian Company, Data Terminals and Communications, and sold in this country by EXTEL, the DTC U.K. agents.

The box itself houses the floppy drives and includes slots for printed circuit cards which would include memory, the processor chip — Intel 8080A, the market leader and nearly the standard device for use in such systems — and the electronics to control the interfacing of the VDU and the printer terminal.

The PerSci floppy disc offers faster data access than most, averaging 200 milliseconds to find and retrieve characters, where on most units the average tends to be more than 300 milli-seconds per character.

Laurence Payne is the founder and proprietor of Computech systems. He began business in 1959 in a partnership of Chartered Accountants but Computech was launched in 1966 as his first commercial venture.

"Although I did not find the time to develop any software in those days I did buy and sell small business systems on an ad-hoc basis", he says. "These were based on the Olivetti P203 and Adler machines programmed for commercial applications.

"One could buy these machines extremely cheap second hand as the hardware value was low without software. Software itself had no resale value.

"This does not apply with the new machines programmed in BASIC as programming is much simplified here. Furthermore you can protect your investment when selling the machine as it is made up of solid state electonics which do not deteriorate or become obsolete as quickly as the old hardwired machines."

As the agency business grew Computech sub-contracted its own accounting requirements to an accounting machine bureau. The workload comprised mainly payroll and invoicing for a large team of temps.

This system was found not to be satisfactory for various reasons. There was a slow turnround time and lack of control over debtors. No records were readily available at the office where they were needed. As the agency developed a policy decision was taken to discontinue hiring out temporary staff and the outside bureau service then ceased to be cost effective. Finally, there were no interactive facilities for registers of applicants. "The system we are using now is capable of both accounting and word processing applications.

"We developed our own text editing package to supplement the Microfile's. The system now allows for the operator to start a new letter just by using the keyboard and having commands appear on the screen. These interactive facilities make this process completely self teaching.

'All the routines we've written are in BASIC
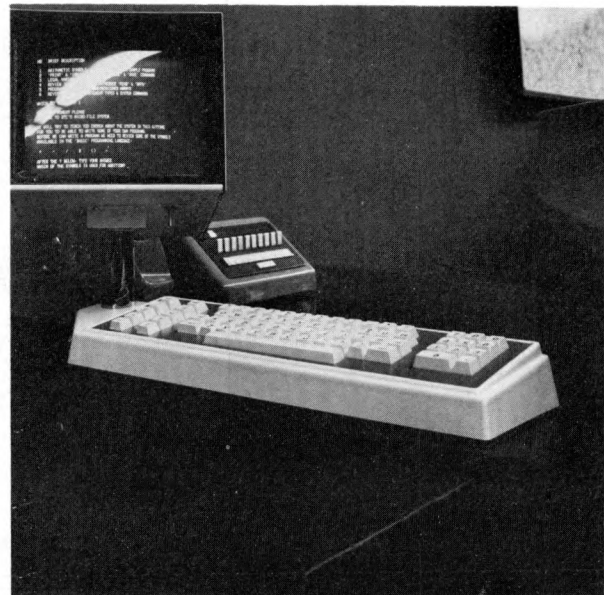
including multi-access for the connection of several terminals. The standard version is only single user BASIC.

"We have a total of around a thousand applicants on our files which are held on diskette. Only about three hundred are 'live' at any time and we can hold a fairly large record of their particulars on one diskette. From these we can make parameter-driven enquiries all under operator control.

"The link with word processing on these files allows us to merge the files and send out a great deal of letters to clients and applicants. This morning we sent out about 130 letters most of them standard in format with variables such as names and addresses in the text. All we do is key in a few control instructions and the machine does the rest. The operator is free to carry on with other work while the letters are being typed out. Although the need for file searching was the main reason for implementing the system, text processing has turned out to be a major application."

Laurence Payne had started looking around for hardware since 1974. The concept for the system had existed long before but around that time there were many minicomputers being announced which made it possible for a company with a total budget of around £30,000 to buy their own in-house computer. The problem here was that a system built around a mini would have needed a great deal of effort in terms of software and a software house would have had to be found.

Laurence then looked at 14 different minicomputer installations and found he favoured the DEC PDP-11 with RSTS operating system.

"Fortunately, we held our horses. Firstly, this system could not be built a step at a time. We would have had a lot of spare capacity during the devlepment stages. Also at that time economic conditions and unemployment in all sectors were not conducive to an increase in capacity.

"Another factor of course was the amount of floor space required to house the machine. A mini is certainly smaller than a mainframe but it is by no means small. Then of course we weren't too sure that we wanted one system to handle all our processing; if it went down our whole business would suffer.

"The next thing we looked at was kit computers but rejected this idea as we couldn't get a manufacturer's maintenance contract on them.

"At an exhibition at the US Trade Centre I saw the DTC Microfile for the first time and met the company's president, Roy Worthington. It seemed this machine could do what was wanted. Although the DTC box only allows for two terminal ports, expansion facilities have been built in by EXTEL for a four port device to enable connection of more than one Microfile together. This allows for redundancy when one machine goes down; a facility not available on the PDP-11.

"Using this machine we will be able to fully computerise three of our four agencies for a *total* outlay of about £20,000. With a minicomputer-based system, it would have easily cost double the amount."

Computech is now in the process of expanding its configuration with a second processor containing four disk drives, and replacing its VDU with four Informer VDUs. Informer did not provide 80x24 upper and lower case screens previously but Computech has just received the first lot direct from the U.S. manufacturers. Laurence Payne expects that the combination of Microfile processors with polled terminals will give him more computing power than with a mini system.

"As for other microcomputers, we did not really find anything that provided dual disk drives and BASIC at that price at that time. As it turns out now we have had a company order a similar kit and ask us to do their programming for them. I think this kit can be pretty competitive as a lower-end key-to-disk system with Redifon and CMC systems."

With the dual processor configuration it will be possible to have one spooled for printer output while the other will be free for processing independently of the printer.

"We've been so successful with this machine that another employment agency has now approached us to install a similar system on a turnkey basis. The INFORMER terminals are ideal for this type of application. The screen contains the full 24x80 characters but is much smaller than average and sits on a stem to the side of the keyboard, which makes it easy for an interviewer to enter particulars of applicants without having to crane her neck over the screen."

This is all quite an impressive record considering that the hardware was bought only in November 1976 and first live running was already under way in April 1977: from do-it-yourself user to systems house in under eighteen months!

"Yes it does cost more but it does everything a cheaper one can do."
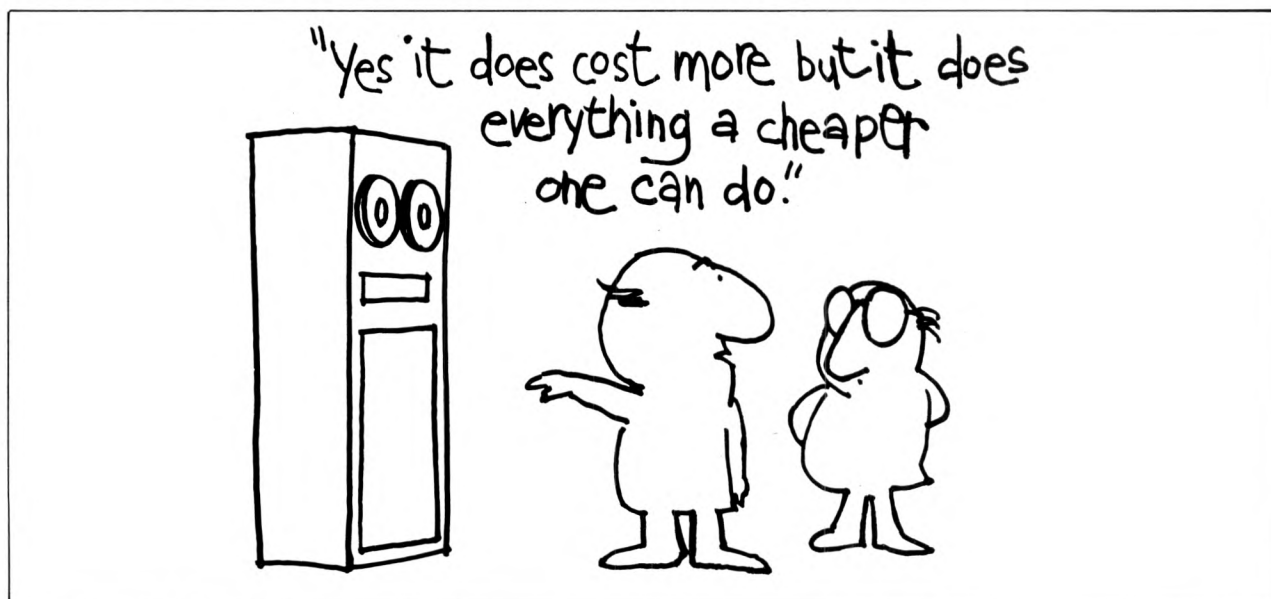
MK14-the only low-cost keyboard-addressable microcomputer!

The new Science of Cambridge MK14 Microcomputer kit

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.
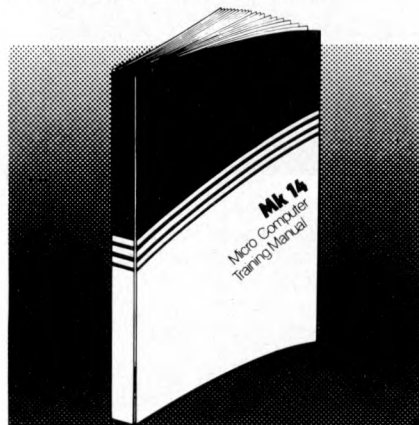
Designed for fast, easy assembly

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.

Just £39.95

(+ £3.20 VAT, and p&p)

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

THE ENTREPRENEURS
PCW INTERVIEW
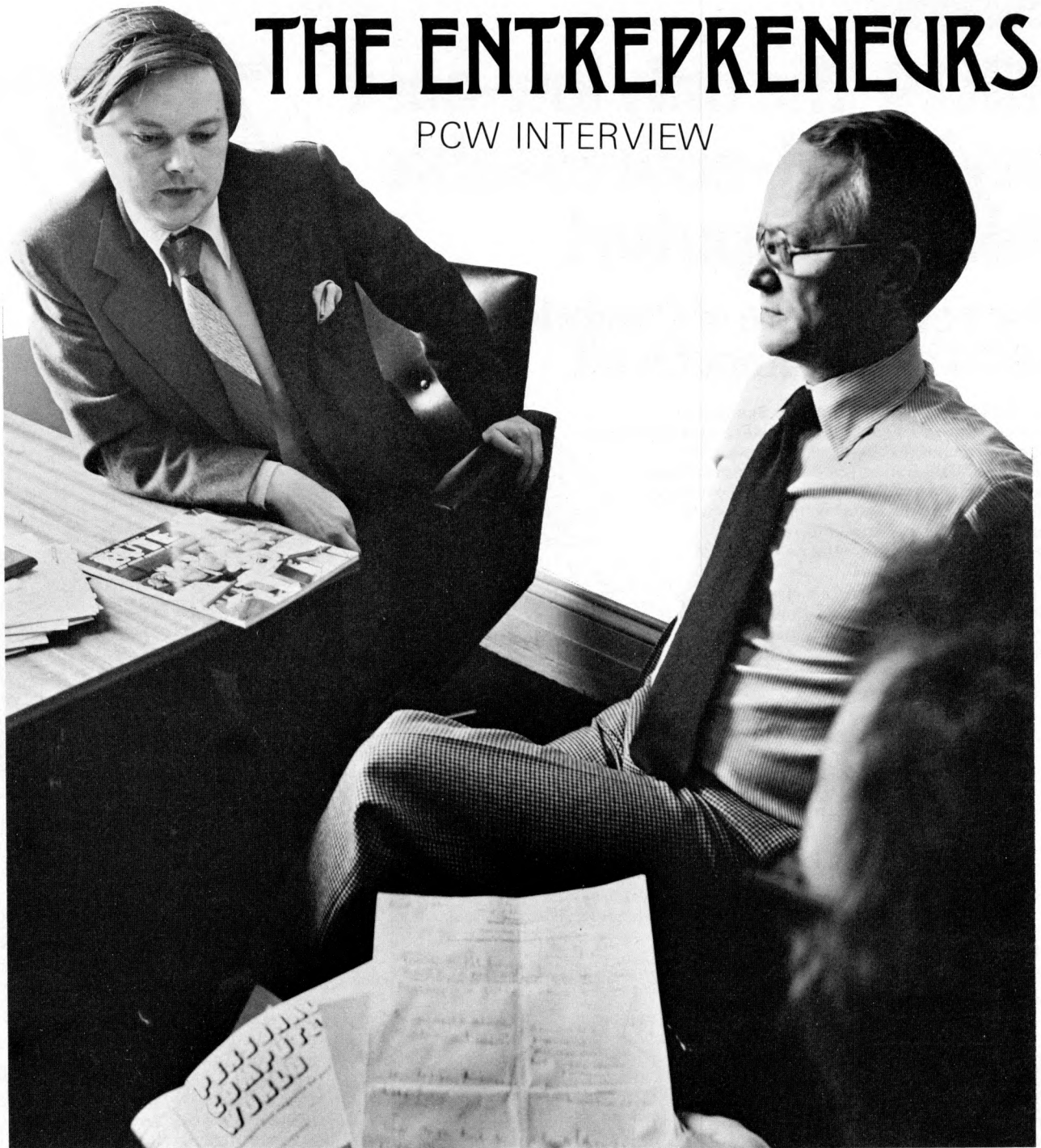
Ten minutes after we arrived at Computer Workshop, London, photographer Yoshi Imamura discovered he had a faulty camera and rushed off for a replacement. So the rest of us went upstairs to a little office to talk, with the secretary giving a busy and entirely appropriate background to the **PCW** interview with **John Burnett** and **Gordon Ashbee** of Computer Workshop.

PCW Creation stories always begin at the beginning . . .

Burnett We opened for business on 1st May 1976.

PCW How did you get together?

Burnett We'd known each other for eighteen months before. I called on Gordon as a representative and it went on from there. Over drinks we would build electronic castles in the air. I was then working for Computer Services Centre in Acton, London, and computing had been Gordon's hobby for three or four years.

PCW Give us an idea of your backgrounds.

Ashbee I did National Service as a fighter pilot in the RAF, worked for Coty, was a representative, then in 1958 started my *Entryphone* business. I had to be independent.

Burnett I did articles as a clerk in a firm of chartered accountants, then trained for two and a half years in a computer bureau. After that I had a short spell with ICL,

then joined the Computer Services Bureau as a representative. I was initially involved at all levels — I liked that. But later the usual job routine took over, I found myself a join in a rigid structure, a set-up which of necessity led to a division of labour. Eventually, not for me.

PCW When did you connect with Southwest Technical Products?

Ashbee We were ordinary customers at first, but we happened to be quite big ones. From there it was only a step to arranging a franchise. Incidentally, there is now a factory in Peterborough which is a joint venture with the Americans but separate from Computer Workshop.

Burnett The idea of starting Computer Workshop was Gordon's. I began by getting business from him and ended by getting into business with him.

PCW What do you think of the hobby market?

Burnett Ah yes, the hobby market. Is there a genuine future for it? I'd be able to answer the question better if I could identify the hobby market. What is it? Where is it? At the moment it seems to me that it's subsumed in a company ordering a small system and whoever using it indulging his hobbyist inclinations together with bread and butter work. It's different in the United States because

there the level of personal disposable income is higher and the tax structure different. Over here I'd say that the majority of so-called amateurs are trying to make a business out of their hobby.

Ashbee I agree. In the US the amateur is truly an amateur. In Britain — up to now at least — people don't want to spend money on a system without regarding it as an investment which must bring a return — in money terms.

Burnett Here the genuine amateur will go for a genuinely low cost system — a barebones. In America he is as likely as not to wallow in expensive peripherals.

Ashbee Think about it. Systems with 24k of memory, floppy disks with sophisticated operating systems: in reality, general purpose systems, sometimes with a lot more power than the user has the time or ability or enterprise to use.

Burnett And the amateur over here can't really substitute cassette for disk. Cassettes are a lot more awkward, a lot less flexible.

Ashbee Mind you, I see more dramatic falls in prices of peripherals — except for printers. There's a big market awaiting the first person who comes out with a really cheap and effective printer, one retailing at under a hundred pounds. Even floppy disk prices are falling. For example, drives are about two thirds the price they were a year ago. The problem as always is with the mechanical parts. Their prices tend to remain static.

PCW How about the education market?

Burnett It's very large and it's coming to us.

Ashbee Computer education is percolating down into the schools.

Burnett Many teachers are eager to introduce personal computing to schools. There was a case where a science department bought a kit out of its own pocket.

Ashbee The future of personal computing — in the home and business — is being created now in the schools of our country. What the future will be depends on how sensitive education authorities are to the computer revolution, how flexible and farsighted they are.

PCW Can we now get down to business?

(The question was auspicious. Yoshi returned with an eager new camera and acrobatically began taking photographs).

Burnett Computer Workshop business with businesses is brisk and increasing rapidly. To give you an idea, in the very early days a businessman with no computing background rang us up, more out of curiosity than anything, to ask if we could help in processing a large number of forms which needed forty minutes each in handling manually. It seemed to me like a thirty line program in Basic. I took the details, wrote the program overnight, then he came over for a demonstration. The following week he ordered a system. All this time he's been using just that one program and is still saving money, effort and time. Now that he's more aware of the full

power of the SWTPC 6800 he wants to go on to word processing. We've suggested adding a floppy disk and he's in line to upgrade his system so that it's general purpose.

Ashbee I might point out that in education people are more concerned with systems software, because many schools and nearly all universities are perfectly capable of writing their own applications software, a lot of it brilliant. But in small businesses there's a need for applications software. This offers profitable opportunities for small systems houses and even free-lance programmers.

PCW Lets go on to customer relations. Are you on the side of the angels?

Ashbee We try to be! We have very few complaints on either the hardware or software side. We do have occasional complaints about delivery.

Burnett Now that the factory in Peterborough is a fact, complaints should get even fewer. It's hard for people to realise the difficulty we've had setting delivery dates. In each system, there are roughly two hundred and fifty different types of components from various manufacturers. It only takes a shortage of one of them to affect delivery. Which is not to say that we ourselves don't sometimes make mistakes.

In the early days we had friendly relations with each of our customers — we were small enough then. It paid off because our reputation is founded on the soundness of the SWTPC 6800 and this goodwill. Now we're expanding at a terrific pace — currently at nine per cent a month, cumulative. We do have a self imposed restriction on growth since we're working at full capacity and not advertising indiscriminately.

Ashbee It's a tribute to the product.

Burnett I really regret that now we don't know every customer by name. With the best will in the world, we can't. When customer relations become more formal they are in my opinion not so good. But we do still give individual attention to someone who has a problem. We do our best and we're proud of this.

PCW So, do you regard yourselves as amateurs in business?

Ashbee Perish the thought. Our approach is professional, though we have an amateur's love of computing. Let's say that one does one's best under intense pressure.

PCW Finally, what do you think of possible Japanese inroads into the personal computer field?

Burnett Not what but when. It's only a matter of time. The Japanese are currently investing hundreds of millions of dollars in computing in all its aspects. It is Japan's intention to become the world's first information society, ahead even of the US. As other markets run down or are restricted the Japanese are turning to one of the biggest growth areas of all time — computing. Their impact is sure to be felt in personal computing even if only as a by-product of a major effort in big computers.
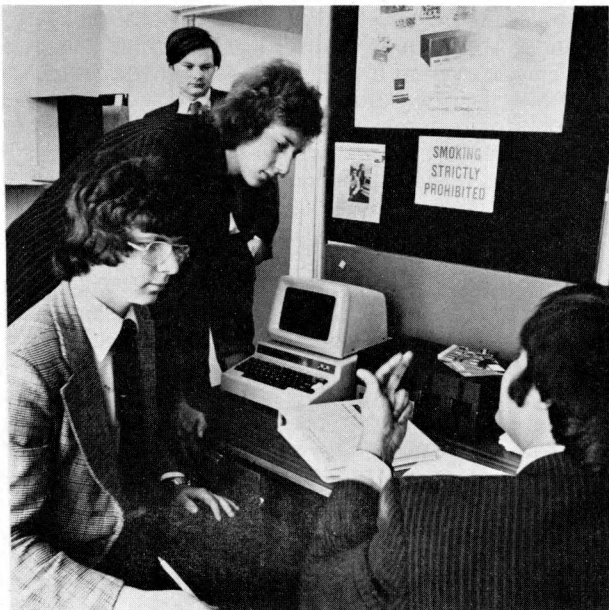
PET 2001 Review

by John Coll

PCW The PET is a very attractive small computer in appearance and design. It is a successful attempt at placing computing within the reach of thousands and there seems little doubt that thousands worldwide will buy it. Thus features which now seem to be non-standard will come to be accepted and it is probable that software and hardware which is CBM-compatible will be offered in the same way that there are now alternative choices to IBM software and peripherals. PCW

Let me start by giving you my overall impression of the PET 2001:— it is a very well constructed machine with many nice features but I was unable to spend more than a day with the machine: a week would have been more appropriate.

The machine is well built, solidly constructed and has a bright sharp V.D.U. display. The internal construction is very clean — a single printed circuit board carries all circuitry except the power supply and monitor scan circuits. Connections to the outside world are made via 4 edge connectors directly on the main p.c. board. I found the calculator style keys on the keyboard too small to use comfortably at speed. The integral cassette drive resembles an ordinary cassette transport but it was quite impossible, in the time available, to examine it in detail.

The keys normally produce upper case output on the screen, but if the "SHIFT" key is depressed then the displayed character is a graphic. The graphics characters are not of the usual Viewdata/Ceefax/Oracle "3 squares by 2" format but instead are generated by a "graphics ROM". This can generate such characters as "hearts" and "diamonds" as well as a good range of lines and shaped area from which complex pictures can be easily built up. This is a very interesting departure and makes picture-drawing very straightforward. The lack of compatibility with other systems may well be a price worth paying.

Getting into lower case, however, is not so easy. The command is "POKE 59468,14". Having executed this command the display will show upper and lower case letters but not graphics characters. "POKE 59468,12" enables upper case and graphics to be displayed but not lower case letters. That is a pretty inconvenient facility. POKEing 59468 with anything else usually causes the system to crash with total loss of program and data — but fair enough, one expects the command POKE to wield havoc if used carelessly.

The BASIC interpreter is resident in Read Only Memory — which is convenient in some ways but it does mean that if (when) any bugs are found or extensions to the language are needed there will be

nothing that you can do except buy a new set of ROMs from CBM, and that could be very expensive. It also means that if you find a better version of BASIC available from someone else you will not be able to put it in without buying a whole lot more memory from CBM. The interpreter has been written by Microsoft who supplied Altair with their BASIC. The timings of those benchmark programs which were given in the last issue are:

BM1 1.5 s
BM2 9.7
BM3 18.2
BM4 20.3
BM5 21.8
BM6 32.4
BM7 50.9
BM8 12.2
BM7+8 63.1

The really outstanding feature is the screen edit facility. (See Patrick Sutton's article in this issue — Ed.) One can move the cursor to any line of program that is displayed on the screen and with single keystrokes one can insert, delete or change any character. On hitting the RETURN key the new line is input to the interpreter. Thus small errors are very easily corrected. One can also use this feature to enter a series of identical lines with different line numbers. To do this one enters the line in the normal way and then edits the line number before hitting RETURN each time. That is a really useful facility.

There are two major omissions in the software. The first is the lack of a RENUMBER command. Yes, one *can* manage without it, one *can* manage without almost any particular command, but anyone who has used an interpreter with RENUMBER will know that it is vital. (Do you remember that strange idea that one heard once, that Computers were meant to save you time? No . . . perhaps I imagined it.) The other major omission is the impossibility of stopping a program in mid-stream, changing a line or two and then continuing with variables intact. If ANY changes are made to your program then *all* variables are zeroed. When a program is saved the status of variables is not automatically saved — so if you intend doing a two hour run then you had better make sure that your own "variable save" program works first — and don't decide half way through that you want to save one of the other variables!

Strings are limited to 256 characters and there is no PLOT command to match that found in other interpreters. It is not possible to tell BASIC to leave a section of memory free for machine language subroutines. BASIC stores strings downwards from the top of memory and BASIC program text and variables, upwards, so you have to take a leap into the dark when trying to place any machine language routines in memory. There is one 192 byte block of memory available between $033A and $03FF. The Random number generator is not a True Random Number Generator, as stated in their white leaflet, but is the usual pseudo-random sequence generator. It seems to work well. One slightly awkward feature is that although the machine works in ASCII
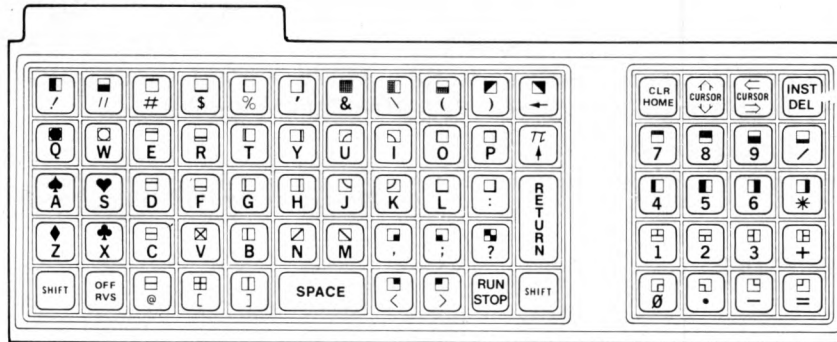
The keyboard

internally the V.D.U. does not store alphanumerics with their normal ASCII code. A is stored as 01, B as 02 etc. down to Z stored as 26. Zero to nine are stored with codes between 48 and 57. That seems really strange since the ASCII code is one thing that everyone understands, and it surely would have been just as easy to put the right code in the right place in the ROM. This is going to make interchange of program material difficult.

The machine is certainly individual: non-CUTS standard cassette, BASIC in ROM, non-ASCII VDU display, IEEE interface not plug compatible; and they "PEEK protect" the screen editor ROM so you can't use the disassembler to look at the routines they use to convert the internal ASCII to their own VDU code.

Talking of disassemblers brings us to the machine language environment. Well, quite simply, there wasn't one. There was no machine language monitor in any form — nor was data of any system made available. I understand that the KIM monitor is on its way. There is no editor (in the accepted sense of that word) or assembler. I did use their disassembler which was written in BASIC. I was told that it would be available for £10 and that seems reasonable if it proves to be so. It is quite adequate though it did not attempt to label statements that were referenced elsewhere or attempt to deal with text strings in any way. I did not have enough time to test it thoroughly, though. As I have said, areas of ROM are PEEK protected and therefore cannot be immediately disassembled. The protected area runs from 49152 to 57599 ($C000 to $E100) and includes the whole of BASIC, and at least some of the screen editor routines (if their memory map is accurate). Clearly anyone working with machine language text routines will need to use similar (identical?) routines. He will therefore need to copy the ROM programs to an area of memory that is not PEEK protected and set the disassembler going on it there — as an attempt at protection it is totally ineffective, and could be an irritant.

At present there is no floppy disk system or hard copy unit and C.B.M. are not at present giving any indication of when they are likely to be available. Anyone who has used a computer will know that these two are absolutely essential in the long run. I will be very interested to see the pricing of these items. Will they be as much of a bargain as the PET itself is, or will they carry a substantial price tag which the already committed PET owner will have to pay? Anyone want to place bets? I think prospective users would be well advised to wait and see.

The cassette system uses a non CUTS standard format, which is unfortunate, and in the time

available I was unable to see if it would be easy to write machine language software to read and create CUTS format tapes. It is not clear from the introductory booklet whether the second cassette that the system can support needs to be special in any way. One would hope that an ordinary audio cassette machine could be used unmodified. One quote from their glossy leaflet: "A second cassette is available through a built-in interface. This allows true file keeping and places the PET as a true and conventional electronic data processing system equivalent to just about any large commercial computer". That statement is, in my opinion, utterly misleading. The access times (for file maintenance) are vastly different.

The cassette **file handling** system has the unfortunate facility that file name matching can work on substrings. Thus if the command LOAD "CAT" were given the file handling routine would load in any of the following files "CATALOG", "CAT3" or "CATCH". The lack of a tape counter or of any means of fast-searching a tape — features found on other systems — are serious omissions. I was unable to establish whether binary files could be saved, read or appended and whether or not a "link" address could be associated with such a file. The cassette system is pretty much a bare minimum. It was also noticeably easy to "crash" the PET into a state from which recovery was only possible by removing the mains plug with resultant total loss of program and data. A "RESTART" switch is essential. Even if a system "crashes" it is often possible to recover much, or all, of your program if an ordered RESTART is provided.

It is fairly easy to form an impression of the merits and demerits of machines and their software, given adequate opportunity. What is far more difficult, but very very important for the prospective customer, is to weigh up companies' attitudes to their customers. The system that you own *will* expand. You *will* want new facilities as they become available, you *will* want to expand memory as its price decreases, you *will* want to run competitors' software and to write your own routines. You will have problems and need help from companies. To suggest otherwise is naive. When purchasing your first substantial chunk of computer you will want to consider companies' previous records. Have they been prepared to help with your difficulties? Have they offered software freely at reasonable prices with good documentation and listings of vital parts such as the input-output routines?

These are questions worth asking and worth getting answers to.

LP ENTERPRISES

From the representatives in Europe ... for America's leading Micro-computer magazines and books, for the hobbyist, educationalist and professional alike, we bring you a little light Browsing

Reading maketh a full man ... Francis Bacon (1561-1626)

** TICK HERE OR INDICATE QUANTITY ORDERED

From Adam Osborne Associates

INTRODUCTION TO MICROCOMPUTERS
Volume 0: The Beginners Book	£5.95
Volume 1: Basic Concepts	£5.95
Volume 2: Some Real Products (Revised Late 1977)	£11.95
6800 Programming for Logic Design	£5.95
8080 Programming for Logic Design	£5.95
Z80 Programming for Logic Design (Available May 78 approx)	£5.95
8080A/8085 Assembly Language Programming	£6.95
Some Common BASIC Programs	£5.95

BUSINESS PROGRAMS IN BASIC
Payroll With Cost Accounting	£9.95
Accounts Payable & Accounts Receivable (Available from June 78)	£9.95
General Ledger (Available June 78)	£9.95

From Scelbi Computer Consulting Inc.
6800 Software Gourmet Guide & Cookbook	£7.95
8080 Software Gourmet Guide & Cookbook (updated 78)	£7.95
8080 Programmers Pocket Guide	£2.25
8080 Hex Code Card	£2.25
8080 Octal Code Card	£2.25
8080 Guide and One 8080 Code Card	£4.20
8080 Guide and Both Code Cards	£6.00
SCELBAL High Level Language for '8008/8080' Systems	£39.25
SCELBAL String Handling Supplement	£8.00
SCELBAL Extended Maths Supplement	£4.00
Understanding Microcomputers & Small Computer Systems	£7.95
SCELBI 'BYTE' Primer	£9.95
8080 Standard Assembler (In Book Format)	£15.95
8080 Standard Editor (In Book Format)	£9.95

From Peoples Computer Company
Reference Book of Personal & Home Computing	£4.95
What to Do After You Hit Return	£7.00
Dr. Dobbs Journal Volume 1	£10.00

*From Kilobaud/73 Magazine Inc.
Hobby Computers Are Here	£3.95
New Hobby Computers	£3.95

From Dymax Inc.
Instant BASIC by Jerald R. Brown	£4.95
Your Home Computer by James White	£4.95
My Computer Likes Me . . . When I Speak BASIC By Bob Albrecht	£1.65
Games With A Pocket Calculator by Thiagarajan & Stilovitch	£1.75
Games, Tricks and Puzzles For a Hand Calculator by W. Judd	£2.49

*From BYTE Publications Inc.
Paperbytes:
Tiny Assembler for 6800 Systems	£5.75
Bar Code Loader for 6800, 8080, Z80 & 6502 Micros	£1.75
Best of BYTE Volume 1	£8.95

** TICK HERE OR INDICATE QUANTITY ORDERED

	Price UK	Price Overseas If Different
*From Creative Computing Press		
Best of Creative Computing Volume 1	£ 6.95	
Best of Creative Computing Volume 2	£ 6.95	
101 BASIC Games (Revised & Reprinted Feb. 78)	£ 5.50	
The Colossal Computer Cartoon Book	£ 3.95	
Computer-Rage (A new Board Game)	£ 6.95	
Artist and Computer	£ 3.95	
Three Binary Dice	£ 1.00	
*From Everyone Else		
TV Typewriter Cookbook by Don Lancaster	£ 7.95	
Magazine storage boxes (Holds 12 minimum)	£ 1.75	
Sybex: Microprocessors	£ 7.95	
Sybex: Microprocessor Interfacing Techniques	£ 7.95	
Dilithium: Home Computers Volume 1: Hardware	£ 6.50	
Dilithium: Home Computers Volume 2: Software	£ 5.95	
MAGAZINES: Back Issues		
Personal Computing	£ 1.75	
Interface Age	£ 2.00	
Dr. Dobbs Journal	£ 1.60	
Computer Music Journal	£ 2.50	
Peoples Computers	£ 1.50	
BYTE	£ 2.25	
Creative Computing	£ 1.75	
Calculators & Computers	£ 1.75	
ROM	£ 1.75	
Kilobaud	£ 2.25	
73	£ 2.00	

MAGAZINES: Subscriptions
		Price UK	Price Overseas If Different
Personal Computing	(Twelve Issues Yearly)	£16.00	£17.00
Interface Age	(Twelve Issues Yearly)	£20.00	£20.50
Dr. Dobbs Journal	(Ten Issues Yearly)	£13.00	£13.50
Computer Music Journal	(Four Issues Yearly)	£ 8.50	£ 9.00
Peoples Computers	(Six Issues Yearly)	£ 8.00	£ 8.50
Kilobaud	(Twelve Issues Yearly)	£20.00	£21.00
*BYTE	(Twelve Issues Yearly)	£15.00	£15.00
Creative Computing	(Six Issues Yearly)	£ 8.50	£ 9.00
Calculators & Computers	(Seven Issues Yearly)	£10.00	£10.50
ROM	(Twelve Issues Yearly)	£16.00	£17.00
73	(Twelve Issues Yearly)	£20.00	£21.00

PCW2

HOW TO ORDER

Please note our prices include postage and packing, but not insurance, if wanted add 12p for every £10 of books ordered. Make cheques, PO's etc payable to:-

L.P. Enterprises, Dept. PB
CREDIT CARDS accepted:
BARCLAYCARD/VISA/
ACCESS/DINERS CLUB/
AMERICAN EXPRESS
Phone 01-553 1001 for all credit card orders (24 hour service)

Send to address above for the attn. of Peter, Dept PB
Indicate Payment Method:

All Orders must be Prepaid

Total Enclosed £.........

.........My cheque, P.O., I.M.O. is enclosed in **Sterling** on U.K. Bank
.........Charge to Barclaycard/Visa/Diners/American Express/Access

Credit Card number . . . . Expiry Date . . . .

Name . . . .

Address . . . .

. . . . POSTCODE . . . .

Card holders signature . . . .

All publications are published in U.S.A. and shipped air-freight by L.P. Enterprises. In unusual cases, processing may exceed 30 days.
*BYTE Subscriptions are processed in the U.S.A., and are air-freighted then posted from Amsterdam, allow 3 months for sub to start.

TRADE ENQUIRIES WELCOME
You may Photocopy this page if you wish to leave your PCW intact

PCW REVIEW

Mike Dennis

1
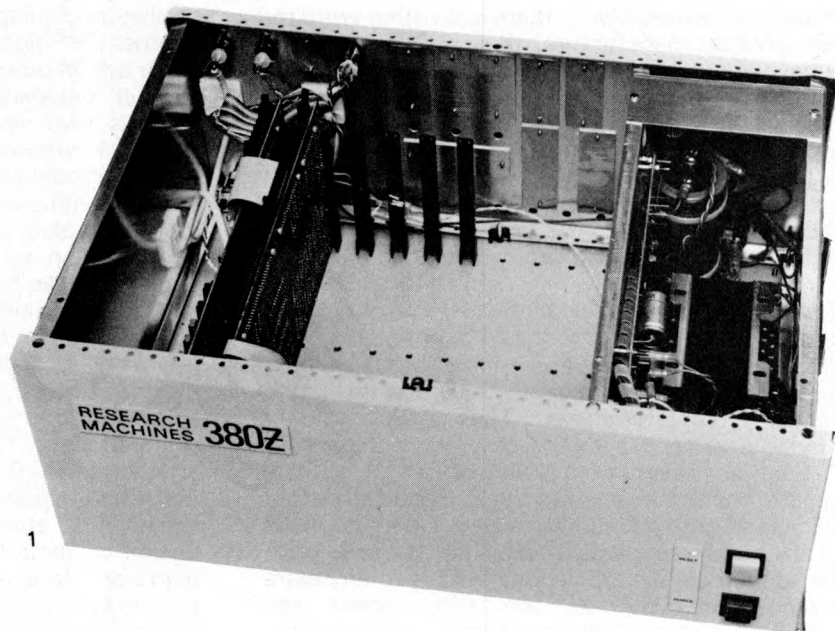
RESEARCH MACHINES 380Z

About the beginning of 1977, various advertisements appeared under the Sintel banner for the RML 380Z microprocessor systems. In fact, it was not until August 1977 that the first prototype machines went out and December 1977 that the first production models were despatched. The prices ranged from about £400 to £1400 depending on facilities and memory size and it was the grand-daddy of the range with 32K of RAM that I was given to review.

Originally I was to have it for only a week but I must have smiled nicely for Research Machines (RML) were quite happy for me to have it for a much longer period. In fact, I wholeheartedly endorse the editor's comments about the interest and assistance offered by this firm. Such attributes are all too often non-existent in larger organisations.

Appearance can be deceptive!

Surprisingly, the 380Z is much lighter than its' size suggests. It weighs only about 25lbs. in a cabinet the size of a small suitcase. No wonder, really, as when you open up the case all that you see are two PCB's, a power supply and a lot of air! (See photograph 1).

The 380Z was designed to be modular in construction, which is good. What this means is that fault finding is greatly assisted and spare boards or power supplies can be swiftly changed and thus reduce your downtime. What I didn't like was the use of Lucar connectors on some wires which were far too thin to carry the relatively heavy weight of the connectors. There are many other more suitable connectors on the market and so to use this particular type is sloppy. The general standard of construction inside was, I felt, more adequate than elegant. For instance, odd pieces of perspex sheet were stuck together in layers and then stuck to the inside of some of the panels presumably to stop them from rattling. The actual fixing flanges in the internal panels distort permanently with use as the screws are tightened up. You can see this in the photograph. Generally, in the professional equipment one doesn't normally find this. Surprising, too, was the lack of

Serial Number or other means of individual identification.

The power supply has been mounted well out of the way to give plenty of room in the case. More space has apparently been saved in this area by using a transformer designed by RML. All mains connectors were suitably protected against 'fairy fingers'. The mains input itself is via the detachable new IEC connector and comes ready to go with a mains lead of decent length and 13A plug. I must take RML to task, though, for not fitting a fuse of lower rating in the plug. I would have thought that one at 3A was more than ample and not 13A! (Dear reader, are all *your* plugs correctly and, therefore, safely fused?)

Beautiful PCB's

Another grey mark is the lack of protection for the two PCB's. They rest on the bottom of the case directly and although they are free to slide up and down, they are still vulnerable to shock or impact through the bottom panel. Perhaps if RML agree they could fix some form of foam plastic to the bottom of the case to rest the boards on? The PCB's

themselves are *beautiful* — there is no other word for it. Of German manufacture, they are double sided and have about 1400 plated through holes *each*. The pair that I had were slightly spoilt by solder splashes which can cause intermittent faults. RML say that they hadn't been cleaned properly and certainly other boards that I saw were much better. In any case, I am really nit-picking as since the boards are mounted vertically then any solder splashes that fall off shouldn't have any catastrophic effect.

The physical size of the boards is 10″ x 5½″ and that's the nearest that RML get to the S100 bus — the same width! Provision has also been made for mounting a shorter six inch board. I think their bus technique *particularly cost effective.* They make use of connectors similar in style to Scotchflex but made by ITT Cannon. Essentially, they consist of a male (female?) half mounted onto the PCB and the other half crimped onto a flat ribbon cable. Extension leads with any number of plugs/sockets can be made up in order to connect up the various busses to any extra boards. Mind you, as the CPU board can accomodate *up to 32K* it would be some time, probably, before you needed any extra boards for memory! It was a pity that no indexing was provided as in theory it is possible to insert the plugs either way round. In practice, the physical layout and wiring prevents you from doing this. You can do it, though, with the dc supplies which is potentially disastrous. You could end up with a very expensive mistake on your hands. Take note!

Connections to the outside world are by a seven pin DIN socket and two sockets for a visual display. RML supply one DIN lead and one co-ax lead for connecting up. Provision has already been made both in the wiring, hardware and software to couple up two cassette machines via a suitable splitting box to the DIN socket (see later). I was not supplied with any of this but have seen a working system. Clearly, an indication that RML are continuing to expand the system, which is to their credit.

Signals
Although both sockets for the VDU were Belling Lee type co-ax, only one socket is fed by the UHF modulator. In practice, there was sufficient stray oscillation coming from the second socket to provide another UHF signal for a second TV. However, under these conditions it is possible that some TV's could interact between themselves to produce inferior pictures. RML never intended it to be operated in this way but nevertheless you *could* get a small bonus.

I tried feeding the UHF signal into a colour set and had great difficulty in tuning in the signal. Now most, if not all, colour sets have an AFC circuit which you disable when you tune in a station. When a similar technique was tried with the 380Z, as soon as the AFC was enabled again, it pulled the set off tune and so produced a bad picture. But if the AFC was permanently disabled by going inside the set, the problem then ceased as one would expect. On investigating further, the modulator was giving out a signal of about 10mV! More than enough to swamp any normal AFC circuit and so cause the de-tuning that I had experienced. The simple cure is either to use an aerial attenuator (available from any good TV dealer) or use a black and white set. However, this high signal level is rather naughty as the frequency of the modulator was about 592MHz or Channel 36 and so outside the broadcast band. There are other users around this frequency and so at signal levels of this amplitude it is conceivable that interference could be caused by the 380Z to any authorised users of this band.

The second socket is for a line-fed monitor and as such should have been fitted with the proper socket for the job i.e. a BNC (readily available from RadioSpares). Belling Lee co-ax sockets are definitely *not* professional for this type of work and in any case necessitate a convertor lead for use with any monitor. Murphy's Law applies when looking for this lead. Again, the video output level is a non-standard one of 0.15v pk-pk of syncs and 0.4v pk-pk of video. (If your system measures differently then this is due to production tolerances). I measured these when the signal was terminated (as it should be) in 75Ω. Leaving the signal unterminated increased the amplitude but made the characters look a bit ragged, which was expected. The definition in the monitor was, of course, much better than that obtained from the UHF feed. This is generally the case and I wish that better modulators were readily available.

2
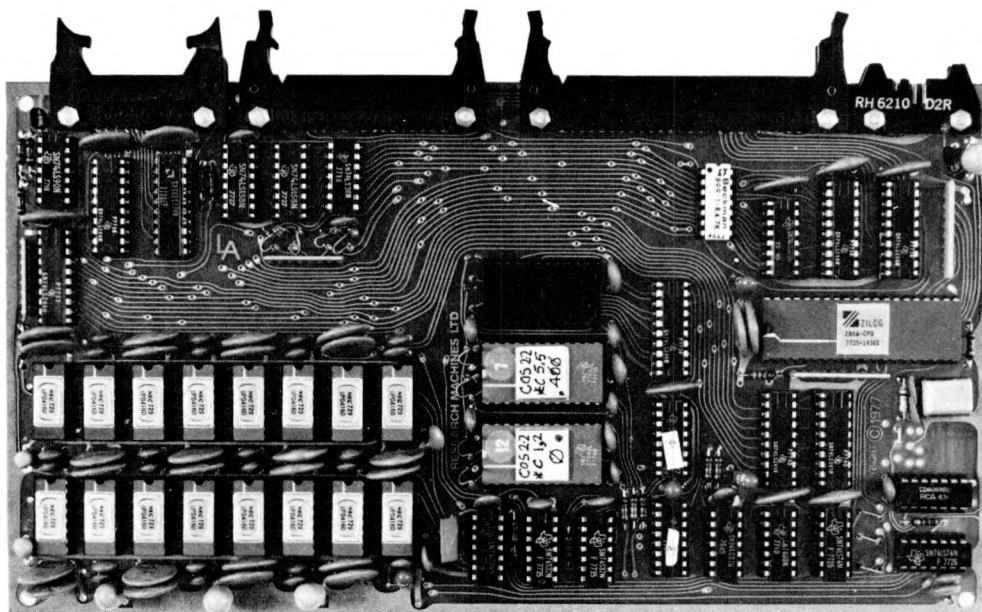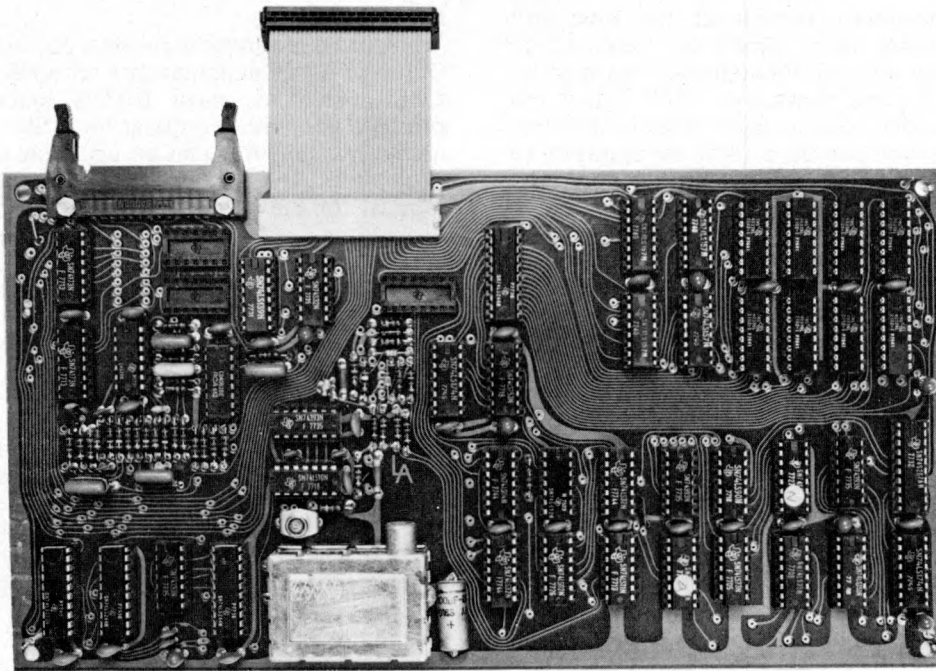
3

Hardware

Talking of volts brings us conveniently to a brief look at the actual hardware used, as I expect that some of you may be particularly interested in this aspect. As can be seen from the photographs 2 and 3, there are remarkably few IC's for so versatile a machine. There are approximately 45 ic's per board together with their respective sockets. I could write an entire article on the various and neat design aspects of the 380Z but suffice to mention just a few points.

1) 16k x 1 350nS dynamic RAM's were provided. They are NEC/Nippon devices and similar to the Mostek 4116 etc. As such, 4k x 1 dynamics are a direct replacement fit and provision has been made for this.

2) Thick film resistors have been used where any pull-up and pull-down resistors are required.

3) An 8MHz crystal is used for the system clock and divided down to provide a 4MHz clock of equal mark-space ratio.

4) One Wait state is put in all Memory Request (MREQ) cycles at the factory. Should you wish to run at a slower rate then the user can re-arrange on-board wire links to insert a Wait state in each M1 cycle.

5) Much use has been made of LSI devices such as the 74LS393 (two 7493's in one package). There is a liberal sprinkling of 74LS157, 158, 241, 244 and 374's. The majority of these are comparatively new devices to the TTL family and many have been designed with the microprocessor in mind. If you are at all interested in designing your own system then I can only urge you to buy an up-to-date TTL Data Book from one of the distributors. They make fascinating bed-time reading!

6) Liberal decoupling means *gbod* noise performance. RML have spread tantalum capacitors and 0.1uF's around like confetti and the result has been very little noise on any of the dc lines. Most commendable. Even the 3A regulator which is situated away from the CPU board by a very long wire seemed perfectly able to cope. In fact the 380Z only uses about 1¼ of the available 3Amps which again allows ample expansion.

7) The COS (Cassette Operating System) lives in two 2708 ROM's and there is space for an optional ROM be it user programmed or as, in my case, supplied by RML (more anon).

8) I spied a couple of other ROM's on both boards. On the CPU board were both a 74S287 (256 x 4 bits) and a 74S288 (32 x 8 bit) ROM. The 287 is used to decode which block of RAM is being addressed and there are four versions available depending on what type of memory you have (eg. 2 x 16K, 2 x 4K etc.). Should you win the pools or strike lucky at the finance meeting and wish to install more RAM, then all you do is swap the 287 and plug in another RAM. Slight alterations to one pin have to be made depending on whether you are using 4K or 16K RAM's and these are done on wire-wrap connections. Should you have 8K of RAM and wish to update to a 16K version then rather than supply a separate board as originally advertised, RML will buy back your 8K of RAM and sell you a 16K block. Any extra cost doing it this way will be absorbed by them.

One output of this ROM connects up to the 'S288 as an enabling signal and indicates that the lower 8K bytes of memory are being addressed ie up to 1FFF. This and address lines A9-A12 inclusive are then sufficient to decode the three ROM positions, the ROM data buffer, VDU memory select signal and 'I/O' port select signal. This latter when NANDed with A2-A8 inc. provides a low when locations OFFC-OFFF are being addressed. However, I/O is really a misnomer as the IORQ signal from the CPU is never used as these particular I/O ports are memory mapped. OFFC is either the keyboard in the read mode or Port O in the write mode. The pattern of bits written into this port control various functions in the monitor. OFFD is a counter used for the software single step and reset. OFFE is Port 1 and dedicated to the cassette/program storage facility etc. and other misc. functions. OFFF is a user port and has even been provided with a buffered octal latch.

So with only two ic's, RML have decoded all of this whilst still maintaining a high degree of flexibility. A really good example, I think, of what a fresh approach can do in a situation like this. That really sums up the CPU board.

The VDU board contains both VDU and enough interface for two cassette recorders. Again, ingenious use had been made of two ROM's ('S287s again). These, in conjunction with their respective

line and field counters, provide all the sync and blanking waveforms, reset signals etc. required by the VDU together with *all* the addresses required by the display RAM's and charactrer ROM. This is the first time I have seen such a clever design. The sync width is a little non-standard (4uS as opposed to 4.7uS) but in practice this should make very little difference.

Any calls to output an alphanumeric character to the screen occurs in Field Blanking. In order to use graphics in an interactive mode a faster rate is required. One sub-routine in COS uses line blanking (24uS) to output one character per line. Now, 24uS is not long, even with a 4MHz clock so if you wait for line blanking to arrive before beginning your o/p routine then you soon run out of time. However, since this output routine consists of a known length of machine cycles and hence time, you can anticipate the lineblanking signal and use a pulse of earlier timing. Because RML have done their sync generation the way they have, the line blanking ROM quite conveniently has just such a waveform with the right timing!

One design point that I personally dislike is the software conversion of capital O to □. This is necessary because the Texas 262 character generator used doesn't discriminate between capital O and zero. I personally prefer O and ø.

Note from the photograph (2) the conspicuous lack of any UART's (a UART is a large 40 pin ic that can convert 8 bit parallel data into serial data and vice versa). This is because about 256 bytes of COS has been used to record and replay a cassette tape to CUTS standard. At 2708 prices this costs very approximately £2.50 whereas two UARTs could cost together about £10 — clearly a good example of an economical software/hardware tradeoff. However, this is only the half of it as with the option ROM supplied there is a fast cassette option of 1200 Baud. Bear in mind that their cassette file handling is also *very sophisticated.*

Recording

To record a programme on tape you either type D for Dump or SAVE depending on whether you are in the COS system or have BASIC loaded in. Either prompts you with a request for a file or programme name. In COS, this can be up to six letters long; in Basic, ten. The system then prompts you with a request for the first and last address of the data or programme and also the start address. The programme is then parcelled up into either 128 or 255 (yes, 255 as in order to save bytes in the monitor, RML use the zero count for end of data block) byte blocks. Each block has the programme name attached to it.

Loading

To load a programme from cassette, select your recording speed (300 or 1200) if necessary, enter the program name and then replay the cassette. The COS system reads in all the data blocks and echoes the name at the beginning of each block on to the VDU screen. When the name coming off the tape matches the programme selected by you, the monitor loads the programme at the correct address. Since each block is named, either you or the 380Z can search through an entire tape for a specific programme. All that you need to do after entering the programme name is to fastwind through the tape, periodically stopping and replaying a short section of tape to see in which programme you are in. Likewise, by using cassette recorders (such as the Hitachi loaned by RML) with a remote pause means that the 380Z can load a block of data into its Assembler (ready soon!), then pause the cassette while it digests the data. Of course, this latter facility is only provided by their Cassette File Handling Programme which is supplied free with the more expensive versions. This programme can also be used, so RML say, for loading tapes recorded to other standards such as the INTEL format. An interesting side effect of the basic COS system is that

```
→PC  4400  32  D3  4B  B7  28  58  C5  CD
 SP  40FE  FE  40  00  44  86  03  C3  B1
 IY  FFFF  FF  FF  FF  FF  FF  C3  3B  03
 IX  4100  00  44  86  03  C3  B1  41  C3
 HL  0386  D3  00  F7  16  FE  4C  28  3F
 DE  0005  3B  03  C3  70  03  54  4F  C3
 BC  0000  FF  FF  FF  FF  C3  3B  03  C3
 AF  1A4A   Z   N       ↑

 43F4  06      43FC  32      4404  64      440C  38
 43F5  CD      43FD  D3      4405  46      440D  49
 43F6  91      43FE  4B      4406  7D      440E  AF
 43F7  49      43FF  B7      4407  FE      440F  CB
 43F8  C3     →4400  28←     4408  50      4410  3D
 43F9  77      4401  58      4409  30      4411  30
 43FA  43      4402  C5      440A  4C      4412  01
 43FB  78      4403  CD      440B  07      4413  3C

 !
 !M>4400
 !■
```

programmes recorded in this fashion have a certain (I admit, not fantastically great) security against easy pirating.

I used the cassette system quite extensively and, at first, had one or two failures. This prompted a visit from RML with various spare bits but we were unable either to reproduce the fault or to explain why they had occurred in the first place. Certainly, some failures can be put down to my lack of familiarity with the cassette machine. This is one of the few cassette machines that has both volume and tone controls on the output. Most cassette machines have neither but, in this case, it was necessary to adjust them correctly. One annoying feature of this cassette player was that connecting the DIN lead mutes the internal speaker. Personally, I like to have the volume up a bit so that I can hear what's going on and so if this too is your preference then this is not the cassette player for you.

A more annoying feature was the automatic return after the sixth letter in the name when loading in COS. If you make a typing mistake in the sixth letter, then you have to either *Reset* the 380Z or ignore the mistake, replay some tape and then hit Control C. (This is because you cannot return to COS if the tape isn't running. This is one drawback of using a software cassette interface in this way.) With either you have to go through the whole loading procedure again.

Mind you, hitting the Reset isn't as disastrous as it could have been had RML not had the forethought to do it carefully and keep the refresh going for the dynamic RAM's during reset! However, you've got to remember to reset your fast tape option, if necessary, after reset, otherwise you'll enter your name, start the machine and then have to start all over again! This is because COS 'fails' to normal tape speed after a reset. A better way would be a 'fail' to 'fast mode in' as most programmes will be stored on tape in this way.

One nice feature of the COS loading is that should you get a read error then the machine won't abort completely, unlike some. You have the option of either aborting deliberately or of rewinding the cassette a bit, typing 'L' and the programme will be reloaded at the start of the next block, and hopefully go in OK the second time round. I noticed that you couldn't do this when Saving a Basic tape.

I tried playing back pre-recorded cassettes made on various cassette machines and had no compatibility problems. I measured the "dynamic range" over which the COS would load and save tapes satisfactorily and found the system was still coping at a level of 20dB below peak recording level. What this means is that you should have no need to worry unduly about what level to record at when saving programmes. Obviously, when using the high speed option, tolerances regarding drop-out etc. on the tape are that much tighter and RML quite sensibly recommend the use of high quality tapes.

One possible problem with some cassette machines is that the output is 180° out of phase with the input. RML have thought of this and provide a pre-recorded programme called "TSTCS" or Test Cassette. This enables a check to be made both for optimum replay level (assuming you have control) and for phase. This is all fairly well documented in the literature and any out of phase machines can easily be cured by using a 1:1 transformer suitably wired.

Software

Any system, no matter how good the hardware, must stand by its firmware and software and it is here that the 380Z scores. In particular, the 'software front panel' is one of the clearest and most versatile that I have met (see photograph 4.). Briefly, it displays in the upper unscrolled 20 lines of the screen, the contents of all the main registers, stack pointer and programme counter. Those registers that can be paired up are and the resultant 16 bit address

thus derived is displayed alongside. The contents of the memory location thus addressed by each pair is *also* displayed together with the contents of several locations on either side, the actual location being pointed to by the arrows. Personally, I would have preferred the display of these locations to stand out from the rest of the display (bright-up?) as it was often difficult to remember where each one was. The contents of the registers and programme counter can be altered at will.

The data display, of course, will alter as these registers are changed and so the effect on the system of any changes can be instantly seen. The stack pointer cannot which could possibly prove annoying at a later date. Neither the I nor the R registers are shown. It is possible to switch to the 'other' registers. Another good point is the ability to single step through a programme. Pressing 'Z' causes an NMI into the single step routine. The programme counter, stack pointer and registers all change before your very eyes and one can really appreciate the effects of the various instruction code, stack pointer manoeuvres etc. *I can't recommend this feature strongly enough for anyone involved in teaching micro's.* Go and see for yourself. A nice touch is the actual printing of the various flags when set e.g. Z and C if both the zero and carry flags are set.

All this display is contained in ten lines. The remaining ten lines are taken up by a separate memory pointer. The address of this can be entered by the user following an 'M' command. The pointer will alter and 32 bytes of data on either side of this new pointer will be displayed. The pointer can also be incremented or decremented by either 1 or 8 bytes. Data can be written into memory directly and this is how a hex programme would be keyed in. The bottom four lines of the screen echo keystrokes and scroll.

Another nice feature is the 'Memory Shift' command. This works out the best way to move the memory blocks so as not to corrupt the data should you wish to move one block to an area that, in fact, overlaps part of the original block. Many other monitors eg. Zapple will corrupt the data under certain conditions.

One or two minor bugs were: —

1) any non-hex character acts as a terminator and in one case caused the system to lock-up necessitating a 'Reset'. This means that the monitor is not very forgiving and that if you make a typing mistake, COS will not let you correct it! One man's meat . . . though, as RML say that some people prefer a system this way because you *don't* have to always hit CR to terminate. You pays your money . . .

2) Confusion due to choice of keys for certain commands eg 'Z' in 'frontal panel' single steps (a Go type signal) but 'Control Z' in Basic halts the programme!

3) Good though the single step facility is, it would soon get pretty tedious stepping through a loop with a count of 255. RML have thought about this and included the provision for setting break points. What can happen is that the breakpoint is never reached. The keyboard is then locked out and so the only resort is 'RESET'. Unfortunately, as this wipes the registers, tracing the bug can prove difficult. What is needed is some form of "time-out" facility to return to the monitor. Not many monitors do this and unfortunately COS is no exception.

COS resides at 0000-07FF (see memory map). Beware, there be dragons, in the form of the restart vectors. RML have used these to great effect. Four are jump outs into RAM for user control. One is the normal reset to 0000, one is used for break points and another for simulating RELATIVE CALLS. (see later). By far the most ingenious, however, is F7 which is a Trap Call and is a two byte instruction equivalent to a 'CALL'. The second byte following F7 contains the trap code of the desired target sub-routine. If all the commonly used sub-routines eg. Get keyboard character etc. are routed through as 'transfer vectors' using this Trap Call, then the number of bytes saved by not having to use a 3 byte Call every time will more than offset the bytes needed for the Trap Call routine. As Trap Call only uses a few vectors, you the user have well over a hundred left! Space does not allow me to do justice to, what to my mind, is one of the carefully thought out and refreshing (no pun) monitors that I have seen for a long time. Drawbacks? Well, often programmes are designed to run at this address so you'll have some relocating to do. The other drawback is slightly greater execution time.

The relative call simulation is used to great effect in the PIC or Position Independent Code. The Y index pointer is used in conjunction with this to enable any programme written with PIC to run elsewhere in the system. The range is limited to +/— 128 bytes but by writing in extension routines within this range it is possible to extend the range to anywhere. Many RML programmes are written in this way and when loaded are automatically loaded at the top end of the available RAM (stored as HIMEM in COS). HIMEM is then altered to its new lower value and if desired another programme could be loaded ad infinitum (well almost). Each programme then uses PIC to find out where its been put!!

Yet another facility is the Page mode should you wish to address >48K of memory. Of course, you'll need a Memory Extension Board plus more RAM/ROM but that's a small price to pay and of little consequence. The main point is that RML have provided for you.

One cannot end this exhaustive (and exhausting) review without mentioning their 8K Basic which is good, very good. There are many extra facilities, in particular the Line Editor, which one would normally find in Extended Basics of greater length. This Basic is also very fast — look at the Table in John Coll's article in Vol 1 No. 1 of *PCW*. It isn't the fastest but its times are still very respectable. How do they do it? Well, for a start all the Basic command words like 'Print', 'If' are encoded as each line is entered into a one byte code with bit 7 set. This means that you can save on memory space, and speeds up programme execution time as each programme is part compiled. They are continuing to refine their Basic and, in fact, are up to Basic which has graphics to: —

1) **Plot x,y,z   where x = 0-79   else error**
   y = 0-47   ,,   ,,
   z = 0 = dot off
   1 = dot on dim
   2 = dot on bright
2) **Graph 0 = Normal scroll**
   ,,   1 = Clear top 20 lines, scroll bottom 4.
   ,,   2 = No clear, 4 line scroll.

Another facility is to execute 'Control S' before executing the programme or listing command as this stops the scroller. Each 'Control A' then scrolls one page and then stops the scroll and programme. 'Control Q' restores to normal operation. It is very easy in this mode to wonder why the programme has stopped (you've forgotten to press CTRL A) and so RML give you a clue by switching the cursor off.

Personally, I think it would be better to blink it on and off. Another minor irritation is the carriage return automatically on character 40 and not 41 as one expects.

During a visit to RML, I was shown their new Assembler and Text Editor in a working system with twin cassette machine floppy disc and TTY. The software also caters for paper tape. Again space and time did not permit a thorough examination but it looks as if it is very powerful, very flexible and very good.

The summing up

Obviously, many of my criticisms were purely personal and should be taken as such. Technically, the product was excellent (the boards really are a work of art!) and there were many innovative and ingenious design features. Much though has gone into the design to make it as flexible and easy to interface to as possible. The styling is a bit functional and the general construction could be tidied up. The monitor ROM and software backup are *excellent* and

continually being updated and improved. Existing customers can get either free or at cost the fast scroller, fast tape I/O, Intel loader and TTY routines from RML. I just have the slightest unease about the lack of consistency between some of the characters used for commands and terminators. There is a hint of illogicality about some of them — probably as a result of this 'fine-tuning' to the system. And don't forget, you don't need to take out a second mortgage. You can buy a 280Z version which consists of CPU board and VDU board fully built and tested (soak tested for two weeks as are all their machines) for £400. If I was on a tight budget, then this is the version that I would buy. Otherwise any businesses, firms, institutions, schools etc considering buying a good all-rounder could do a lot worse than take a long hard look at any of the RML versions. And don't forget, you aren't necessarily tied to buying any extra memory from Research Machines.

To conclude, a few quirks but in general an excellent machine. I was sorry to see it go.

MEMORY MAP

Address		Notes
FFFF		
Cooo	User RAM	48k bytes of user RAM possible without
8000		paging
4000		
3000		
2000	Assigned	May be used for ROM etc.
1000 / 0E00	Control ports	
0800	VDU	
	COS Monitor	
0000		

Address		Notes
4100	Start of user RAM	
40FF	Monitor stack	
	I/O vectors	256 bytes
4000	Monitor variables	
3FFF		
	Unassigned	
2000		
1FFF		
	Can be used for	
1000	ROM	
0FFF / 0E00	Control ports, KBD, User I/O	512 bytes
0DFF	Memory mapped VDU	1½K bytes but physically only 1K
0800		
07FF		
	COS monitor	2K bytes
0000		

FOUR EASY PIECES

Neil Harrison

Thou art false but thou art beautiful . . .

Pseudorandom Number Generator

A sequence of random numbers for use in micro-processor games and simulations is usually generated by software. For application where memory is limited or software generation of random numbers is too slow the following circuit can solve the problem. Strictly speaking the circuit produces a pseudo-random (p.r.) sequence of numbers repeating after 255 shifts. The shift registers (IC1 and IC2) can either be clocked every time the microprocessor reads a random byte or allowed to 'free run' on the system clock. Presettable shift registers must be used since a p.r. sequence will not be generated if the shift registers contain all zeros. A logic 1 on the RESET line for a minimum of one clock period will load non-zero parallel data into the registers to avoid this problem.

Longer sequences can be generated by increasing the number of shift registers and altering the feedback connections. Table 1 gives the feedback connections and sequence lengths for p.r. sequence generators using up to 16 shift registers.

Shift register stages	feedback bits	sequence length
8	2,3,4,8	255
9	5,9	511
10	7,10	1023
11	9,11	2047
12	2,10,11,12	4095
13	1,11,12,13	8191
14	2,12,13,14	16383
15	14,15	32767
16	11,13,14,16	65535

Table 1.

Multiplication's the Name of the Game
8 by 8 bit Multiply Routine

This little 12 byte subroutine for the Z80 will multiply two unsigned 8 bit numbers together to give a 16 bit result. The subroutine expects to find the multiplier in the H register and the multiplicand in the E register. The result is in the HL register pair.
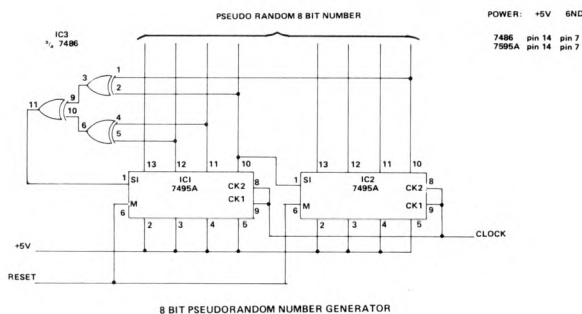
```
0000          ;       Z80 UNSIGNED 8 BY 8 BIT MULTIPLY
0000          ;
0000          ;       ENTER WITH THE TWO 8 BIT NUMBERS
0000          ;       IN THE H AND E REGISTERS. THE 16
0000          ;       BIT RESULT IS IN THE HL REGISTER
0000          ;       PAIR.
0000          ;
0000          =       100
0100 06 08    MPY:  LD    B,8      ;LOOP COUNTER
0102 16 00          LD    D,0      ;D REG = 0
0104 6A             LD    L,D      ;L REG = 0
0105 29       MPY1: ADD   HL,HL    ;SHIFT HL LEFT
0106 30 01          JR    NC,MPY2  ;SKIP IF C FLAG=0
0108 19             ADD   HL,DE    ;ADD IN MULTIPLICAND
0109 10 FA    MPY2: DJNZ  MPY1     ;LOOP UNTIL DONE
010B C9             RET            ;RETURN TO CALLER
```


8 BIT PSEUDORANDOM NUMBER GENERATOR

Scrooge's Delight
Economy Potentiometer input for your MPU

Whenever your MPU needs to know the position of a potentiometer the usual approach is to use an expensive Analogue to Digital converter to measure some related voltage. For applications such as games where stability and accuracy are not too critical the following circuit and program can provide a very cheap alternative.

The circuit, fig 1, uses the good old 555 timer in the astable mode with the potentiometer R_1 controlling the output pulse width. The output waveform is shown in fig 2 and the times t_1 and t_2 are related to the component values by the following: —

$$t_1 = 0.685 . R_3 . C_1$$
$$t_2 = 0.685 . R_3 . C_1 + 0.685 . R_1 . C_1$$

The value of potentiometer R_1 and therefore its position is proportional to $(t_2 - t_1)$. Resistor R_2 is to protect the 555 and has such a relatively low value that it need not be included in the timing calculations.

The output of the 555 is monitored by a single bit on an input port on a Z80, on my system bit 6 of port 4 was used.
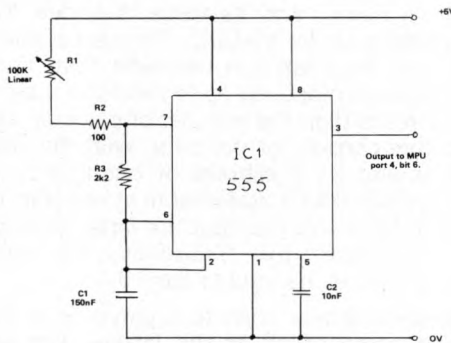
fig. 1
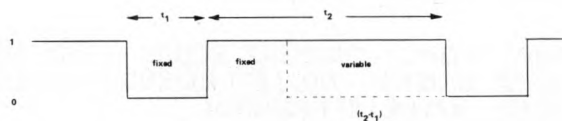
fig. 2 555 Output Waveform

Listing 1 shows a Z80 code subroutine which returns a value in the HL register pair, corresponding to the time (t_2-t_1) and therefore the position of the potentiometer. Time t_1 is subtracted from t_2 by starting with HL=0 and counting down during t_1 and up during t_2. Before this can happen the subroutine must make sure that counting starts at the beginning of t_1. Since the subroutine finishes its counting at the end of t_2 we can use this to indicate the start of a new t_1. To do this we use the trick of calling the counting loops starting at POT1 as a subroutine. At the end of the next t_2 the HL registers are set to zero and the two loops starting at POT1 and POT2 perform the (t_2-t_1) calculation. When the end of the next t_2 is reached the routine returns to the main program with the value in HL corresponding to the position of the pot.

For the component values shown in fig 1 and a Z80 running at 2MHz the range of values returned in HL is 3 to 500 decimal. This can of course be increased or decreased by altering the value of C_1. The worst case calculation time taken by the subroutine is 2 x $(t_1+t_2$ max).

```
0000          ;       POTENTIOMETER INPUT USING A 555 TIMER
0000          ;
0000          ;       THE OUTPUT OF THE 555 IS CONNECTED TO BIT
0000          ;       6 OF INPUT PORT 4.
0000          ;
0000          ;       THE ROUTINE RETURNS WITH A NUMBER IN HL
0000          ;       PROPORTIONAL TO THE POSITION OF THE POT.
0000          ;
0000 CD 06 00 POT:   CALL  POT1     ;WAIT FOR THE END OF T2
0003 21 00 00        LD    HL,0000  ;INITIALISE THE COUNTER
0006 2B       POT1:  DEC   HL       ;REDUCE COUNT BY 1
0007 DB 04           IN    A,(4)    ;GET THE PORT INTO A
0009 CB 77           BIT   6,A      ;TEST THE BIT
000B 28 F9           JR    Z,POT1   ;IF 0 WE'RE STILL IN T1
000D 23       POT2:  INC   HL       ;INCREASE COUNT BY 1
000E DB 04           IN    A,(4)    ;GET THE PORT INTO A
0010 CB 77           BIT   6,A      ;TEST THE BIT
0012 20 F9           JR    NZ,POT2  ;IF 1 WE'RE STILL IN T2
0014 C9              RET            ;RETURN TO CALLER
```

BONUS TIME
Extra Z80 instructions

Many microprocessors can do things that the manufacturers either haven't bothered to mention or don't know about themselves. The Z80 is no exception to this and a number of previously undefined opcodes can be used to good effect.

Perhaps the most interesting extra is the ability to use the high and low bytes of the index registers IX and IY as if they are just ordinary 8 bit registers. The trick is to notice that opcodes for instructions involving the index registers are the same as for those using the HL register pair but preceded by DD or FD. For example loading the HL register pair with FFF0 is done by the three byte instruction 2100 F8. Loading the IX register with the same number takes the following four bytes DD 21 00 F8. This also applies to references to H and L as individual 8 bit registers so if,

6A = LD A,L ; Load the accumulator from register L

then,

DD 6A = LD A, IXL; Load the accumulator from the low byte of the IX register. References to the L register will give access to the low byte of the index register, the H register will give the high byte. This technique can be used for all Load, arithmetic and logical operations and probably lots more.

Not quite so useful is the extra shift instruction which uses opcodes CB30 to CB37. These codes give an arithmetic shift left similar to the SLA instruction. The difference is that bit 0 of the register is filled with a 1 instead of a 0. Perhaps someone can think of a use for this!

Thanks to **Mike Banahan** and **Mike Blandford** for details of these extra instructions.

SCRAMBLE!

Graham J. Trott

SCRAMBLE is a word game for two or more players. I conceived it after realising that there is an important element missing from games played against a computer — the competitive element, or the satisfaction of beating a human opponent. After all, a computer is a somewhat emotionless creature — it doesn't sulk or lose its temper — neither does it gloat when it's winning, and in any case there's always the reset button if it gets too smart, isn't there?

Once we decide to write a game for people to play against other people, the computer is just there to act as dice thrower and scorekepper; in fact SCRAMBLE can be played just as well with a pack of LEXICON cards and a score pad. For some reason, however, there's more appeal when the TV set is the game board; even children seem to have no difficulty in concentrating on the game for quite long periods. Besides, our computer isn't earning its keep while we sit around playing cards!

Since VDU's deal with alphanumerics, a word game is an obvious choice, and since I had ruled out most existing games, I had to invent my own. This turned out not to be too difficult. Word games are (to me at least) attractive since they depend less on blind luck and more on mental effort. Although the rules of SCRAMBLE are particularly simple, the program has to perform a certain amount of gymnastics with strings. I have drawn an outline flowchart, but the detailed operation of the functional blocks of the program are best understood by studying the listing. The particular features of SCRAMBLE that make it easy to implement are 1) the display is simple and 2) the only secrecy involved is in the players' intentions.

HOW THE GAME IS PLAYED

The program first asks for the name of the players. It will go on asking until a null return is given (just a carriage return). As the program stands, the arrays for the players' names, the letters they hold and their scores are not dimensioned; in most versions of BASIC this is OK but sets an upper limit of nine players. In practice, between three and five players gives the best game, and if your VDU only has a 16-line display then seven players is the upper limit. Dimensioning all arrays to six or seven elements will save storage space, as will restricting the number of elements in a string to 15 or so.

Once all the names have been entered, the main display is set up: —

PLAYER	LETTERS	VALUE	SCORE
TOM		Ø	Ø
DICK		Ø	Ø
HARRY		Ø	Ø
LETTERS	A N J * M R O K		
VALUES	1 1 8 Ø 3 1 1 5		

WHAT WILL YOU TAKE, TOM?

The player addressed may do one of three things. Firstly, he (she) may take one of the letter from the pool by typing it followed by carriage return (CR). The star is a joker and can be used as any letter, later on in the game. The chosen letter appears against the player's name, and its value is added to that already existing under VALUE. The next player then has his turn. As a letter is removed from the pool another takes its place, so as to maintain a constant number (5 more than the number of players). Letters are randomly chosen for the pool, with the value of each letter and its likelihood of occurrence being specified by the DATA statements at the *start* of the program. Thus if you feel that the letter D is under-valued and occurs too frequently, the numbers following it can be changed to (say) 3,3.

The second choice open to a player is to form a word with some or all of the letters that he has collected from the pool on previous turns. To do this just type the word, which must have two or more letters. It's not necessary to type the star; type instead the letter it represents. The program checks whether you do in fact possess the letters required for your word, then asks "ANY OBJECTIONS?". An answer beginning with "Y" will cause the repeat of the question "WHAT WILL YOU TAKE?", and any other response (including CR) will tell the program to accept the word. The letters in the word are removed from the player's stock, and his score is calculated according to the following formula: —

NEW SCORE = PRESENT SCORE + (NO. OF LETTERS USED — NO. LEFT BEHIND) * (VALUE USED — VALUE LEFT BEHIND)

with a minimum score of 1. It is obvious that there is a lot to be gained from having a long word, although only if you use all of your letters. The winner of the game is the first person to have 200 points, although the program only checks to see if someone has won after all players have had their turns.

The third action a player may take at his turn, as an alternative to either of the above, is to change the pool of letters, in the hope that a crucial letter will appear for his next turn (and not be snapped up by one of the other players). To do this just type CR. A new pool of letters will then be selected by the program, and play will pass to the next person. The number of times a player may change the pool is dependent upon the number of players, but varies from two to five, and is set by lines 1700-1750 of the program. The only way of getting rid of a letter once taken is to use it in a word, so it is unwise to take a Q unless one already holds a U.

THE FUNCTIONAL BLOCKS OF THE PROGRAM

100-130	The letters, with their frequency of occurrence and values.
140	Sets N to the total no. of letters (at present 83).
150-190	Control Characters — these will depend on the terminal used. The values in the listing are for a SWTPC CT-64. (Note: If the interpreter sends out a line erase character as part of its CR-LF string, this must be changed to a null. If this is not possible then delete line 2022 and remove the "A9-1" in line 2090.) The purpose of K2 is to delete the cursor while the display is being updated.

1000-1060	Gets the players' names and finds out who starts. Makes a note of who plays last (X9).
1700-1750	Decides how many changes each player is allowed.
2000-2090	Sets up the display and asks the first player for his choice.
3000-3150	Takes one letter from the pool and puts it against the player's name. Replaces it with another selected at random.
3200-3290	Changes the pool of letters for the next player.
4000-4290	Removes the player's word from his stock of letters. Asks for approval then calculates his new score.
5000-5520	Sets up for the next player and checks if there is a winner yet.
7000-7020	Subroutine to select a pool of letters. There are 5 more letters than players.

```
SCRAMBLE - WRITTEN FOR A 16K 6800 MACHINE, RUNNING SWTPC OR COMPUTERWARE 8K BASIC

0100 DATA A,6,1,B,2,3,C,2,3,D,4,2,E,7,1,F,2,4,G,3,2
0110 DATA H,2,4,I,5,1,J,1,6,K,1,5,L,4,1,M,2,3,N,5,1
0120 DATA O,5,1,P,2,3,Q,1,9,R,6,1,S,4,1,T,6,1,U,3,1
0130 DATA V,2,4,W,2,4,X,1,8,Y,2,4,Z,1,9,+,2,0
0140 FOR I=1TO27:READL$,N1,V:N=N+N1:NEXTI:RESTORE
0150 K1=25:REM HOME UP
0160 K2=1:REM CURSOR ON/OFF
0170 K3=21:REM ERASE LINE
0180 K4=22:REM ERASE FRAME
0190 K5=12:REM SCROLL/PAGE TOGGLE
0500 REM
0510 REM INTRODUCTION AND NAMING PLAYERS
0520 REM
0530 REM FIRST PUT VDU IN PAGE MODE, HOME CURSOR AND CLEAR SCREEN
0540 REM
1000 PRINT CHR$(K1),CHR$(K4),:P=1
1010 PRINT "SCRAMBLE - THE COMPUTER WORD GAME":PRINT
1020 PRINT "PLEASE TYPE YOUR NAMES":PRINT
1030 PRINT "PLAYER ";P;:INPUTP$(P):IFP$(P)=""THEN1050
1040 P=P+1:GOTO1030
1050 P=P-1:N9=P+5:L$="":X=INT(1+P*RND(0))
1060 X9=X-1:IFX=0THENX9=P
1700 IF P<3 THENC1=5
1710 IF P=3 THENC1=4
1720 IF P=4 THENC1=3
1730 IF P>4 THENC1=2
1740 FOR I=1TOP:C(I)=C1+1:NEXTI
1750 PRINT "EACH PLAYER CAN CHANGE THE LETTERS ";C1;"TIMES"
1760 GOSUB 7000
1800 REM
1810 REM SET UP DISPLAY AND INPUT CHOICE
1820 REM
1830 REM HOME CURSOR AND CLEAR SCREEN AS BEFORE
1840 REM
2000 PRINT CHR$(K1),CHR$(K4),CHR$(K2),
2010 PRINT "PLAYER",TAB(20),"LETTERS",TAB(35),"VALUE    SCORE"
2020 PRINT :FORI=1TOP
2022 IF A9=1 THENIFIC<>X1 THENPRINT:GOTO2035
2025 PRINT P$(I)
2030 PRINT TAB(20),L$(I),TAB(36),V(I),TAB(44),S(I)
2035 NEXT I:PRINT
2040 PRINT "LETTERS",TAB(20),
2042 FOR I=1TOLEN(L$):PRINTMID$(L$,I,1)," ";:NEXTI:PRINT
2050 PRINT "VALUES",TAB(20),
2052 FOR I=1TOLEN(V$):PRINTMID$(V$,I,1)," ";:NEXTI:PRINT
2060 PRINT CHR$(K2),
2070 PRINT :PRINT"WHAT WILL YOU TAKE, ";P$(X);
2080 PRINT CHR$(K3),:INPUTA$
2090 A9=1:PRINTCHR$(K4),
2500 REM
2510 REM TAKE ONE LETTER
2520 REM
3000 IF A$="" THEN3200
3005 IF LEN(A$)>1THEN4000
3010 FOR I=1TON9
3020 IF MID$(L$,I,1)=A$THEN3050
3030 NEXT I:FORI=1TON9:IFMID$(L$,I,1)="+" THENA$="+":GOTO3010
3035 NEXT I:PRINT"THERE'S NO ";A$;" TO TAKE";
3040 PRINT CHR$(13),CHR$(11),CHR$(,11),:GOTO2070
3050 L$=LEFT$(L$,I-1)+RIGHT$(L$,N9-I):L$(X)=L$(X)+A$
3060 V$(X)=V$(X)+MID$(V$,I,1)
3070 V$=LEFT$(V$,I-1)+RIGHT$(V$,N9-I)
3080 N1=INT(1+N*RND(0))
3090 READ I$,N2,V:N1=N1-N2:IFN1>0THEN3090
3100 RESTORE :L$=L$+I$:V$=V$+STR$(V)
3130 V(X)=0:FORI=1TOLEN(V$(X))
3140 V(X)=V(X)+VAL(MID$(V$(X),I,1)):NEXTI
3150 GOTO 5000
3170 REM
3180 REM CHANGE THE LETTERS
3190 REM
3200 C(X)=C(X)-1:IFC(X)>0THENGOSUB7000:GOTO3250
3210 PRINT "SORRY - YOU'VE HAD ";C1;"CHANGES ALREADY";
3220 GOTO 3040
3250 PRINT CHR$(13),CHR$(11),CHR$(11),CHR$(11),
3260 X1=X:X=X+1:IFX=P+1THENX=1
3265 IF X1<>X9THENPRINTCHR$(K2),:GOTO2040
3270 FOR I=1TOP:IFS(I)>S9 THENS9=S(I):I9=I
3280 NEXT I:IFS9<200 THENPRINTCHR$(K2),:GOTO2040
3290 PRINT :PRINT:PRINT:PRINT:GOTO5030
3500 REM
3510 REM MAKE A WORD
3520 REM
4000 IF LEN(L$(X))=0THEN4090
4030 S1=0:S$=L$(X):T$=V$(X):FORI=1TOLEN(A$)
4040 FOR J=1TOLEN(S$)
4050 IF MID$(A$,I,1)=MID$(S$,J,1)THEN4200
4060 NEXT J:FORJ=1TOLEN(S$)
4070 IF MID$(S$,J,1)="*"THEN4200
4080 NEXT J
4090 PRINT "IMPOSSIBLE!!":GOTO3040
4200 S$=LEFT$(S$,J-1)+RIGHT$(S$,LEN(S$)-J)
4210 S1=S1+VAL(MID$(T$,J,1))
4220 T$=LEFT$(T$,J-1)+RIGHT$(T$,LEN(T$)-J)
4230 NEXT I:INPUT"ANY OBJECTIONS",I$
4240 IF LEFT$(I$,1)="Y" THENPRINTCHR$(11),:GOTO3040
4250 L$(X)=S$:V$(X)=T$:V(X)=V(X)-S1
4260 S1=S1-V(X):IFS1<1 THENS1=1
4270 S2=LEN(A$)-LEN(L$(X)):IFS2<1 THENS2=1
4280 S(X)=S(X)+S1+S2
4290 PRINT CHR$(11),CHR$(K4),
4500 REM
4510 REM SET UP FOR NEXT PLAYER
4520 REM
5000 IF X<>X9THEN5500
5010 FOR I=1TOP:IFS(I)>S9 THENS9=S(I):I9=I
5020 NEXT I:IFS9<200 THEN5500
5030 PRINT "*** ";P$(I9);" IS THE WINNER ***"
5040 PRINT CHR$(K5):PRINT"SCORES -":PRINT
5050 FOR I=1TOP:PRINTP$(I),TAB(20),S(I):NEXTI
5060 PRINT :PRINT:PRINT:STOP
5500 X1=X:X=X+1:IFX=P+1 THENX=1
5505 PRINT CHR$(11),CHR$(K2),
5510 FOR I=1TOP+6:PRINTCHR$(11),:NEXTI
5520 GOTO 2020
6900 REM
6910 REM SUBROUTINE TO SELECT THE POOL OF LETTERS
6920 REM
7000 L$="":V$="":FORI=1TON9:N1=INT(1+N*RND(0))
7010 READ A$,N2,V:N1=N1-N2:IFN1>0THEN7010
7020 L$=L$+A$:V$=V$+STR$(V):N1=0:RESTORE:NEXT:RETURN
7030 END
```

The program requires a BASIC with full string handling facilities, including string arrays, and will run in 4-5k (depending on the number of players) over and above the interpreter. Where *multiple* statements on one line follow an IF . . . THEN it is essential that on the test failing, control is passed to the next line, not to the next statement on the same line. This may seem obvious, but not all BASICs follow the same conventions.

COMPUTER IN THE CLASSROOM
TEACHING PACKAGES

Charles Sweeten

It is becoming apparent that within the next five years many schools will have the opportunity to have a computer in the classroom. Those that are lucky enough to have one now will realise that there are possible ways of using the computer other than for teaching computing. One such way is to use the computer as a tool to illustrate a method or a problem to a class.

You need a computer that will run a fast and efficient version of BASIC. The BASIC must handle program files easily and quickly, and ideally these should be contained on a disk. You will need a large screen which can be connected to the keyboard so that a whole class can read it at a time.

The teaching package consists of teacher notes, pupil notes, details of knowledge required, together with a program that requires no knowledge of computers and virtually no knowledge of programming. Knowledge of how to control a terminal is necessary of course. The program itself should be foolproof and this is no easy matter to achieve. The program should be self explanatory in operation and it should not be necessary to list it or modify it in any way once it has been adapted for a particular system.

The idea is that the teacher should lead the class towards the point to be illustrated, and should provide the theoretical or practical experience that may be required. Having achieved a sufficient level of competence in the class the package may be run in the way that is best suited to that package. It is of course essential that the requirements of knowledge to be expected from the class are known exactly. Otherwise a class will be as unhappy as they usually are with a chapter of a book that is either too hard or too easy for them.

I will now take these points together with an example from A-level Mathematics. Many single subject mathematics courses include some statistics, and where this is not the case there is usually a separate Statistics A-level. One of the most difficult concepts to grasp is that of taking samples and of making judgements based on such samples. Most teachers would agree that the best way of approaching this topic is by experimental work, but there is frequently little time to do much that is worthwhile. It would be pleasant to get away from those boring experiments with dice, and measure something which has relevance to the class. Different classes will have different interests, but many who study statistics will have an interest in biology or social implications. The experiment that I

have designed is about a river that is being polluted by a factory.

The class is asked to take samples from the polluted river in order to establish whether it is sufficiently polluted for someone to take action of some kind. For example, if fish are dying then the River Authority may take legal proceedings. They are told a little about the difficulties of sampling the river adequately, and they are told about the difficulty of estimating the accuracy of their measurements. They have to make a decision as to how the results are to be tabulated in groups and they have to decide how many measurements to take to establish any sort of an answer.

The knowledge required by the pupils is experience in using the Normal distribution, together with some knowledge of the relation between the mean and standard deviation of the population and those of the sample. In this case the mean of the population is the true pollution level of the river (whatever that may mean), and it is this level that we are attempting to discover. Some discussion of the actual process of sampling will be needed as most pupils have little idea of what this involves. In this case discussion might centre on making up a plan for taking the samples, in terms of point of the river, depth, time of day, and etc .. It is also worth discussing how long it would take to carry out the work (don't forget travel and meal breaks).

The class should have discussed the effect of arranging measurements into groups and should be aware that choosing the group widths too wide or too narrow will conceal the shape of a distribution, but the package is designed to let them see this for themselves.

Pupils should also have some knowledge of the distribution of sample means so that discussion can take place over the range within which the true pollution level may be expected to lie.

In this particular package teacher and pupil notes are the same but there is no pressing reason that they should always be so. The notes are shown in figure 1.

Figure 2 shows a sample run. The user first decides whether instructions are wanted. They then decide how many samples they want to take from the river and in return they are given the time it will take to do this and process the results. (It is worth discussing with a class the effects of too few or too many samples).

They are then asked to decide how to classify the readings in groups. This is a practical point that is often glossed over because of the lack of real experimental work, but makes a great deal of difference to the usefulness of statistical work. If the user requests a ridiculous number of groups the program will refuse to cooperate and asks for more realistic instructions. The program will take samples from a Normal distribution of pollution levels, whose population mean and standard deviation are themselves chosen randomly. The full statistical calculations are then set out. The package may be modified and left on the computer system for pupils, and in this mode all that may need to be done is delete lines   to   , so that the pupils must work out the figures for themselves.

The package has been tested thoroughly on a class of 2nd year sixth form boys at Oundle School. Of these perhaps half may fail A-level Mathematics in the summer and the rest are not expecting high grades. They lack confidence and have little enthusiasm for anything that will further diminish that confidence. They know very little about computing but have been quite enthusiastic about using the package on their own. I have often found a

couple of them at a terminal trying out the packages they have done some four months earlier. It is always hard to tell whether some educational methods are an advance on others, but I think the package approach has given this particular group more motivation than they would have had otherwise.

MATHS RIVER STATISTICS Normal: Estimate μ

You will be supplied with information taken from a polluted river. The pollution is caused by a factory who will only make an effort to clean up their effluent when enough pressure is put on them. You are being asked to provide the evidence. The effect of action against the factory is unfortunately only temporary, and afterwards pollution levels gradually return to the same levels as existed before. The factory never releases information about any action they may have taken; nor do they admit that they have taken action. So it is impossible to judge when to start bringing pressure to bear other than by sampling the river.

Pollution levels appear to vary throughout each day and vary at different depths and places, but no pattern has been observed. The method of measurement gives variable results in any case, due to causes such as the time between the sample being taken and being analysed, the temperature of the water and of the air, etc. It is not possible to do anything about this as sufficient funds are not available.

You are required to show that the samples taken from the river form roughly a Normal distribution. It is known that the level at which fish begin to die is roughly 15 pu's (pollution units) and you are asked to provide evidence as to whether immediate action is necessary or preliminary pressure or no action.

The machine which analyses the samples gives readings correct to .0001 pu, but this is no indication of the actual accuracy of your samples. In fact the last inspector who did the work thought that his sample measurements were only accurate to 1 pu due to the various factors such as temperature, depth, etc. that we have already mentioned. You must decide for yourself what accuracy you want in the table of results.

The readings will be grouped by the technicians who do the analysis into groups and they will need to be told the width of these groups.

You must decide before you start how many samples you think will be needed to establish your results. Remember though that sampling takes time. The width of your groups will depend on the number of samples you take; the fewer samples the wider will be your groups. The idea is to get a histogram which is more or less smooth.

You will first be asked the number of samples : you may answer any number from 1 to 10000.

The group width should be anything from .0001 to 10.

Here is a sample of the program :

```
You must decide before you start how many samples you think will
be needed to establish your results.  Remember though that sampling
takes time.  The width of your groups will depend on the number of
samples you take; the fewer samples the wider will be your groups.
The idea is to get a histogram which is more or less smooth.

    You will first be asked the number of samples : you may answer
any number from 1 to 10000.

    The group width should be anything from .0001 to 10.

    Here is a sample of the program :

HOW MANY SAMPLES ARE YOU TAKING FROM THE RIVER ?  200
IT WILL TAKE 48 HOURS TO PROCESS THE RESULTS.
    IS THAT ALRIGHT ? YES

YOU ARE GOING TO GET READINGS SUCH AS 14.8647
THE READINGS WILL BE GROUPED IN A FREQUENCY TABLE
HOW WIDE DO YOU WANT EACH GROUP TO BE ? .5
```

```
FREQUENCY TABLE
-----------------------
11.5   -  12    XXXXX  3
12     -  12.5  XXXXX  3
12.5   -  13    XXXXXXXXXXX  8
13     -  13.5  XXXXXXXXX  6
13.5   -  14    XXXXXXXXXXXXXXXXXXX  14
14     -  14.5  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  30
14.5   -  15    XXXXXXXXXXXXXXXXXXXXXXXXXXX  20
15     -  15.5  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  33
15.5   -  16    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  27
16     -  16.5  XXXXXXXXXXXXXXXXXXXXXXXXXXX  20
16.5   -  17    XXXXXXXXXXXXXXXXXXXXXXXXXXXXX  22
17     -  17.5  XXXXXXXXXXX  7
17.5   -  18    XXXXXXXX  5
18     -  18.5  XXX  2
18.5   -  19    0
-----------------------
PRESS RETURN WHEN YOU HAVE READ THIS


MEAN = 15.1825 PU        STANDARD DEVIATION = 1.33574  PU

ESTIMATE OF POPULATION STANDARD DEVIATION = 1.3391 PU

STANDARD DEVIATION OF THE MEAN = 9.46883E-02 PU

95% CONFIDENCE LIMITS FOR THE MEAN ARE
14.9969 PU  TO  15.3681  PU
```
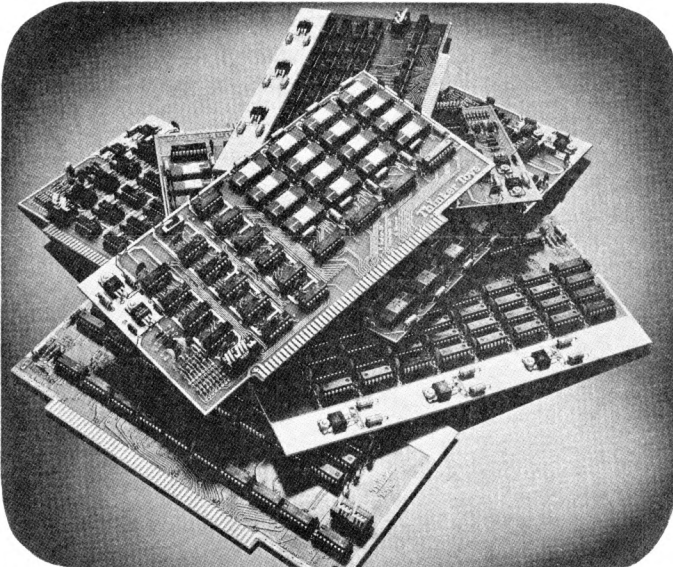
```
0010 REM *****RIVER*****
0020 REM THIS VERSION RUNS IN ZPL 8K BASIC
0030 REM TESTED ON RM 380Z
0040 CLEAR 200
0050 RANDOMIZE
0060 PRINT CHR$(12)
0070 PRINT "        ";
0080 REM SCREEN SHOULD NOW BE CLEAR
0090 PRINT
0100 PRINT "SIMULATION OF RIVER POLLUTION"
0110 PRINT "-----------------------------"
0120 PRINT
0130 PRINT
0140 PRINT "DO YOU WANT INSTRUCTIONS  ";
0150 INPUT Q$
0160 IF LEFT$(Q$,1)="Y" THEN GOTO 1200
0170 PRINT
0180 PRINT
0190 LET M=14+4*RND(9)
0200 LET S=.5+RND(9)
0210 PRINT "HOW MANY SAMPLES ARE YOU TAKING FROM THE RIVER  ";
0220 INPUT N
0230 LET D=INT(N/55+1)
0240 PRINT "IT WILL TAKE ";
0250 IF D=1 THEN PRINT "24 HOURS";
0260 IF D=2 THEN PRINT "48 HOURS";
0270 IF D>2 THEN PRINT D;"DAYS";
0280 PRINT " TO PRODUCE THE RESULTS ."
0290 PRINT "  IS THAT ALRIGHT    ";
0300 INPUT A$
0310 IF LEFT$(A$,1)="N" THEN GOTO 0170
0320 PRINT
0330 PRINT "YOU ARE GOING TO GET READINGS SUCH AS 14.8647"
0340 PRINT "THE READINGS WILL BE GROUPED IN A FREQUENCY TABLE"
0350 PRINT "HOW WIDE DO YOU WANT EACH GROUP TO BE ";
0360 INPUT G
0370 IF G<=0 THEN GOTO 1200
0380 LET A$="THAT WILL GIVE MORE THAN 250 GROUPS !"
0390 IF 6*S/G>250 THEN PRINT A$
0400 IF 6*S/G>250 THEN GOTO 0350
0410 LET R1=M-3*S
0420 LET R1=INT(R1/G)*G
0430 DIM X(6*S/G)
0440 LET F=0
0450 FOR N1=1 TO N
0460   GOSUB 2000
0470   LET Y=INT((X-R1)/G)+1
0480   LET X(Y)=X(Y)+1
0490   IF X(Y)>F THEN LET F=X(Y)
0500 NEXT N1
0510 LET K=1
0520 REM THE DISPLAY MAY BE WIDENED BY CHANGING 19 TO 40
0530 IF F>19 THEN LET K=19/F
0540 REM THIS SECTION SETS A DELAY PROPORTIONAL
0550 REM TO THE DAYS REQUIRED
0560 PRINT "SAMPLING HAS NOW STARTED"
0570 FOR T=1 TO 30000*D
0580 NEXT T
0590 PRINT
0600 PRINT "FREQUENCY DISTRIBUTION"
0610 PRINT "----------------------"
0620 LET M1=0
0630 LET S1=0
0640 FOR Y=1 TO 6*S/G+2
0650   PRINT R1+(Y-1)*G;
0660   PRINT TAB(6);"- ";
0670   PRINT R1+Y*G;
0680   PRINT TAB(15);
0690   FOR C=1 TO INT(X(Y)*K+.5)
0700     PRINT "X";
0710   NEXT C
0720   PRINT " ";X(Y)
0730   LET X=R1+(Y-.5)*G
0740   LET M1=M1+X(Y)*X/N
0750   LET S1=S1+X(Y)*X*X/N
0760 NEXT Y
0770 PRINT "PRESS ANY KEY FOLLOWED BY RETURN"
0780 PRINT "WHEN YOU HAVE READ THIS";
0790 INPUT Q$
0800 LET S1=SQR(S1-M1*M1)
0810 PRINT CHR$(10)
0820 PRINT "MEAN =";M1;" PU","STANDARD DEVIATION =";
0830 PRINT S1;" PU"
0840 LET S2=S1*SQR(N/(N-1))
0850 PRINT CHR$(10);"ESTIMATE OF POPULATION STANDARD";
0860 PRINT " DEVIATION =";S2;" PU"
0870 PRINT CHR$(10);"STANDARD DEVIATION OF THE MEAN =";
0880 PRINT S2/SQR(N);" PU"
0890 PRINT CHR$(10);"95% CONFIDENCE LIMITS FOR THE MEAN ARE"
0900 PRINT M1-1.96*S2/SQR(N);" PU TO ";M1+1.96*S2/SQR(N);" PU"
0910 LET M=0
0920 LET S=0
0930 END
1200 REM *SO THAT THE INSTRUCTIONS CAN BE READ ON YOUR
1210 REM *SCREEN COUNT THE NUMBER OF LINES DISPLAYED AT
1220 REM *A TIME--SUPPOSE THIS IS N LINES--THEN INSERT
1230 REM *THE FOLLOWING INSTRUCTIONS AFTER EVERY N LINES
1240 REM PRINT "PRESS ANY KEY FOLLOWED BY RETURN"
1250 REM PRINT "WHEN YOU HAVE READ THIS";
1260 REM INPUT A$
1270 PRINT CHR$(12)
1280 PRINT "        "
1290 PRINT "  YOU WILL BE SUPPLIED WITH INFORMATION TAKEN FROM"
1300 PRINT "A POLLUTED RIVER.  THE POLLUTION IS CAUSED BY A"
1310 PRINT "FACTORY WHO WILL MAKE AN EFFORT TO CLEAN UP THEIR"
1320 PRINT "EFFLUENT ONLY IF YOU CAN PROVIDE THE EVIDENCE."
1330 PRINT
1340 PRINT "  POLLUTION LEVELS APPEAR TO VARY THROUGHOUT EACH"
1350 PRINT "DAY AND VARY AT DIFFERENT DEPTHS AND PLACES, BUT"
1360 PRINT "NO PATTERN HAS BEEN NOTICED."
1370 PRINT
1380 PRINT "  YOU ARE REQUIRED TO SHOW THAT THE SAMPLES TAKEN"
1390 PRINT "FROM THE RIVER FORM ROUGHLY A NORMAL DISTRIBUTION"
1400 PRINT
1400 PRINT "  IT IS KNOWN THAT FISH BEGIN TO DIE AT ABOUT 15"
1410 PRINT "PU (POLLUTION UNITS) AND YOU ARE REQUIRED TO"
1420 PRINT "PROVIDE EVIDENCE AS TO WHETHER IMMEDIATE ACTION"
1430 PRINT "IS NECESSARY."
1440 PRINT
1450 PRINT "  THE MACHINE WHICH ANALYSES THE SAMPLES GIVES"
1460 PRINT "READINGS CORRECT TO .0001 PU , BUT THIS IS NO"
1470 PRINT "INDICATION OF THE ACTUAL ACCURACY.  IN FACT THE"
1480 PRINT "LAST INSPECTOR WHO DID THE WORK THOUGHT THAT HIS"
1490 PRINT "SAMPLE MEASUREMENTS WERE ONLY ACCURATE TO 1 PU DUE"
1500 PRINT "TO THE VARIOUS FACTORS SUCH AS TEMPERATURE, AND"
1510 PRINT "EXPOSURE TO AIR, AND ETC.,  YOU MUST DECIDE FOR"
1520 PRINT "YOURSELF WHAT ACCURACY TO TAKE"
1530 PRINT
1540 PRINT "  THE READINGS WILL BE GROUPED BY THE TECHNICIANS"
1550 PRINT "WHO DO THE ANALYSIS INTO BANDS AND THEY WILL NEED"
1560 PRINT "TO KNOW THE WIDTHS OF THESE GROUPS BEFORE THEY"
1570 PRINT "START."
1580 PRINT
1590 PRINT "....THE NUMBER OF SAMPLES: ANYTHING FROM 1 TO 10000"
1600 PRINT "....IS THAT ALRIGHT: YES OR NO OR Y OR N"
1610 PRINT "....HOW WIDE IS EACH GROUP: FROM .0001 TO 10"
1620 PRINT
1630 GOTO 0130
2000 REM THIS GIVES SAMPLES FROM A NORMAL DISTRIBUTION
2010 LET X=6*RND(9)-3
2020 LET Y=RND(9)
2030 IF Y*4*EXP(-X*X/2)/2.50663 THEN GOTO 2000
2040 LET X=M+X*S
2050 RETURN
```

⨅INTERAM
puts its cards
on the table . . .

Wonderful things have happened to computers since the introduction of the S-100 BUS in 1975. Reliability has shot up like a rocket and prices have dropped like a rock. Today there are now over 100 manufacturers making plug-compatible computers, circuit-boards and peripherals. INTERAM supply a fully comprehensive range of computer Memory boards, Intelligent Front-Panels, Buss-Boards, I/O Interfaces, Control Boards, Extender Cards, Speech-Recognisers and Synthesisers. All of which plug directly into your Altair/EQUINOX/IMSAI etc. S-100 Bus computer.

THE KEYED-UP 8080

This board is a combination 8080A CPU & Intelligent Front-Panel. The 12-Key gold contact keyboard works in conjunction with ½K of on-board ROM & RAM to allow you to examine or deposit data into any Register, Register Pair, Memory location or I/O Port. Single step or SLOW-STEP at a programmable rate through your program. The ten 7-segment LED's offer a concise OCTAL display which all in all add up to control over your computer so complete it surpasses any mini-computer! £225

THE SPEAKEASY I/O BOARD

Slot this single board into your S-100 BUS computer and you're ready to plug in 3 audio cassette units, a Teletype or RS232 terminal and any 8-bit parallel device . . . graphics display, line printer or paper tape reader. On-board ROM & RAM pre-programmed with COPE handles dozens of interfacing and data handling tasks. COPE can even bootstrap your programs. £135

THE WUNDERBUSS

This the heart of every S-100 BUS computer — the BUS itself. No other bussboard, though, has each bussline isolated by an interlaced system of ground lines and cross-coupled ground planes plus active termination to prevent cross talk and eliminate signal reflection. £75

ECONORAM II 8K STATIC RAM

This S-100 compatible memory board features two independent addressable 4K blocks. All lines fully buffered, tri-state outputs, wait state and write strobe select switches. Board access time of 450ns. Can handle DMA devices. £135

AI CYBERNETICS SPEECH SYNTHESISER

This is a revolutionary form of peripheral for your S-100 Bus computer. Interactive computing will never be the same again once your computer has actually spoken to you. The unit is a hard-wired analog of the human vocal tract. All of the information necessary to produce English speech has been programmed into ROM's which reside on the synthesiser board. £275

HEURISTICS SPEECHLAB SPEECH RECOGNISER

Now you can have low-cost voice entry to your computer. All models in the SPEECHLAB range plug directly into your S-100 BUS computer. Use it for scientific and engineering experiment data entry and countless other applications. From £145

MULLEN OPTO-ISOLATOR/RELAY CONTROL BOARD

Although a computer may perform interesting operations and programs, its usefulness in controlling real world devices — such as motors, audio signal lines, display lights, etc. — is severely limited. This board controls low-power devices directly, or triggers heavy duty relays or triacs. Additionally, 8 opto-isolators accept an 8-bit word from the outside world and send it to the computer for handshaking or further control purposes. £105

MULLEN EXTENDER CARD

When many boards are loaded side by side in a computer, trouble-shooting becomes almost impossible. This useful accessory "extends" a given board over the tops of other boards for easy trouble-shooting and examination. £39

⨅INTERAM
COMPUTER SYSTEMS LTD.

59 MORETON ST.,
VICTORIA,
LONDON SW1V 2NY

TEL: 01-834 0261/2733

ALL THE ABOVE BOARDS ARE FULLY ASSEMBLED, TESTED, AND WARRANTED FOR 1 YEAR

ALL PRICES EXCLUSIVE OF VAT

DEALER ENQUIRIES INVITED

The computer is just what the doctor hasn't ordered

Robert Johnson

Why is medical data processing still in its infancy? All other service industries have embraced computer techniques to the unprecedented benefit of themselves, and their clients. Only the health services lag behind: why has so little advantage been taken of data processing techniques?

The problem is not a small one. The National Health Service costs £7 billion a year to run. It has more employees than any other service industry, and shortly (in 1979) the drugs prescribed by general practitioners will cost £1 billion a year.

Cost, however, is the least of the hazards. The very foundation of health care is now in jeopardy, merely through faulty data processing. Tragedies such as Thalidomide and Eraldin will recur with increasing frequency simply because the drug information cannot humanly be processed in time.

Information pollution?
These assertions may seem alarmist, perhaps sensational, but we do not require a medical education to appreciate their inevitability. Indeed, it might almost be said that a medical training makes it more difficult to visualise the simple logistics of the situation. The quantity of drug information is constantly growing: new drugs are being marketed, and new facts about old drugs being published. Using conventional methods, this information is delivered to every GP in the country in the form of monthly and annual books with official government approval, if not statutory backing.

Disregarding all the promotional literature and medical periodicals supported by advertising, each GP receives a mammoth total of $2\frac{1}{4}$ million words every 12 months. This figure could rise in the 1980s to four and then five million.

To computer personnel, two and a quarter million words may translate roughly to 10 Megabytes: small beer compared with the hundreds of megabyte storages now available. But for the humble GP, these are not on electronic data files: they are printed words on paper pages — side by side they constitute a book 6 in. thick; and when one year finishes, there is another 6 in. for the next.

If this is not deterrent enough, the cards on which patients' clinical records are stored are similarly exploding in size. It stands to reason that as more elaborate clinical tests are devised, and more biochemical readings are fished from the blood stream and elsewhere, the results must be stored somewhere.

And, since the GP sees the patient three or four times as often as do the hospitals, for the client to benefit from these exertions at all the results must be available in the GP's surgery. On average it has been estimated that each patient's record will grow in this manner by 8% annually, doubling therefore every decade.

Picture then the GP sitting at his surgery desk with 6 in. of drug books on one side, and a 1 in. wadge of records for each patient on the other. He has to extract relevant information from each 'data bank', compare it with the details being presented in the current interview, and then prescribe an appropriate, economic, safe and effective treatment.

The picture is not quite complete. In a 12-month period in general practice, I saw 11,365 patients — roughly 200 a week. With a nominal 40 hour week, this allowed a gross average of 12 minutes for each patient contact.

If an additional five minutes were required to check an item that had slipped the doctor's (or patient's) memory, the penalty incurred would be an additional 16 hours per week. So yet another natural resource, time, is running out.

One might have thought there would have been something of an outcry from doctors, or a little more activity on the part of the Department of Health and Social Security to remedy a situation that is already far from healthy.

The gulf between need and action
If the need is there, though not officially acknowledged, why has so little been done? The primary obstacle is obviously the DHSS. What other service organisation could justify spending less than 0.5% of its budget on research into improved data processing?

But the chief difficulty is lack of imagination and fear of innovation. It is illusory to hope that modern electronic techniques can be grafted onto centuries-old medical practices, without profound and radical analysis of both.

The illusion persists, and will prove increasingly costly. Doctors have been reluctant to tackle the basis and essentials of computation, and few computer experts would volunteer to spend a decade learning medical techniques from the inside.

Computers accordingly have a bad name in the medical profession, and their progress is therefore resisted. This is partly because of a few early but sensational failures, partly because of cost, and partly because there is no real understanding of how today's technology can assist so skilled an area as medical practice.

In my view the practice of medicine will not survive without computer assistance. There is an enormous virtually untapped market for both hardware and software; and there are simple reasons why most early medical computing attempts were doomed.

Taking each of these points in reverse order: most extant medical computing systems store records in free (that is, uncoded) text. The clinical data going in may be of the highest quality, but there is simply no *cost effective* method of getting it out, unless one is prepared to forgo all the benefits of data processing.

Medical data *must* therefore be entered in numeric or digital form — a requirement which is greeted with the profoundest suspicion by the medical profession. You might as well be asking them to write their notes blindfold. Nevertheless 'free text in, free text out' achieves little, at great expense — something which was not applauded by a recent Parliamentary accounts committee.

Cost is a convenient hedge behind which most doctors will hide — a position rendered untenable of course, by the microprocessor. For £1,000, a kit using audiocassettes, domestic TV screens, and perhaps 16K RAM of memory would provide the basic requirements for a sophisticated medical computing system, to which more could be added in due time.

The numeric codes need not be memorised, but can be read *from the screen* by the doctor, or, if need be, his receptionist or secretary. Being exclusively numeric, the data can be tightly packed. Thus the entire 12 months records for a busy general practitioner can be stored on a single C60 audio-cassette, costing say 20p in bulk. Five thousand pounds would then cover the entire annual medical records of all 25,000 GPs in the NHS.

Once entered the data could at last be manipulated electronically. The machine would cease to be an over-grown electronic notebook, but become an active *processor* of the numerically encoded medical data. So having coaxed the medical profession into adopting more up-to-date recording techniques, what can computerisation offer them?

The pay-off
From a commercial point of view, the administration of general practice is a gold mine for microcomputer-based management systems. The GP of today is increasingly paid on an 'item of service' basis: that is to say, of the myriad individual 'treatments' he delivers, for some he receives a fee which goes up and down depending on fairly arbitrary rules: and for others reimbursement, which again wanders over a wide range. In such a topsy-turvy world, the GP can easily lose up to £1,000 a year, in 'lost' fees, either claimed too late or not at all.

Leasing a system for £5-£10 a week would be very attractive to the average group practice, and could soon be shown to pay for itself, by running electronic rings around the paper-bound log-jams of the local administrators. A game of profit and skill that many GPs would relish.

Prevention is better than cure
Cost in terms of cash can no longer therefore be used as a barrier to further medical computing. Costs in terms of drug-induced diseases are only just being reckoned. Thalidomide is well known, and had an efficient drug data distribution service been in operation at the time, there would have been no victims in this country. Eraldin is perhaps less well known, but is even more relevant.

Eraldin is an excellent drug for angina, but it took five years to find that it sent some patients blind, and gave others a lethal form of peritonitis which had not been met with before.

If such poisonous effects occur at a rate of only one in 10,000 or even one in 1,000 the individual GP prescribing the drug has just no chance of picking them out from the 30,000 or so symptoms he sees every year.

Only by pooling data from many if not all GPs can a feasible strategy be adopted. Official proposals for more paper work for newly-released drugs are likely to sink under their own weight and cost. Meanwhile, we, as a community, continue to incubate other 'Eraldin-type' disasters.

Using the system mentioned, every GP (or if he is unwilling, his receptionist) would *record* the symptoms presented, the diagnosis derived, and the treatment prescribed. These would be stored initially on a 2K RAM scratch pad, to be read off at leisure after the surgery onto the 20 pence C60 cassette. 2K of eight-bit bytes will cover 50 consultations on

average, enough to handle all but the most Herc-ulean workload.

Computerisation thus offers the GP savings in his practice costs, it offers the first practicable means of detecting totally unforeseeable side-effects from all drugs prescribed, whether newly-released or of longstanding 'respectability'.

I estimate that there are around 500 million symptoms presented to GPs every year in the UK, 50 million of these are pains, 15 million headaches. *Prompt* main-frame analysis of data of this magnitude and topicality would be of unprecedented value, not only to drug companies with new products, but but also to the Committee on the Safety of Medicines, and of course to the GP himself.

It could inform him in retrospect as to the effectiveness of his particular treatment in delaying or abolishing specific symptoms, or any other research topic which took his interest.

The patient (that is to say ourselves) will benefit most, when the system is developed to operate in real time, interactively. Costs would double to say £2,000 capital, £10-15 lease per week, since a floppy or other disk would be required.

Filling in the prescription

An example of a computerised system for GPs is the APRIL system. With such a system, the benefits of computerisation have an impact everyone, especially the public, can appreciate.

Indeed, it may well be the case that only public pressure will induce the DHSS and the NHS it administers to take a more active interest in such systems.

The APRIL system analyses each individual patient record, and *compares it electronically and infallibly with the data on the drug about to be prescribed*, with the consequence that no known harmful drugs will be received. Drug treatment is optimised to suit each individual patient.

The basis of the APRIL system is as follows: once the clinical data and the drug information have been encoded in the same numerical code, adverse interactions between the two may be detected simply by subtraction.

Programmes for this are still being developed, but they need not exceed 8 or 16K. Compared with the Megabytes of lexicons, and the large data banks required for the 'free text' exponents, the amount of hardware required is minuscule.

One further technical point. In the current prototype, only 4K bytes are set aside for the 10-letter words used for visual decoding of the numerics on the screen. This size of memory is adequate for the simplest system which covers those items which occur with greater frequency than once weekly on average.

The law of diminishing returns applies to the remainder, which could with this basic system only be covered in generalities. Larger studies could readily determine which additional items need including. Similarly, it is possible to envisage a heuristic model, whereby the drug names used most commonly by a given doctor receive more space in his personalised system.

With the APRIL system operational, it would be impossible for you or me to be prescribed a drug that had previously been known to poison someone with your (or my) particular clinical history. Drugs suspected, for example, of affecting embryos (as with Thalidomide) would automatically be screened against possible indications of pregnancy on the part of the patient, BEFORE the drug could be prescribed,

so providing a degree of security in safe prescribing quite unobtainable elsewhere or earlier.

Similarly, a past history of what could be construed as heart disease would automatically, and without further fallible human intervention, exclude a drug or combination of drugs found (anywhere in the world) to have been detrimental.

There are many points which cannot be included here: drug actions and interactions are immensely complex. Some drug interaction patterns require arrays of up to five or even 10 dimensions — a computation somewhat beyond the standard cerebral capacity of the normal hard-working GP.

The paper epidemic

A cautionary tale to end with. Drug-induced disease is an epidemic arising from the growth of new and more powerful drugs. Its cause is inefficient data processing. It is therefore analogous to the earlier self-inflicted epidemics, such as typhoid and cholera, which are consequences of inefficient sewage processing.

A combination of public and professional effort is needed to contain both types of epidemic. One famous episode from the annals of public health concerns the doctor who averted just such an outbreak by removing the handle of the 'Broad Street Pump'. He had deduced that the water from it was infectious, and acted accordingly.

In our present case, however, the pump handle in question is the actual pen with which the doctor writes and suggestions for improvement may not, as I can testify from experience, be immediately welcome.

I should not be at all surprised to find patients deploying computer systems before their doctors do. After all, as actual consumers of the treatment, their interest in the safety of the prescription is that bit keener.

"You're in luck, Mr Grimchin. By the time I find the drug you'll have died of old age."

PCW OPEN PAGE

The Amateur Computer Club View

Mike Lord

Homer G. Homebrew

So PET has finally arrived in the U.K. — or to be precise a system has been sighted and CBM are now willing to take your money and put your name onto the queue. A pity that they are offering only the larger, 8K, version: £695 is a bit much for the average amateur. However, CBM have obviously decided to aim for the top end of the market while demand exceeds supply, and why not? This is a fascinating machine, the graphics look superb and their extended version of BASIC seems to be *powerful and easy to use.* A pity they didn't go for a more solid looking keyboard than the calculator style 'Finger Fumbler' that they have fitted.

Nearer to the size of the average amateur's wallet; the saga of Lynx's NASCOM 1 continues. At the time of writing I haven't heard from anyone who has actually received a kit; but Lynx say that the PCB production problems are now ironed out and within a week or so deliveries will start. Although not a Z80 enthusiast myself (it's too complicated a device for me to remember how to use it from one weekend to the next) I think NASCOM is a significant milestone in the U.K. amateur computing scene, and look forward to hearing users' experiences with it.

Both PET and NASCOM 1 attracted great attention at a recent ACC London Group meeting, which also featured several homebrew systems, including **Neil Harrison's** Z80 running some nice game programs; where does he get the time from?

The Midlands Section of the ACC has been holding successful meetings at approximately six-week intervals, anyone interested in attending should contact **Ray Diamond,** 27 Loweswater Rd., Coventry CV3 2HJ (telephone Coventry 454061) for the latest schedule.

One group which obviously believes in planning ahead is **ACC NorthWest.** They are running a series of elementary meetings, which occupy the first half-hour of each meeting and are followed by a more advanced talk, starting at 6.30 in the National Computing Centre, Oxford St., Manchester. The current program is;

    4 May   Computer Arithmetic
    8 June   Computer Architecture

However, in case of any last minute snags, anyone wishing to attend should confirm with **Ken Horton,** 50 Lymfield Drive, Worsley (061 799 0192).

Mancunian Micro Mashers: the Northern Radio Societies Association's exhibition on Sunday 2nd

April featured microprocessors and personal computing. Exhibitors booked included *Computer Workshop, Comart, Lynx, Newbear* and *LP Enterprises.*

Bob Cottis is starting a Thames Valley Group. centred on Stoke Poges. For further details contact him at 'Pippins', Boulters Lane, Maidenhead, Berks SL6 8JT (Maidenhead 22445).

All of these groups are run on a purely voluntary basis, therefore please remember that it is only common courtesy to include a SAE when writing to their organisers.

Southampton University

An enthusiastic amateur computer club also exists at Southampton University. They are building a 9900 based system for Hardware Hackers (bolt holes will be provided for all conceivable goodies) and are also purchasing a NASCOM kit so they can get something up and running quickly. As if all this wasn't enough, they also publish an excellent magazine; 'Benchmark'. **Paul Maddison** is the contact, at 47 Westridge Rd., Portswood, Hants.

Finally, the ACC has now reached the end of its fifth year, and the Annual General Meeting was held on Thursday 13th April, 7.30 pm at the South Bank Polytechnic, Borough Rd., near to the Elephant & Castle.

For further details of the ACC, send a SAE to Mike Lord, 7 Dordells, Basildon, Essex.

Developments in computing at Five Ways

The approach we are adopting to computing at King Edward VI Five Ways School, Birmingham, that of using the computer as a problem-solving tool rather than to teach programming, is one that has aroused considerable interest among both representatives of industry and secondary education. We bring this to your attention as we hope it may be of interest to your readers.

The school currently possesses four terminals accessing the Newcastle-upon-Tyne Open University computer which we use extensively as an aid to course work in such 'O' and 'A' level subjects as Statistics, Economics, Chemistry, Geography and Mathematics. Some limited experiments in English,

French and Latin are also being undertaken. In addition to this work, every sixth-former, starting with the present Lower Sixth, follows a half-year course which aims to give practice in the approach to and solving a variety of problems, learning how the computer can assist him in this process, and it is this course that has attracted the interest of local industry. Representatives from BL Cars, Cadbury-Schweppes and Rubery Owen have visited the school and, arising out of these visits, some joint ventures are being planned. The first of these, a day-long business simulation exercise at the BL Cars Systems Training Centre at Studley, took place on 13 March and again on 14 March.

The programs we use at Five Ways are either those on the Open University system or are written by our programmers. These are boys who learn to program outside school hours under the guidance of staff and senior boys. Their training lasts for some two years before they start to write programs to the specification of subject staff, programs that, when tested, will be used in subject course work. The senior operators are also responsible for the day-to-day running of the terminal room.

If you would like to visit the school and take a closer look at what we do or to discuss our future plans please do not hesitate to telephone.

For further information: Mr A F R Clements, Director of Computing, Mr H Reeves, Head of General Studies, King Edward VI Five Ways School, Scotland Lane, Bartley Green, Birmingham B32 4BT. Telephone: 021 475 8944 or 021 475 3535.

*For 6800 users.

Having grown rather tired of being the recipient of a rather subtle advert every time my processor goes back into the BASIC command wait situation I decided to amend the appropriate area of memory by using a very small program written in BASIC. It is a building block for other tasks of a similar nature and can be altered very easily from that shown below.

In line 10 the variable I is set to the first decimal location at which the change is to be made. Line 20 reads the first item in data which is the decimal number 84 which is in the ASCII code. A check is made for a dummy number which, if equal to, branches to the end of the program. If not equal the number is POKE'd into the Ith location being converted into hexadecimal by the interpreter. Line 50 then loops back for another number and so on.

It is not strictly necessary to have the dummy and check as the FOR/NEXT loop would end before it got there anyway but is useful to show it as under different circumstances one could amend line 50 to read GOTO 20 and delete line 10. If you did this you could almost amend line 20 to be INPUT A which would be the decimal equivalent of the hex character and line 10 could be INPUT I. Line 45 would then read LET I = I +1 in order to increment the location number.

```
5 REM THIS PROGRAM CHANGES 'MSI READY'
  TO 'THE GHOST'
10 FOR I – 349 to 357
20 READ A
30 IF A = 99 THEN 90
40 POKE (I,A)
50 NEXT I
80 DATA 84, 72, 69, 32, 71, 72, 79, 83, 84, 99
90 END
```

COLIN CHATFIELD
Micro-Aid
Lloyds Bank Chambers
Camborne, Cornwall

EDITOR'S PLAYGROUND

THE SMALL SYSTEM APPROACH TO PROGRAM EDITING

Patrick Sutton

It is frequently stated that the approach to personal computing in Europe will be greatly influenced by the smaller amounts of money available for investment in computer hardware. To this end it is likely that the most popular machines in Europe will be small systems. It is my firm belief that this factor could favour development of systems which are far more user oriented and therefore better "personal computers".

The approach whereby the initial system is small and is only expanded when the need arises should, in my opinion, be greatly encouraged. A small system does not necessarily mean a weak system. On the contrary, I feel that by subscribing to large system approaches, a fair proportion of the advantages inherent to the personal microcomputer are lost or obscured. A good example of this is found in the area of program development aids. I hope, in this article, to illustrate this point by discussing a small system editor.

This particular algorithm has been available since 1966 but has been ignored by the majority of mainframe manufacturers and, following in their footsteps, many microcomputer manufacturers.

Line editors

The vast majority of microcomputer based editors available today (though not all, cf. the Research Machines and PET editors) are line oriented. This means that in order to edit the programme manuscript, numbered lines must be added or deleted. This is so even if the mistake occurs in only one character within a particular line. This type of editor evolved due to the extensive use of TTY terminals with mainframe machines. Here, single character editing is difficult to program because of the obvious printing limitations. However, most personal computer users today use visual display terminals (VDU's) and these printing limitations, ie, deletion of a character, are not a problem. Indeed, those VDU's which are memory mapped, that is, those whose character positions on the screen are available as addresses in memory, are highly suited to extensive text manipulations as I will mention later. Some of the character based editors that are available require the programmer to manipulate the text with a series of commands, *insert, replace* etc. With some practice this approach can be very effective. However, the approach discussed here is one which replaces editing commands by use of the visually obvious.

The scroll editor

In 1965 Mary Allen Wilkes developed what is known as a **scroll editor**. (1) This program was originally written on a Laboratory INstrument Computer (LINC) which included a keyboard, display oscilloscope, two randomly accessible tape drives and only 2K of store. (2)

Using the keyboard and 'scope interactively the programmer types in all corrections directly to the displayed manuscript. All editing commands, such as *line feed* and *rub out,* are delivered by single keystrokes. The major features of this scroll editor can be summarised as follows:

1) Characters can be inserted directly into the displayed manuscript from the keyboard. No separate insert instruction is required.

2) At any position in the manuscript any amount of material may be added or deleted. The manuscript may be moved across the display in either direction and edited in any order.

3) The editor is unspecific and machine code up to high level languages such as Basic may be edited using the same algorithm.
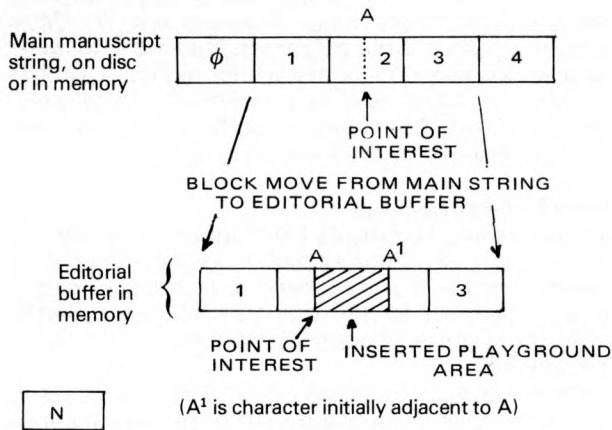
4) Line or sequence numbers are not stored in the manuscript. A reference can be made to a line position using a program-maintained line counter, and because of this approach any additions or deletions of lines are automatically accounted for.

5) User responsibility for manuscript maintenance, for example, packing of the characters, is removed and is dealt with by the editor.

The playground area

In order to understand just how this is accomplished, the reader should examine figures 1 to 4. These can only convey the general principles involved, and the interested reader is directed to Wilkes's original paper (Ref.1) in order to get more information in the form of flowcharts etc.

Fig. 1. Creating the playground area.

Main manuscript string, on disc or in memory

A

| φ | 1 | 2 | 3 | 4 |

POINT OF INTEREST

BLOCK MOVE FROM MAIN STRING TO EDITORIAL BUFFER

Editorial buffer in memory

A   A¹

| 1 | | ▨ | | 3 |

POINT OF INTEREST   INSERTED PLAYGROUND AREA

| N |

(A¹ is character initially adjacent to A)

Represents a functional block of characters i.e. 256 for a disc sector or 40 for a VDU line length

The point of interest is located at point A in block 2. Blocks 1, 2, 3 are moved to the editorial buffer in memory and a playground area is inserted between point A and the next character at point A¹ in block 2.

Figure 1 shows that the editing procedure depends upon the creation of a "playground" in the manuscript to be edited. The manuscript is kept on tape, disk, or in a portion of memory designated as the main manuscript area. Upon selection of a point of interest within the main manuscript, part of the main manuscript is transferred to an editorial buffer. Here an additional space, the *playground area,* is inserted between the point of interest and its adjacent character. This additional space allows characters to be inserted into that portion of the manuscript contained within the editorial buffer, and displayed on the screen.

Insertion and deletion of characters are illustrated by figures 2A and 2B.

Fig. 2. Insertion and deletion of characters

2A. Insertion

A   A¹

| 1 | | ▨ | | 3 |

CHARACTER INSERTED HERE   PLAYGROUND DECREASED IN SIZE
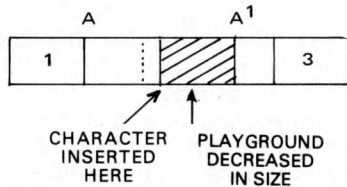
The character is inserted at point A. The playground is reduced in size by one character and point A moves one character closer to point A¹.

2B. Deletion.

A   A¹

| 1 | | ▨ | | 3 |

CHARACTER DELETED HERE   PLAYGROUND INCREASED IN SIZE
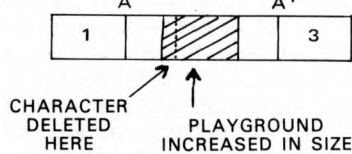
The character is deleted at point A. The playground is increased in size by one character and point A moves one character further away from point A¹. Note that insertion and deletion are compensatory in their effects on playground size.

It is important to note here that in creating the playground in the editorial buffer, adjacent blocks of characters (in this case blocks 1,3) are *also* transferred to the buffer. This was included in the original LINC program so that manuscript transfers from LINC tapes to memory would be infrequent, as the tapes were slow, even if the point of interest moved between various lines in the displayed manuscript. If floppy discs are to be used, the same approach can be employed as disc transfers are also relatively slow.

But, if this editor is to be used in a situation where the main manuscript is held in memory, then these adjacent blocks need not be loaded. In this case character transfers can occur as quickly between the main manuscript and editorial buffer, as within the buffer itself.
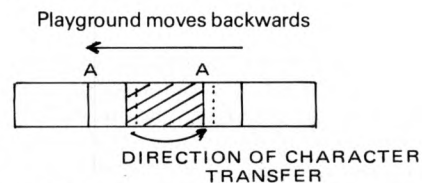
Fig. 3. Moving the point of interest

A. Forwards in the string

Playground moves forward

A   A¹

| 1 | | ▨ | | 2 |

DIRECTION OF CHARACTER TRANSFER
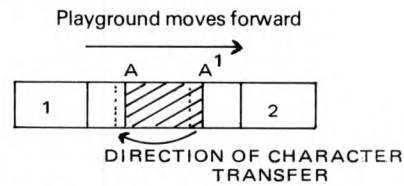
To move forward in the string, characters are transferred from point A¹ to point A. The playground does not change size but effectively moves forward.

B. Backwards in the string

Playground moves backwards

A   A

| | | ▨ | | |

DIRECTION OF CHARACTER TRANSFER

To move backwards in the string, characters are transferred from point A to point A¹. The playground remains the same size but effectively moves backwards.

Figure 3 shows how moving the point of interest (which is shown on the screen by a cursor moving through the manuscript) is achieved. It can be seen that by moving characters within the editorial buffer, the playground can effectively be inserted between any characters, so allowing extensive editing.

Should the next position of interest in the manuscript lie outside the editorial buffer, then the buffer must be reloaded. Figure 4 shows how this is accomplished moving in either direction and also, indirectly, the true power of the playground approach.

Fig. 4. Moving to characters not contained in the editorial buffer — block transfers

4A. Forwards   Main manuscript string

| φ | ┆ | | 4 |

BLOCK MOVE   BLOCK MOVE

A   A¹

EDITORIAL BUFFER | 1 | | ▨ | | ┆ |
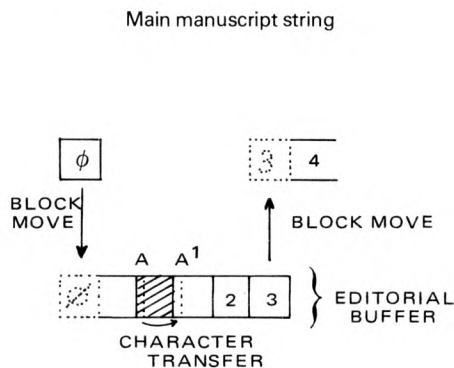
CHARACTER TRANSFER
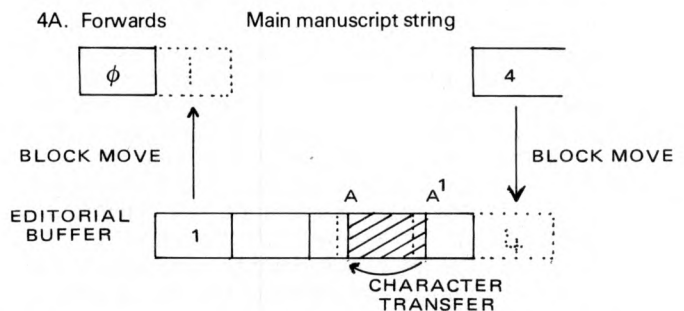
To move forwards, block 1 is transferred back to the main string and block 4 is transferred to the editorial buffer. Character transfers occur as in Fig. 3A to move the playground to the point of interest.

Note
The original boundaries of block 1 (i.e. which characters are in block 1) could change during editing. The important point to note is that the same number of characters are transferred back to the main manuscript, as are being moved to the editorial buffer from the manuscript. Any characters overwritten in the main string, will be contained in the editorial buffer. Thus the playground is continually concerned with containing characters which have been added to the string thereby increasing the string's overall length.

4B. Backwards

Main manuscript string

φ

3　4

BLOCK MOVE

BLOCK MOVE

A　A¹

2　3　} EDITORIAL BUFFER

CHARACTER TRANSFER

To move backwards, block 3 is transferred back to the main string and block Ø is transferred to the editorial buffer. Character transfers occur as in Fig. 3B to move the playground to the point of interest.

It has been stated previously that line numbers do not form part of the manuscript. The manuscript consists of a character string which contains no control information other than line terminators. Thus, the division of the manuscript into blocks defines only the number of characters, and not their functional relationship. When the transfer of block 1 back to the main manuscript occurs (fig. 4a), different characters — ie edited manuscript — may be transferred back. On examining the figures carefully we can see that any information which is transferred to the main manuscript will over-write only information which is already contained in the editorial buffer. This is so because of the way the playground area is created, and maintained. All information that is required — that is, not edited out — is retained during these transfers.

Insertions and deletions are in general compensatory in their effects on playground size. However, should there be more insertions than deletions the total manuscript must increase in length.

In fig.4a, as block 1 is transferred to the main manuscript string, block 4 is transferred to the buffer. Any increase in the text length due to "excess" insertions will be accounted for by a decreased playground area. On completion of the editing process the original manuscript will be rewritten by passing it through the editorial buffer and therefore across the playground area (fig. 4a). When all the text has been passed through the buffer and back to the main manuscript, the playground will effectively be pushed off the end of the manuscript. The important point to note here is that this total rewriting is only done at the end of editing, and not after each insertion.

The exception. One exception to this rule of rewriting only at the end of editing is when the playground area fills up. If this happens, the main manuscript is rewritten via the editorial buffer until the playground area is pushed off as before. When this is completed, the point of interest is relocated using the line counter, and a new playground is created as in fig. 1.

Wilkes has commented (1) that the minimum size of the playground area for efficient editing should be about five times the length of a typical manuscript entry. For instance, when programming in assembler language a typical entry (line length) will be about 15 characters, and so the playground area should be at least 75 characters long. Programming in Basic should require a larger playground, whereas to edit a typical machine code entry would require a smaller playground. Obviously, by choosing the larger playground all texts from machine code to Basic could be edited efficiently by the same program.

Small system power

In my opening comments I said that I would use the example of this editor algorithm to show that small systems can in fact be powerful. In the light of this editor approach let us now look at two currently available systems: the NASCOM 1 and a newcomer, the MICROS.

Both these systems use the Z-80 microprocessor, and both have a keyboard with a memory mapped VDU included in the basic kit. This configuration, built originally as an economical approach to personal computing, is ideal for this powerful and versatile editor. The Z-80 has a single instruction block move, useful in this case for buffer maintenance. The memory mapped VDU provides an exceptional editorial buffer and playground facility. Combining these two factors would alone provide the user with single character editing, of an advanced nature, with its associated ease of programme development — in the minimum system offered!

In order to achieve this with a minimum system, both the editor and the main manuscript would reside in the random access memory (RAM), whilst the *video* RAM would be the editorial buffer and playground.

As mentioned earlier, if memory is used to contain the manuscript then adjacent blocks of characters need not be loaded into the editorial buffer, thus reducing the amount of store required. In the case of video RAM, adjacent blocks of characters would be those required to display lines of manuscript, both before and after the line containing the point of interest.

In applications where large amounts of text are to be edited, the system may be expanded by adding more memory or by adding a disk. If a disk is added, then by storing the main manuscript on it a virtually unlimited length of text may be edited using only 2K of RAM. It is important to stress here that, in the context of my opening remarks, a disk is not essential to start editing. I believe that given the Z-80's powerful instruction set and the memory mapping of the video display, a good text editor plus a fair sized manuscript would fit into the 2K of memory available in these basic kits.

I hope that this article has gone some way toward showing that small well developed systems can be very powerful. I also hope that users of micro-computers in Europe will not fall for the inefficiencies found in many American systems, where in order to develop even a short program at least 8K of store is needed for the development software. It is small wonder that, with this approach to software, megabyte systems are frequently encountered.

References

1. *M.A. Wilkes,* Scroll Editing: An On-line Algorithm For Manipulating Long Character Strings; IEEE Transactions On Computers, Vol. C-19, No 11, Nov. 1970, pp1009-15.

2. *W.A. Clark* and *C.E. Molnar,* A Description Of The LINC; in, Computers in Biomedical Research, Vol 2, *R.W. Stacey* and *B. Waxman,* eds., New York, Academic Press, 1965, pp35-66.

E78
THE EUROPA BUS

A MICROPROCESSOR BUS
SPECIFICATION FOR
EUROCARD

What is a bus?

In its simplest form a computer may be considered to consist of three parts, the central processor (CPU) which does all the hard work, the memory, which is in effect a large number of individually labelled 'holes' in which data or instructions can be stored, and the input and output devices (I/O), through which it can talk to the outside world. For the system to work the CPU must be able to communicate with the I/O devices and the memory, and in some cases it will be necessary for the I/O devices to transmit data directly to and from the memory. Potentially, *individual* connections could be made between the various devices and the CPU, but this would require a lot of wires, buffers, edge connector pins and so on. A much more satisfactory solution is provided by the concept of the *bus,* which has become an essential part of microprocessor technology, both inside the devices and in functional systems. In essence a bus is a collection of wires which are connected to and shared between *all* of the subsections of the system. For a computer system the bus may be considered in three parts.

Firstly there is the *address* bus, which is used by the CPU to tell the rest of the system which item of data or which I/O device it is interested in. In simple systems this is a *uni*-directional bus, that is to say it is always "driven" by the CPU, other components simply "listening for their name to be called".

The second bus is the *data* bus. This is used to transmit data around the system, and is therefore a *bi*-directional bus. It is driven by the CPU when it is sending data *to* the memory ('writing'), while it will be driven by a memory or I/O card when the CPU is asking *for* data ('reading').

The final ingredient necessary to get all this to work properly is a set of signals, known as the *control* bus, to make sure that everyone knows who's doing *what* and *when.* As an example, the four most important control signals in the bus defined here are MREQ, IORQ, RD and WR, all of which will normally be driven by the CPU. MREQ tells the system that the CPU wants to talk to the memory location corresponding to the address on the address bus. IORQ similarly shows interest in an I/O device (obviously MREQ and IORQ should not *both* be active at the same time). RD tells the memory or I/O device that the CPU wants it to put its

Bob Cottis
Pippins, Boulters Lane,
Maidenhead, Berks.

Dave Howland
186, Courthouse Road,
Maidenhead, Berks.

Pat Crowe
22, Ringsbury Close,
Purton, nr. Swindon,
Wilts.

current data onto the data bus. Similarly WR tells the addressed memory or I/O device that the CPU has put new data for it onto the data bus.

Why do we need another standard?

There are many ways in which a microprocessor bus can be controlled. For example, 6800 based systems often combine the functions of RD and WR defined above into a single signal. This tells memory whether to read or write when MREQ goes active. The objective of any bus specification is to define a mechanical and electrical system such that printed circuit cards can be produced to run in any system without problems. The two buses which have achieved widespread use are the S-100, specified in 1974 by MITS for their Altair computers, and the S-50, defined by South-West Technical Products Corporation for their 6800 based machines. A large number of standard parts are available for these buses, so why do we need another standard?

The main disadvantage of both these buses is that they were designed around one *particular* processor, and all the quirks of these machines have been built in. It is possible to squeeze other processors onto the S-100 and S-50 buses, but this inevitably involves compromises. Anyone who has seen the "trailers" gradually being released for the new breed of 16-bit microprocessors due out this year will be aware that the amateur is very soon going to have access to micros with the power of today's minis. These

devices can probably be accommodated on the 8-bit S-100 and S-50 buses, by transferring two 8-bit bytes for each 16 bit word, but this will *slow the processor down by a factor of two.* Additionally, without extra hardware the standard buses will only accommodate 64K bytes of memory, whereas the Intel 8086 is reputed to be able to address 1M bytes, and the Zilog Z8000 8M bytes! At present the cost of filling 64K bytes may seem sufficient to make this problem rather irrelevant, but a price of £4 per chip does not seem unreasonable for 16K dynamic RAMS in a few years' time, which will change things somewhat.

Thus the primary aim of this specification is to define a system based on readily available components, which will have sufficient generality to accept all current processors, and hopefully also the devices which wil l be available in the near future (in 15 years we probably won't need a bus, just take a 40 pin chip, connect to + 5V, a keyboard and a TV and you'll have the power of an IBM 370 at your fingertips).

Objectives
When the outline of this definition was first proposed, it was suggested that it should: —

1) Use readily available equipment practice (boards, card-frames, connectors etc.), and allow easy manufacture of printed circuit boards by the individual if required.
2) Be sufficiently flexible in definition to allow its use with any existing (and hopefully future) 8 bit microprocessor without penalizing any one by requiring complex logic to interface with the bus.
3) Be sufficiently flexible to allow good possibilities for implementing personal experimental or non-standard systems without departing from the framework of the standard.
4) Be flexible enough to make it possible to produce equipment to full professional standards while at the same time not penalizing the experimenter with a low budget.

Subsequently it became clear that many more 16-bit processors would soon be available at reasonable prices, and the bus definition was modified to allow for 16 data lines. This may at first seem something of a waste of 8 valuable pins for people using 8 bit machines, but it should make it possible to switch to a 16 bit machine with very little difficulty. Devices such as memory mapped VDUs and cassette interfaces can be used directly by 16 bit machines, simply by writing the software to ignore the top 8 bits. Memory cards can be used in pairs by assigning one to the bottom byte, and the other to the top.

Boards
The board size selected is that known as Double Eurocard. The dimensions of this board are shown in fig. 1. The Eurocard is rapidly becoming the European standard, and even the U.S. is beginning to succumb to its Teutonic charm. Consequently equipment such as racks, cases, prototyping cards and connectors are readily available from many sources. The smaller Single Eurocard can be used with the same connectors and bus convention, but it is not recommended, as considerably less than half as much useful logic can be fitted on the board, for the same expenditure for connectors and bus buffering.
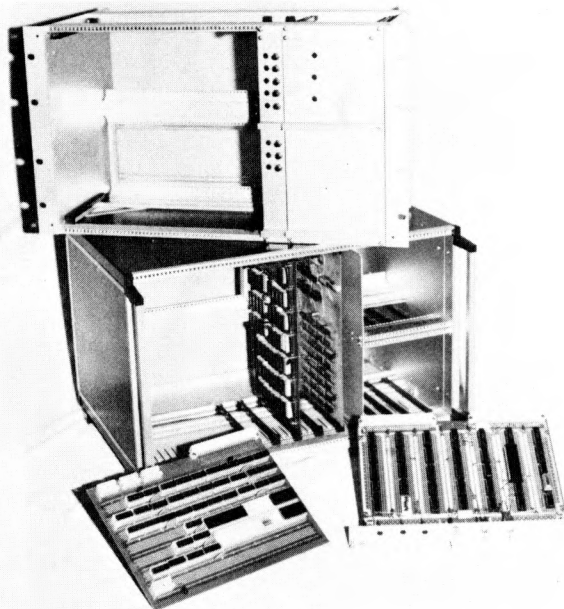
Connector
Eurocards can be used with standard edge connectors *or* with two-part 'indirect' connectors. The two-part, 64-way connectors have been specified for the following reasons,

1) They are highly reliable and, in the unlikely event of damage, replaceable.
2) They make it possible for the amateur to produce reliable printed circuit boards without needing access to gold-plating equipment.
3) They are not all that cheap (about £5 per pair), but the circuit boards will cost a pound or two less because they do not require to be gold plated.
4) They appear to be the "industry standard" connector for Eurocards, and are readily available.

One 64-way connector provides just enough pins for the bus. The other connector is optional, and may be used for input and output connections, or for any other functions the user wishes. Memory cards will normally be fitted with only one connector.

Two versions of the 64-way connector are available, referred to as 64/64 and 64/96. The first number refers to the number of pins actually fitted to the connector, while the second number is the number of pin positions in the connector housing. The most significant difference between the two versions is the spacing between the two rows of 32 pins each. This is 0.1 inch for the 64/64, and 0.2 inch for the 64/96. This makes it much easier to produce a printed circuit backplane for the 64/96 connectors, and for this reason they are preferred. However printed circuit boards can and should be designed to accept either connector (see fig. 2).

Viewing the board from the component side, with the connectors on the right, the bottom connector is the bus connector. The connectors are mounted on the component side of the board.

Electrical
The pin assignments for the various bus signals are given in Table 1.

Five basic signal types are used,

1) **Power Supplies** Two pins are used for each power supply (four for ground). This permits printed circuit backplanes to be designed with fairly solid power conductor traces. The connector rating is 2A per pin, so the maximum current which should be drawn by any one supply of a card is 4A. Ground and +5V are situated at the ends of the connector so that the traces for these supplies can be made a lot wider.

2) **Tri-state Lines** Used for synchronous signals which may be driven from several sources. May be driven by LS, standard or Shottky TTL buffers. LS buffers are recommended for most systems. Each card should preferably present only one LS load to the bus.

The Bus Signals

1) **A0 to A15** These constitute the address bus. A15 is the most significant bit.

2) **D0 to D15** These constitute the data bus. D15 is the most significant bit. 16 bits have been specified to allow for existing and future 16 bit machines. 8 bit systems can use D8 to D15 for other functions, but bear in mind that this may make it difficult to change to a 16 bit processor.

3) **BAI, BAO, IEI, IEO** These signals are to be used for daisy-chained signals (i.e. IEO from one card goes to IEI on the next). These particular signals are for the Z80, being used to control bus access priority during DMA, and to control interrupt priority. Thus, when IEI to a
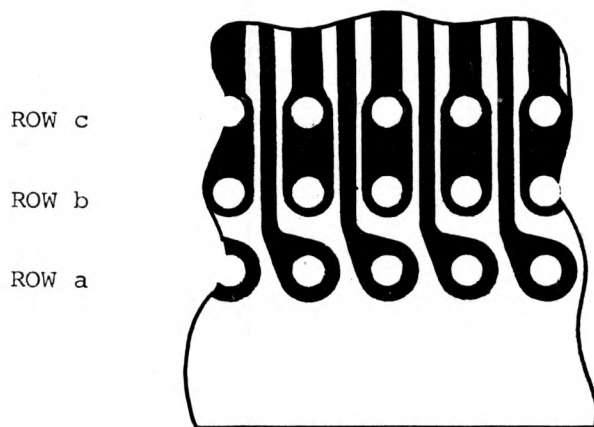

Fig. 1 Double Eurocard

Fig. 2 Pad Configuration to Accept 64/64 and 64/96 Connectors

3) **Daisy Chains** See below for more details on how these work. On cards not using these signals the in and out pins should be connected together. On other cards the output should be driven by one standard TTL output or the equivalent. Inputs should be TTL compatible.

4) **Lines Driven From One Source Only** Used for the system clock(s). Driven by permanently enabled tri-state buffers or the equivalent.

5) **Open-collector Lines** Used for asynchronous signals which can be driven from more than one source. Should be driven by devices capable of sinking at least 16mA.

Signal Polarity
As noted above all signals should be negative true. This means that a '1' on the address or data buses, or an active control signal will be signified by a TTL low voltage. This convention is indicated by the *bars over the signal names.* This convention should be strictly adhered to for control signals as it ensures that they will be pulled up to an inactive state when the control lines are not being actively driven.

For the address and data signals the use of a negative true convention means that cheap inverters such as 74LS04 or 74LS14 will give positive true signals on PROM and I/O cards, which will make life somewhat easier. In general however the individual will experience very few problems if a positive true convention is preferred for the address and data signals.

card is low any interrupt logic on the card will know that no higher priority interrupts are pending, and it will generate an interrupt if required. At the same time IEO will go high to tell lower priority cards that its interrupt is pending. Any system requiring daisy-chained lines should use these pins. These signals will not normally be used by memory cards, and these four pins may be used to extend the addressing range to 1Mbyte by using a 'split'

bus, with memory in one section and peripherals in the other.

4) **Clocks** These signals allow for two system clocks. For machines which generate a signal to indicate that the first byte of an instruction is being fetched, this should be placed on pin 15a. Otherwise this pin can be used for the less important system clock in a two-phase system, or for any other function. Pin 15b/c should be used for the most important system clock (i.e. ø2 for the 6800 and 8080).

5) **DWAIT** This signal is used to hold the processor via the WAIT line, and disable the CPU card tri-state buffers. This signal may be used by a single card dynamic RAM controller to provide for dynamic RAM refresh, often invisibly. It is also necessary for the implementation of virtual memory (there is not space here to describe this technique, see BYTE Vol. 2 No. 7 for an introduction).

Table 1. Bus Signals

Pin	Row a Signal	Type	Note	Row b/c Signal	Type	Note
1	Ground	1		Ground	1	
2	Ground	1		Ground	1	
3	+12V	1		+12V	1	
4	-12V	1		-12V	1	
5	A0	2	1	A1	2	
6	A2	2		A3	2	
7	A4	2		A5	2	
8	A6	2		A7	2	
9	A8	2		A9	2	
10	A10	2		A11	2	
11	A12	2		A13	2	
12	A14	2		A15	2	
13	BAI (A16)	3,2	3	BAO (A17)	3,2	3
14	IEI (A18)	3,2	3	IEO (A19)	3,2	3
15	MI (ø2)	2,4	4	ø	2	4
16	DWAIT	5	5	INTAK	2	6
17	DMAR	5	7	DMAG	4	7
18	NMI	5	8	INT	5	8
19	MREQ	2	9	WR	2	11
20	RESET	5	13	WAIT	5	14
21	IORQ	2	10	RD	2	12
22	INHR	5	15	RFSH	2	16
23	D0	2		D1	2	
24	D2	2	2	D3	2	
25	D4	2		D5	2	
26	D6	2		D7	2	
27	D8	2		D9	2	
28	D10	2		D11	2	
29	D12	2		D13	2	
30	D14	2		D15	2	
31	-5V	1		-5V	1	
32	+5V	1		+5V	1	

6) **INTAK** This signal is generated by the CPU to indicate that it has accepted an interrupt.

7 **DMAR and DMAG** These signals are used to control bus access by peripheral devices using direct memory access (DMA). DMAR is generated by the peripheral to indicate that it wants to use the bus, while DMAG is used by the CPU to signify that the request has been granted.

8) **NMI and INT** Non-maskable and standard interrupt request inputs. The method of response to an interrupt will inevitably be processor dependent, but if possible it is suggested that the CPU should be configurable to respond to an interrupt consisting solely of INT going low until INTAK goes low. This will permit peripheral cards requiring single level interrupts to be produced in a non-processor dependent form.

9) **MREQ** This signal signifies that a memory access is in progress and that the address bus carries a valid address. For the Z80 MREQ can be used directly, while for the 6800 the suggested logic is (VMA. ø2.IOBLOCK), where IOBLOCK indicates that a memory mapped input-output block of 256 bytes is being accessed.

10) **IORQ** This signifies that an I/O access is being performed, and that the lower byte of the address bus carries a valid I/O address. For the Z80 IORQ can be used directly, while the suggested logic for the 6800 is VMA. ø2. IOBLOCK.

11) **WR** Write strobe, indicating that the data bus carries valid data to be written to memory or I/O. For the Z80 WR may be used directly, while the suggested logic for the 6800 is VMA. ø2.R/W.

12) **RD** Read strobe, indicating that the addressed memory or peripheral should place data on the data bus. For the Z80 RD may be used directly, while for the 6800 VMA. ø2.R/W may be used.

13) **RESET** Resets processor and peripherals.

14) **WAIT** Used by slow memories and peripherals to delay processor read or write cycles. The WAIT line should not normally be held low for extended periods as this could cause refresh problems for dynamic memories and processors such as the 6800.

15) **INHR** Used to inhibit reads from RAM boards to allow 'shadow' PROMS to be temporarily placed at the same memory locations during system initialization. Does not prevent writing to RAM.

16) **RFSH** This signifies that the lower order address bits carry a valid refresh address for use by dynamic memories.RFSH.MREQ may be used for the Z80, while a separate controller will be required for the 6800.

Control logic for specific processors

It can be seen that the above specification uses control signals which are similar to those for the Z80. There are several reasons for this, the major one being that of adaptability. Memory boards will normally use MREQ to gate the selection of memory chips, RD to enable the data bus buffers onto the system data bus, and WR to write to the selected chip(s). Using this convention the actual timings for MREQ, IORQ, RD and WR will not be critical to the logic of the memory card, providing the address is valid throughout MREQ pt IORQ, the data is valid throughout WR, and the CPU does not try to drive the data bus during RD. To give a safety margin, address and data buses should be valid for at least 50ns before and after the relevant control signals are active. Other schemes which effectively use only two signals to access memory (e.g. VMA and R/W for the 6800 or MRD and MWR for the 8080) cause considerably more problems, particularly for processors, such as the Z80, which generate WR late in the memory access cycle.

Despite this similarity to the Z80, it must be emphasized that this is not just intended to be a Z80 bus. The detailed logic has been worked out for the 6800 and the Z80, and it has proved possible to make them sufficiently compatible that static memory cards and simple I/O can be directly interchangeable. The detailed logic for other processors has not yet been defined, as the authors of this standard do not have the first-hand knowledge necessary to define the control logic in the optimum way. We have however checked that other processors can, as far as we are aware, be used with this bus structure without serious problems. The second connector can be used for additional signals such as the CRU functions for the 9900 (if they can't be fitted into the bus itself).

It is hoped that user's groups for these other processors will draw up their own definitions for suitable logic. The authors of this standard would be pleased to give any assistance possible in the formulation of thses definitions.

We would also be pleased to receive comments, criticism and suggestions for improvement, either directly, or through the pages of **PCW**.

Products Page

KIM-1, America's most popular low cost microcomputer system card, is now available from PELCO (ELECTRONICS) LTD., Enterprise House, 83/85 Western Road, Hove, Sussex (telephone: (0273) 722155), at £185,00, plus V.A.T.

The product is ideal for the enthusiast who wishes to get down to the real business of microprocessor design, which is the software. The Card comes built and tested, and only requires the addition of a 5 volt power supply, and the user can immediately start programming. Based on the third generation 6500 microprocessor, which itself was developed from the popular 6800 product line, all the powerful software and hardware features associated with this processor are available to the user. In addition, the Card comes complete with interface circuiting to enable a simple connection to a low cost audio cassette player, which can then be used to store user programmes.

The Card also contains 15 additional I/O lines for connection to external circuiting, as well as a purpose built teletype interface for engineers with such facilities.

The basic Card uses a ROM-resident editor programme with de-bug facilities such as single-step ability built in. With 1K bytes of RAM easily expandable to 4K bytes, and with the ability to expand to a full 65K bytes if required, the uses of the product are virtually limitless as far as current microprocessors go. Support for the product is available both from the various 'user' clubs and publications, and from Pelco (Electronics) Ltd., who are Rockwell's Distributors and Representatives in the U.K. Pelco offer full engineering back-up facilities to all users of the wide variety of Rockwell microprocessors from their laboratories located in Hove.
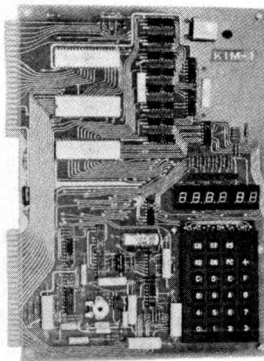
THE MICRONICS COMPANY have released the British designed and assembled MICROS microcomputer system. The price of £550 includes a Z 80 cpu, debug and monitor ROM, 2K bytes RAM, audio cassette interface and checksum error detector, video monitor interface, UHF TV modulator, 53 key contactless ASCII keyboard, power supplies and cabinet. RS232 and 20 MA current loop outputs are provided for a teletype or a matrix printer and the RAM is expandable up to 64K bytes. Options under development include a minifloppy disc, BASIC compiler in ROM and cassette, NPL time decoder, TELETEXT decoder and a selection of cassette games.

The system has been designed to meet the needs of educational establishments, personal computing and small business users at a realistic cost and is also available in kit form. A data sheet is available on request and enquiries from common market and overseas agencies would be welcome.

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

The introduction of the Equinox 100 computer system at the Mini-Micro Exhibition, held recently at the U.S. Government Trade Center in London makes available, for under £3,000, a system based on the popular 8080 or Z80 microprocessors with 32K bytes of read/write memory and dual floppy disc drives.

The Equinox system is based on the industry standard S-100 bus, which supports the largest variety of plug-compatible boards and peripherals available. More S-100 bus based computers are currently shipped than any other system, with an estimate of over 50 manufacturers in the U.S. making compatible boards. These include CPU boards based on the Intel 8080, Intel 8085, Zilog Z80, and the 16-bit Western Digital MPU, as well as a wide range of memory boards. In addition, there are digital to analogue converters, speech synthesisers, speech recognisers and a variety of other interface boards, all making computing power generally available at minimal cost.

The special features of the Equinox include a constant voltage power supply, to overcome fluctuations in line voltage, and the Wunderbuss motherboard, designed by the famous American electronics engineer, George Morrow. This busboard has both full shielding and active termination to keep signals "textbook clean". The size of the mainframe permits expansion to a low cost multi-user, multi-tasking, time sharing system.

Further information is available from Interam Computer Systems Ltd., 59 Moreton Street, Victoria, London SW1. (01-834 0261/2733)

AMI-COS Microprocessor System

In conjunction with INCAA Computer Design Bureau RITRO offers the modular AMICOS microprocessor system. Other currently available systems do not have, in general, universal extension possibilities, whereas the AMICOS system can be configured to the users' exact requirements, thanks to modular design and an effective bus structure.

AMICOS is based on the well known S6800 family of American Microsystems Inc. and is offered in kit-form, with PC boards on Eurocard format. Each module comes with extensive documentation on assembly and applications; and with the first module RITRO supplies an attractive binder with data on the S6800 and general information on AMICOS.

Modules Available or Planned for the Future

* THE CENTRAL PROCESSING UNIT, with a fully buffered S6800 MPU, clock circuitry and memory.
* A CONSOLE DRIVER with operating system in ROM. This module includes switches, pushbuttons and LED's, forming a binary console. With the above two modules the user may begin to get acquainted with microprocessors and study

the instruction set by running small programs via the console.

* THE CPU CONTROL CARD. Through this module the CPU selects the desired I/O card and handles the various interrupts.
* 8-bit parallel I/O module with PIA and decoding.
* Problem oriented modules such as model train controlling, morse generation and decoder, A to D converter, TV-driver, audio-cassette driver, etc.,
* Memory extension (RAM & E-PROM).
* Various output cards (opto-couplers, relays).

Details from RITRO Electronics (UK) Ltd., Grenfell Place, Maidenhead, Berkshire.

This portable video terminal produced by Pronto Electronic Systems, offers a cost effective alternative to the standard teletype, by using the latest microprocessor technology. It gives the computer system or microprocessor user a no-fuss method of data entry at a fraction of the cost normally encountered.

The Model 601A is built into an attractive executive-styled carrying case, measuring only 16½" x 12½" x 5". It has been designed to interface with a video monitor or modified home TV, and the microprocessor or computer system being used via a 20mA loop (or RS232 option).

Applications for the Model 601A are extremely varied, but one important area is as a low cost interface to microprocessor and computer systems, and for educational and commercial applications. It also offers the opportunity for establishing remote terminal or message display facilities at such locations as conferences, exhibitions, business meetings or even in the home.

Further details: Pronto Electronic Systems Ltd. 645-647 High Rd, Seven Kings, Essex IG3 8RA.

Tap-a-dip

New from the American manufacturers of EZ Hooks is a test clip to simplify testing on dual inline packages — the TAP-A-DIP.

TAP-A-DIP is presented in 14-line and 16-line models. Critical width does not change during insertion and removal, making TAP-A-DIP ideal for high density boards. End contacts lock firmly under each of the end leads, and by this strength TAP-A-DIP is able to double as an insertion and removal tool. One piece square contacts extend above the top of TAP-A-DIP to enable EZ Hooks to be easily connected.

As with all EZ Hooks, TAP-A-DIP is represented in the UK by Special Products Division of British Central Electrical Co. Ltd., New Street, Ringwood,
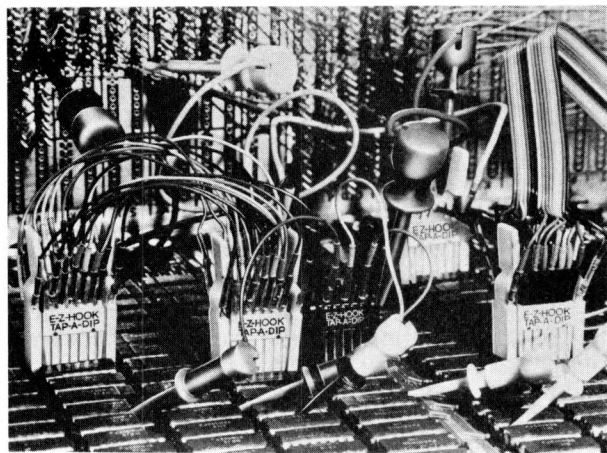
Hants., from whom further information may be obtained.

A step-by-step approach to mastering micro-processors, by ERA

ERA has introduced the MP Experimenter, a training course utilising a simple but highly effective step-by-step procedure which commences with an explanation of basic logic elements and gradually builds up to an appreciation of a full microcomputer.

Central to the course is a portable desk-top microcomputer unit incorporating a control panel which is used in conjunction with a set of overlays designed to guide the student through each teaching step. The system includes an operating manual and a five-volume set of course notes containing a wide selection of practical experiments. This enables the user to teach himself at his own pace in the home, office or laboratory.

Aimed primarily at technologists and engineers, the system will also be invaluable to managers and executives who have perhaps been overtaken by this fast moving technology and can now bring themselves up-to-date using a method which fits in with their busy day-to-day schedules.
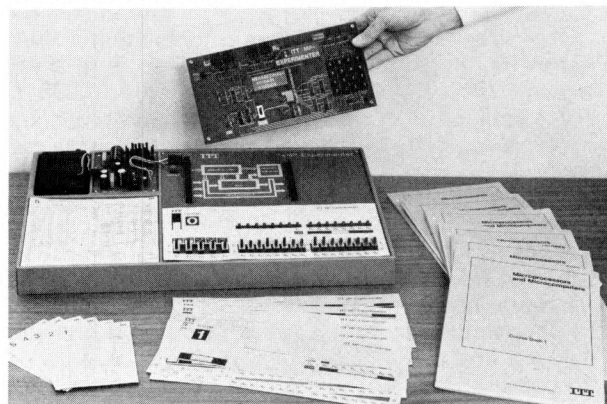
The MP Experimenter hardware is programmed to simulate each part of a microprocessor in turn, providing the following functions: —

* An 8 bit adder/subtractor
* An 8 bit arithmetic logic unit (ALU)
* An accumulator
* An accumulator with data memory
* A simple microcomputer
* A Hypothetical microcomputer
* An 8080 microprocessor system with a monitor

**Further information from:
Chris Nabavi, Computers and Automation , ERA Limited, Cleeve Road, Leatherhead, Surrey, KT22 7SA. Telephone: Leatherhead (03723) 74151 or Hugh Cullimore Worplesdon (048631) 2729.**

Gamma Computer Products, a new member company of the Nottingham-based Gamma Associates Group, has introduced the Gamma Proteus Compact (GP — Compact) microcomputer system.

The powerful desk top system which is making its worldwide debut in the UK market is completely portable. It incorporates a 64K byte DEC LS1/11 cpu, a Burroughs "Self Scan" Plasma Display, a Honeywell "Microswitch" keyboard, a Pertec "Microfloppy" floppy Disk and a Centronics Series 700 printer. The total product comes from the General Robotics Corporation of Hartford, Wisconsin.

Price of the 64K byte system — the maximum memory that the DEC LS1/11 will support — plus three floppy drives is £7,500 — one off — with a full software licence. Susbstantial quantity and OEM discounts are available which can reduce the price to almost the £5,000 level.

The complete sytem is extremely compact in size, being 41 cms high, 53 cms wide and 61 cms in depth. It weighs under 22 kgs.

The new system offers users all the software facilities of the DEC LS1/11 system, including BASIC, a Fortran IV compiler and APL. It also offers TSX — a timesharing executive which allows a number of users to timeshare the background job facilities of the RT-11 operating system. The system will also run RPL — the PDP-11 version of the National Computing Centre Filetab language.

The company also expects to sell to OEMs who will be able to tailor systems for small service operations, such as accountants, solicitors, architects, consultants and small distributors and manufacturing companies.

DECstation 78 from Digital Equipment Co. Ltd., Digital House, Kings Rd. Reading, Berks. DECstation 78 is the new LSI version of Digital Equipment's famous 12-bit PDP-8 family of computers.

The compact, inexpensive DEC station 78 has been designed as an effective, problem-solving tool for a broad range of user environments — everything from personal computer and fixed function applications to real-time, multitasking operations and networking.

The **PCM-12** is an inexpensive small computer that is fully software-compatible with the Digital Equipment Corporation PDP-8 family of minicomputers.

The PCM-12 is designed around the Intersil IM6100 microprocessor, a device which is software-compatible with the most popular minicomputer ever developed — DEC's PDP-8. That means that the PCM-12 can execute nearly all PDP-8 software, without modification. And that includes assemblers, editors, debug routines, interactive languages like BASIC and FOCAL and scientific languages like FORTRAN. With the addition of a floppy disk or other mass-storage device, the PCM-12 is even compatible with DEC's sophisticated OS-8 operating system.

Largely because of its PDP-8 compatibility, the PCM-12 has already been widely accepted into OEM, educational, scientific and home computing applications. PCM-12's are being used today in business data processing; machine tool control; national defense applications; energy development; physiological, geological, chemical, vibration, laser, and medical experimentation; dedicated IM6100 microprocessor system development; and home computer games.
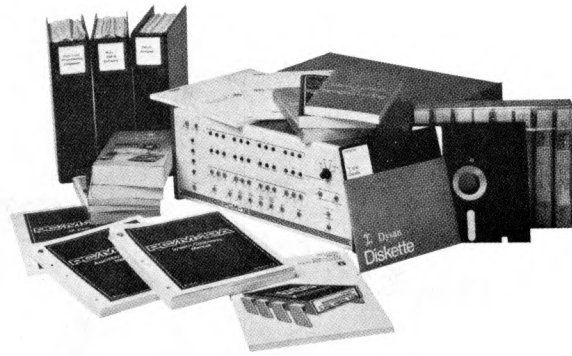
The IM6100 microprocessor is essentially an LSI version of the CPU in the PDP-8/E, the latest version of the classic PDP-8. This chip makes it possble for the PCM-12 to perform like a PDP-8/E in nearly every respect — and even provide features that the '8 lacks. Features like a separate control panel memory which makes the entire main memory available for the user's program. And a built-in binary bootstrap loader — you simply flick a front panel switch to load a machine language program from paper or magnetic tape. The PCM-12 control panel provides essentially all PDP-8/E functions, plus single-clock and decrement-address functions.

With provision in its design for vectored priority interrupts and direct memory access, the PCM-12 is a true minicomputer designed around a powerful microprocessor. The machine's bus-oriented architecture provides 15 identically operating card slots — enough room for up to 32K words of memory and a full complement of peripheral interfaces.

The PCM-12 is a solidly-designed general purpose computer that can serve as the heart of the simplest or the most sophisticated small computing system.

Electronic secretary of the year
The new Mind Reader personal computer is from Optimisation, 45 South St. Bishop's Stortford, Herts. It plans, files and reminds. No more missed appointments, forgotten birthdays, double luncheon dates and late meetings. Its computer memory solves all these problems and further uses its advanced electronic technology to display the time and date. It's also a stop-watch, electronic calendar and two-time zone clock and calendar.

The Mind Reader will manage appointments schedules and maintains four categories in which to file special messages, displaying them at the appropriate date and time.

It will also display a daily schedule — today, tomorrow or any other day. At the touch of a key 'things to do' are on file for each day. The Mind Reader will retail at around £300.
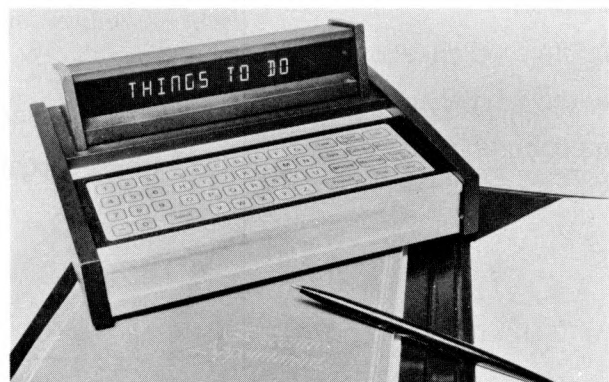
The Exhibition with the difference

PERSONAL COMPUTER WORLD SHOW

AT THE WEST CENTRE HOTEL LONDON S.W.6

ON 21, 22, 23 SEPTEMBER 1978

FEATURING

Systems for small businesses

Computers in Education

Computers in the home
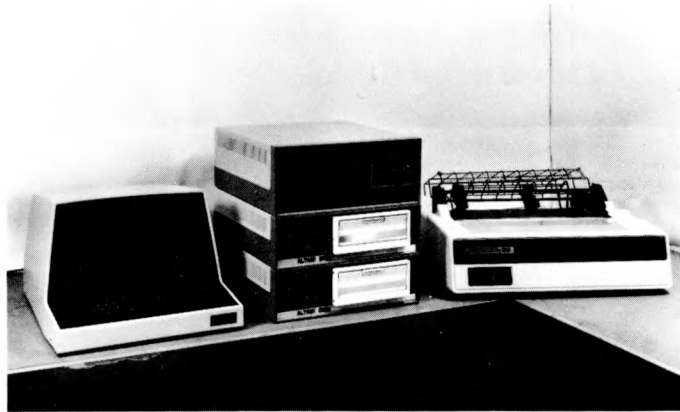
Amateur computing

ATTRACTIONS

Competitions

Best home brew system

Best software

Best school applications

Best home applications

PLUS

Famous Personalities

Innovators corner

Video games

PCW Micro chess championship

Seminars, PCW consultants in attendance