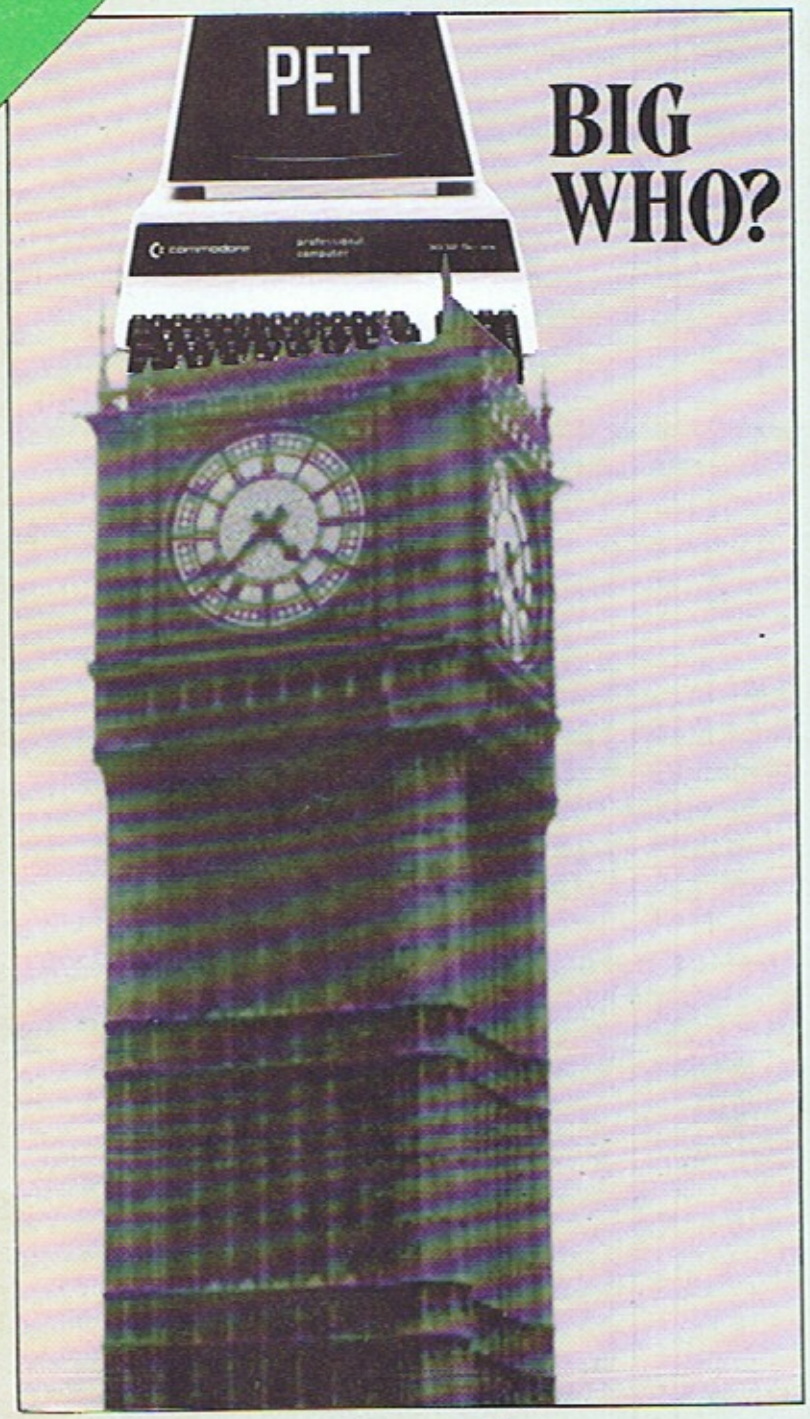
COMMUNICATIONS SPECIAL FEATURE

COMMODORE CLUB NEWS

August 1981



Volume 3 Issue 7

BUTTERFIELD

Start of a special monthly column from PET genius Jim Butterfield

REVIEWS SECTION

Visicalc, Wordpro 4+, and more, get the indepth treatment

NEW PRODUCTS

Up to the minute information

MACHINE CODE

Start of a special course for beginners

COMMUNICATIONS FEATURE

"Using a microcomputer as an Interactive Terminal"

commodore COMPUTER

Contents

Communications

Special centre supplement on communications, featuring an extensive article on the use of a microcomputer as an interactive terminal.....1/12

Next Month's Issue.

Next month's issue will be featuring the usual Jim Butterfield column, machine code for beginners part two, disk drive beginners course part 5, and all the other regulars.

New items include "What the Papers Say", a look around what the rest of the press is saying about us, lots of technical info on the 8024 for those of you with such beasts, an in-depth report on the MMF9000, Dave Middleton on a day trip to Paris, me on a day trip to Ealing (we editors get all the luck....) with the B.B.C., plus many program gems in both Basic and machine code, special hints for those of you with cassette decks, and much, much more.

Special section is back on education this time, with the emphasis very much on the VIC, including many sample programs, hints and news from Paul Higginbottom, software genius. I know many of you will not have VICs yet, but you will, you will!

Editorial

I think I'm slowly getting the hang of this! My first newsletter was done in a bit of a rush, the second one had to be about the PET Show, but with the third one last month I was finally happy – a mixture of 'meaty' articles and programs, quite large in size, and overall a cosmetically pleasing appearance. Thanks to everyone who contributes articles, snippets of information and programs that you've developed which will be of help to others – thank you one and all.

The PET Show

The PET Show. Well, I thought it was a great success, and so, it seemed, did everyone who attended. A lot of good has come from the Show – lots of orders placed, exciting possibilities for distribution of your newsletter overseas, a general feeling that Commodore cares for its' users – and for me personally a great chance to meet a lot of you first hand, rather than via correspondence. Also, there's a liaison beginning to be built up between the Commodore Users Group and the independent groups, something that was put under a bit of a strain in the early months of the year. Thanks must go in particular to Mick Ryan of I.P.U.G. for his help on this – we'll be sorting out all the finances, free disks etc. as soon as possible.

A number of new products were unveiled at the Show, including the micro-mainframe, the 8096, the VIC of course, the cash register, and the single disk drive appeared on the Saturday, provisionally numbered the 2031. We'll be carrying as much information on these as possible, when it appears. In this issue we're featuring the VIC memory maps (courtesy of, guess who? – Jim Butterfield of course!), a report on the micro-mainframe which goes into rather more detail than the original press release featured in Vol.3 Issue 4.

As usual, I must stress that you don't keep trying to find information on these elsewhere, we'll publish everything as and when we get it.

Editor Pete Gerrard

Jim Butterfield

Another great thing about the Show was the promise of a series of monthly articles from the one and only Jim Butterfield. The first one appears in this month's newsletter, and will be featured regularly from now on. Jim has probably more experience in the world of computers in general and the PET family in particular than anyone else on the planet – these articles will make very interesting reading (and, at Jim's re-request, only in OFFICIAL Commodore publications). Having talked to him at the Show, Jim proved himself to be a very amiable and knowledgeable person – I'm glad to have him aboard. Also, to User groups, we now have Adventure for the 8032 and 8050 – if you want a copy, write and let me know and I'll send you one as soon as possible.

Communications

This month's pull-out feature is on communications – something which is gaining importance in the PET arena, as could be gauged by the phenomenal attendance that we had at the communications part of the PET Show. The article is by Dr. Philip Barker, principal lecturer in the department of Computer Sciences at Teeside Polytechnic. Titled "Using a Microcomputer as an Interactive Terminal", it goes into great detail about the PET in this particular role, and gained the seal of approval from Rod Wellburn, Commodore's communications manager. Dr. Barker promises a "sequel" to this, entitled "Algorithms for Intelligent Terminal Operation". We'll publish this as another pull-out as and when it arrives.

Other articles this time around include one from the prolific Dr. Price of the Liverpool Polytechnic on passing parameters in sub-routines, David Pocock continuing his journeys into the inner workings of disc drives, a number of small programs to help you out in your own programs, plus SUPERMON 4 – add more commands to the existing PET monitor for your 4032 or 8032.

Ever wondered what happens to equipment when it arrives in the U.K. from the States? Read inside and find out in an article from Don Goergen, the man in charge of the Q.C. area – if it wasn't for them you wouldn't get any PETs.

Machine Code for Beginners

There have been a number of requests for articles on machine code programming for beginners – as you may know we started such a series a while ago, written by Paul Higginbottom, but we ran into a snag. Paul emigrated to Canada! So, via the American newsletter we have an alternative source for this series, namely Emily Berk. For the benefit of everybody I've started Emily's series of articles from the beginning – some of you will already know most of the information contained in this first article, some of you will not, but I think that to get the best benefit from them it's better that we don't jump in at the middle but start at the beginning and follow it through from there. I hope you find this series useful and informative.

What else have we in store for you? Barry Miles returns to the fold, reviewing in great depth Wordpro 4, with Wordcraft getting a similar treatment next month. There have been in the past a number of articles on these particular two programs, but as they are the best around, and as newcomers keep coming into the land of micros, I felt it about time to give them a really solid review. Being Approved Products, it also ties in nicely with our philosophy of trying to review an A.P. every month. Visical also comes under the hammer this time.

Second-Hand Equipment

A new feature in this month's issue is the advertising of second-hand equipment from User Club members who've upgraded etc. and who now have equipment to sell. Advertising in this is of course free, so if you're in a position to sell, write and let me know. If you're interested in buying, don't write to me but **write direct to the people in the adverts.** Hopefully this will grow into an established area to purchase second-hand PET-related goodies.

Special Offers

As the User Club continues to grow, we can start to do more and more special offers, like the above for instance, to you – details will appear in future months as we expand to make this into a true club – your club. At the moment we've increased the quality and content of this newsletter to give you true value for money for your subscriptions, and now we want to make available to you a far better service for the same amount of money. Watch this space!

My policies with regard to publication material. I mentioned something last time about the kind of material we're after. Feel free to send in anything that you have, with a view to publication. My constraints on what I can and cannot publish are not that strong – suffice it to say that if you send it in, I'll take a very serious look at it. Contributions can be written in any shape or form – quill, blood, wordpro or wordcraft files, or anything else that you might think of.

Finally, another policy matter. I've been publishing pictures of the

various contributors from Commodore. I think this is important – so many of you ring Commodore up about one thing or another, and although you know the name, very rarely do you know the face. So it's nice to fit the face to the name. I think so anyway!

The address to send any material to is:-

The Editor
Commodore Business Machines
818 Leigh Road
Trading Estate
Slough
Berks

Tel. Slough (STD 0753) 74111.

For new subscribers, or if you're renewing your subscription, write to Margaret Gulliford at the same address.

As a footnote, I've said all along that this is a monthly newsletter. Producing it to this time schedule means that I don't get any holidays! So, this newsletter has arrived fairly closely on the heels of the issue in the second week of July, and the next one will be appearing in early to mid-September, while I spend two weeks in August walking along Offa's dyke from North to South Wales. Should produce a few interesting pictures for future newsletters!

PETs in Space II

Following on from an earlier article I received this letter:

Dear Pete,

You mention matrix in your article PETs in space in the June issue of Club News. Incidentally Michael Erlewine, who writes all the matrix astrological software, is a Commodore dealer and so is able to tailor software to particular PETs. His English outlet is through Roger Elliot the astrologer who writes a column for TV Times. His address is: The Manor, Cossington, Bridgewater, Somerset TA7 8JR.

I am the treasurer of the Faculty of Astrological Studies, probably the largest astrological teaching organisation in the world. We have correspondence courses for beginners and for advanced students and run many other activities such as evening classes in London. We bought an 8032 in April and I am in the process of learning how to use it to handle our mailing lists, calculate astrological information and keep our cashbook. Later on we hope to begin to update our correspondence course material using word processing.

The only other source of astrological software for micros that I have come across is a firm called I.M.S. at P.O. Box 1032, Ashland, Oregon 97520, USA. These are much faster programmes as they are written in machine code, but I.M.S. tell me that they only run on CP/M machines. If there is such a thing as a CP/M adaptor for PETs available, then I would be grateful if you would let me know who makes it.

All the best.

Jonathan Powell

User Clubs

User Clubs watch out – we're going to be starting a much closer working relationship with you in the not too distant future, and as I said in the editorial great thanks go to Mick Ryan of I.P.U.G. for all his help in this. I hope to publish details of what precisely we're going to do in the next newsletter (out around mid-September – sorry about the delay but I have to have a holiday sometime! that's why you had a newsletter in mid-July and another one in late July/early August), but it all looks very good at the moment.

Meanwhile, a newcomer to the fold, namely the North Herts User Group. They hold regular meetings, talks, exchange of news and views, etc., and the name and address to go to for all the gen is:-

P.N. Mortiboy
2 Spurr's Close
Hitchin
Herts
SG4 9QE

Alternatively you can telephone Hitchin (STD 0462) 54435. They will be covering any Commodore related equipment, including of course the new VICs. Looks promising.

Worthy Group

Another worthy group who deserve mention in view of the sterling work they're doing is the North East London Polytechnic, who are in fact organising a series of short, intensive "hands-on" courses aimed at PET Users. These will be in September, and are directed as follows:-

Beginners' Basic –
September 21st
Disk Utilisation –
September 22nd
Assembly Language –
September 23rd
Pascal –
September 24th and 25th

The courses themselves are held in idylic, modern, purpose built training accommodation, with easy access to central London. The fee, which includes course material, meals and VAT where applicable, is £50.00 for the one day courses, and £100.00 for the two day Pascal course. The number to ring if you're interested is 01-590-7722, exts. 4074/2047/4073, or alternatively write to:-

Dr. E. Khabie-Zeitoune
Division of Analytical Services (DAS)
Department of Mathematics
North East London Polytechnic
Romford Road
London E15 4LZ

You'll find them well worth the money.

More Educational Software

It's been said many times that one of the great features of the PET is the amount of good software around, and there appears recently to be a great increase in the number of people supplying software, the majority of them at very reasonable prices.

One such supplier worthy of a mention is Tim Scratcherd, of Darlington Educational Software, who produce lots of good quality educational software, mainly for the standard 8K PET that is so common in the educational field. The full address for further information is:-

Tim Scratcherd
Darlington Educational Software
36 Westmoreland Street
Darlington
Co. Durham DL3 0NX

Medical Aid for VICs

The VIC is just about with us now, and one thing that every potential VIC owner would like to know is the amount of dealer support that will be available. It is anticipated that most of the existing Commodore dealers will be supporting the VIC – special training courses have been given – and that means that over 150 outlets around the country will be there ready and willing to help you out.

One that perhaps deserves special mention is the VIC Centre, being set up by Adda Computers, of West Ealing. Over liquid refreshment with Chris Palmer of Adda (he's the male on the right of the picture, unknown female companion on the left) I got the low down on their aims and ambitions.

Their initial aim is to get estab-

lished as the U.K. centre for VIC add-ons, both software and hardware. Everything imaginable will be available from the VIC centre, including items from companies like ACT as well as Commodore, and off the shelf as well. They'll be setting up a special "cash and carry" service at the centre, but if you can't make it there you'll be able to get all the goodies via mail-order from the Adda Store.

Credit facilities will be available, and all the major bankers cards will be accepted as well.

VIC Newsletter

An extra service that the VIC centre will be providing is a VIC newsletter, which will be appearing four times a year. Needless to say we'll be co-operating very closely with them on this. ("Anything for a few pints", said well-known Adda spokesman Chris Palmer). The newsletter will be covering such things as new products, technical info, applications stories, correspondence and so on. There will also be an update service for hardware and software via the mail.

They will be providing technical and engineering support at the centre for people who purchase from there, and judging by what I saw there the other day that level of supporrt should be very good indeed. Chris has already established himself as an expert on the VIC, and indeed has even helped the B.B.C. out of a couple of projects, one of which was VIC based. See the next newsletter for the other story, which your hapless correspondent somehow got himself involved with as well!

Support is something that is vital to any computer user – it looks like the VIC centre will be the envy of the U.K. in that respect, or indeed the world – the VIC centre's support will be world wide.

And where is all this going on?

Their address is:-

154 victoria Road
Acton
London W3

Opening times will be from 10 to 6 on Tuesday to Friday, and 10 to 5 on Saturday. Their 'phone number hasn't been established at the time of going to press, so for the meantime, for further information ring 01-579 5845 and ask for Chris Palmer.

As a footnote Chris asked me to mention that he plays in a very good rock band, and if you ever want to book a good band . . .!

Chris Tarquin (right)

LETTER TO THE EDITOR

Dear Sir, Madam or Labour Pary Member.

Please find enclosed herewith, forthwith and not-withstanding a new fangled Kodak reproduction of myself taken at the Stoat Pluckers and Aardvark scrambling convention in the Sudan back in '14. As you can see it is perfectly suitable for family consumption (in a white wine sauce with anchovies), containing no reference to sex & alcohol, with only a passing drug reference contained in the poppy pinned to my companions left breast (which for reasons of security will hereafter be referred to as left breast X) also the picture contains no artificial flavorings and is surprisingly low in cholesterol.

Your mission, should you choose to accept it, is to publish this picture without fear of reprisals from my worldwide fan club and to return it to me with a large royalty cheque or a MARS bar which ever is the more valuable.

Your humble and faithful servant.
Major Chris Tarquin Lim-Bim-Wim-Bim Palmer, IPA, ESB, Ruddles and Bar.

P.S. I must take this opportunity to critisise your magazine for not containing a regular feature on either music or drink, a most deplorable absence along with no page 3 feature on certain Commodore staff (lech lech slobber slobber!!!).

HELP!!

There is a proliferation of programming aids, add-ons etc. available at the moment, and it is difficult for the newcomer to know which way to turn. I've been lately using what could possibly be the ultimate in this area, a product called HELP. I dislike using the word "ultimate", because as soon as you say it it becomes obsolete, but I think you'll have to go a long way to beat this.

A total of 43 new commands are added to your PET, and it will run on any PET you care to name, from the original 2001/8 up to the 8032 (and no doubt beyond that as well in a very short space of time). Of these 43 commands, 22 are added to BASIC, including the by-now familiar statements like renumber, trace, find and so on. 13 commands are for use in entering, debugging, listing and execution of programs in machine language, and the final 8 cover the disk operating system.

All of the commands are very easy to enter, consisting of just one or two key strokes, and the command range is very large.

All in all this is a most useful addition to any PET system, but as yet I don't think there's a U.K. distributor for it – if anybody knows of one, please write and let me know and I'll publish details in a future newsletter. At present it can be obtained from:-

Print-Technik
Stumpergasse 34
A-1060 Wien
Austria
Tel. (01934 222) 57 34 23

Arcadaphobia

Thanks to those of you who've written in regarding highest scores on Arcade games, and here we have the definitive list at the time of writing. If any of you have beaten these scores, feel free to write in and let me know – I'll publish the details as the newsletters go by.

For the 8 man 'vs' machine games that we sell, here goes:-

Invaders – 57,790
C. Douglas-Fairhurst
Acrobat – 20,900 –
C. Douglas-Fairhurst
Breakthrough – 4,477
C. Douglas-Fairhurst
Night Drive – 1,512 –
M.W. Sargaison
Car Race – 9,560 –
Peter Gerrard
Crazy Balloon – 8,540 –
David Pocock
Cosmic Jailbreak – 80,900 –
Pete Gerrard
Cosmiads – 61,290 –
M.W. Sargaison

great credit must go to M.W. Sargaison for his efforts on Night Drive and Cosmiads – he is just 12 years of age. And if Derek Hipkin is reading this, I know you got a higher score on Cosmiads, but I think it would be unfair to include you – after all, you did write the program!

UMIST

You will have read in the last issue of the courses being organised by Computer Camp and in this issue of those being done by the North East London Polytechnic. Just so that you don't accuse us of southern bias (with an editor who comes from Wigan? – knows no peer) we've received details of a series of residential courses being organised up in Manchester – under cover you'll be pleased to note. Incidentally, if you know of any courses being run, or are organizing any yourself, please send me details and I'll be happy to publish them in the newsletter. Bear in mind 'though that I'll need at least a couple of months notice in order to get details into print in time, and to give you time to decide whether you're going or not.

Four Courses

There are four courses in all, but as one of them will have already taken place by the time you read this I'll have to give that one a miss. The first one that becomes relevant is from the 9th to the 14th of August, for school teachers, and the aim is to "achieve sufficient expertise to make efficient and effective use of their school's microcomputer". There will be some 14 hours "hands-on" experience on PETs, coupled with about 18 hours of lecture/seminar periods. There'll also be a number of teachers from other academies who will also be contributing.

Fully Residential

The cost of this fully residential course is a remarkably cheap £85.00 for the six day period, and this is also the cost for the other courses. These are aimed at school children as opposed to teachers, and in particular for fifth or sixth formers. However, applications from particularly interested and capable children who haven't yet reached the fifth form will also be considered. The dates of these are from the 23rd to the 28th of August, and from the 31st of August to the 5th September. Your child may start out knowing nothing other than interest, but I can guarantee that by the time he (or she of course) has finished the course they'll be solidly equipped for a future in the world of micros.

If you're interested write to:-
The Business Manager
UMIST
P.O. Box 88
Manchester
M60 1QD

and include your cheque for £85.00 and full address for correspondence.

Is your child's future worth £85.00?

Wordpro/Wordcraft

(A letter from Stenodac House Office Training Centre continuing the great debate)

Dear Sir

As I now use WordPro 4 to teach word processing – having graduated up from COMWORDPRO III – I have been particularly interested in the articles in CPUCN 3/1 and 3/2. I have found the WordPro 4 to be a very good and powerful program to which your to articles did not do full justice due to some inaccurate comments and omission of many of the advanced features.

In your first article you state that WordPro 4 has two extra functions when compared with WordPro 1, 2 and 3, namely output of documents to the screen as they will be typed on paper and the availability of the pound sign. the availability of the pound sign in lieu of the "hash" is a function of the thimble or daisy-wheel in use. The important feature of WordPro 4 in this respect is that symbols on the thimble or daisy-wheel not normally accessible from the keyboard can be obtained by assigning any number form 0 to 9 to the appropriate ASCII value.

In the first article reference is made to only 23 lines of a directory being avialable in Extra Text. This depends in fact on the number of Extra Text lines selected when setting up the WordPro program after switching on and may be up to 85. Also one can at any time re-set the distribution of lines between Main and Extra Text without shutting down the computer or leaving the WordPro mode.

It is stated that the WordCraft 80 standard letter capability is superior to that of WordPro because the fill-file is easier to write and the information is taken from the disk. I have had no difficulty in creating a fill file and I would point out that with WordPro 4 one can choose between having the file information in memory or drawing it from a sequential file on the disk (entered in BASIC mode) so it presumably has the same address file capability as WordCraft.

There are other points in favour of WordPro which I would like to make but this letter would become too long. I will end however by saying that the ability to read all the unformatted text as it has been put on the screen with the WordPro is a marked advantage over having to side scroll for each line with WordCraft when working on more than 80 columns of text. With WordPro one can always output to the screen to see the text in final form and revert at any time to carry out corrections or alterations.

Yours sincerely, Mrs M. Burge

Faster Basic, Faster Loading

Two innovative software products from PET specialists SUPERSOFT have recently been announced.

Both products are supplied as re-programmable EPROMs, something that will not come as a surprise to those who have watched SUPERSOFT take the leading role in the plug-in market. Each is designed to make a standard PET work just that little bit faster, but they operate in very different ways.

FASTER BASIC will appeal particularly to harassed businessmen who find that the packages they are

using (or the programs they have written themselves) are that little bit too slow. It can secure up to half of the time saving that might be gained with a £300 compiler, but it costs just one-tenth as much, a mere £30! FASTER BASIC improves the efficiency of the Basic interpreter by a staggering margin, so that in extreme cases statements are executed more than 40 times faster – although gains of 50 to 100% are more usual. And because it operates through the interpreter no compilation is necessary – so programs can be edited and debugged as usual, with the benefit of all the normal error messages. FASTER BASIC will function with existing programs, no modifications are necessary!

ARROW is a chip that enables programs to be transferred to and from cassette at a phenomenal speed using the standard CBM deck. LOADing and SAVEing is speeded up by 6 to 7 times so that a 6k program loads in under 20 seconds, or a 24k program in a minute (normal loading time 7 minutes 35 seconds). Although the CBM disk unit is faster still, the price of ARROW at just £30 should appeal to the thousands of users who are unable to justify the purchase of a set of drives costing £695. VERIFY and APPEND are also supported by ARROW, whilst bonus features include auto-repeating, double-density plotting, and a hexadecimal calculator mode!

SUPERSOFT have been producing top-quality products for the PET since 1978, a long time in the history of microcomputing in Britain. In the last financial year turnover was in six figures for the first time, and sales for the current year look like doubling last year's record. Although partners Pearl Wellard and Peter Calver specialise in software, they hope to unveil at the PETshow an exciting piece of hardware selling at less than half the price of its nearest competitor.

For more information contact PETER CALVER at SUPERSOFT.

Second Hand Equipment

Since the June newsletter a number of you have written in with details of second hand PET equipment, so below is a list of details. Please write to the people named below, not me.

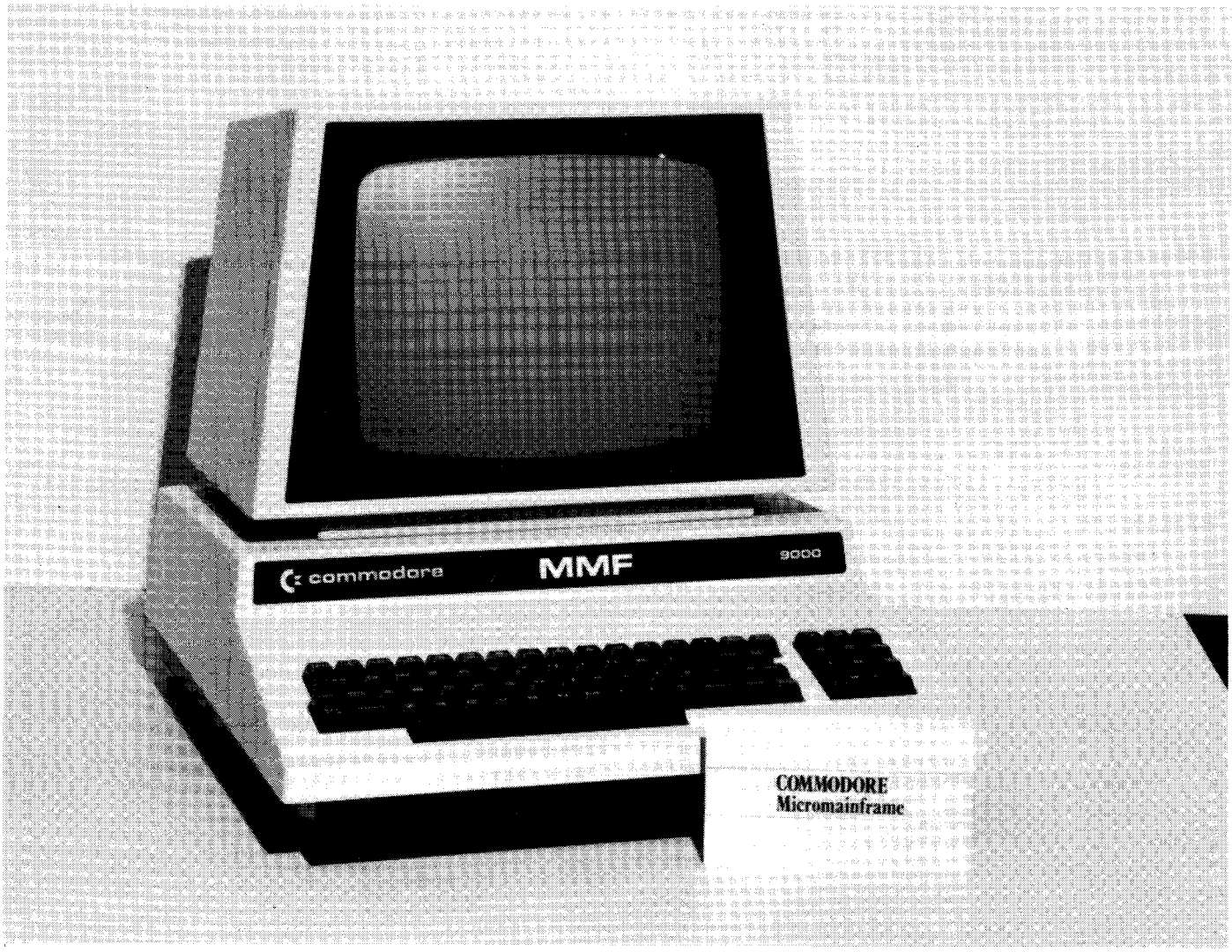
Name	Equipment	Age
Mr. Smith Blake Marstow Priest Ltd. 52 Station Road Egham Middlesex	2001/8 3022 3040, C2N, dustcovers etc.	18-24 months 18-24 months 18-24 months
Mr. Heaver A.G. Stone & Partners 32 Wharf Road Ash Vale Aldershot Hampshire GU12 5AY Tel. 0252-317943	3032 3040	18 months 6 months
Dr. V.R. Rao 17 Priory Close Sandwell Valley West Bromwich West Midlands Tel. 021-553-6151	Basic 2 roms Basic 2 toolkit	? ?
A. Singh 103 Ridgeview Road Whetstone London N20	3032 3040 3022 C2N All interconnecting cables Plus manuals and stationary	

Mr. Singh is looking for around £1,500.00 for the lot, and it's all as new, since he's just upgraded to a set of 8000 series equipment.

R.T. Hills S.W. Hill & Sons Ltd 19/23 George St. Ramsgate Kent CT11 9AS	2023	4 months old

This is being done on a "Cash and Carry" basis, and any inspection or demonstration can be arranged during working hours by contacting Mr. Hills.

New Products



The Micro Mainframe

At the recent Hanover Fair in Germany Commodore International Ltd. introduced the "Micro-Mainframe, a new generation computer that combines the power and languages available on mainframe systems with the low cost of microcomputers.

Applications developed on the Micro-Mainframe can be up-loaded to a mainframe system, and executed without modification.

The computer is based on the standard Commodore Business Machines Model 8032 microcomputer, featuring an integrated green phosphor 12 inch (80 x 25) display, 73-key typewriter style keyboard with standard upper/lower case numeric keypad and full cursor control.

The Micro-Mainframe is a pseudo 16 bit 6809 based system with 36K ROM, 96K user RAM and 2K screen RAM (134K total) that supports all current CBM peripherals except the C2N cassette recorder.

The communication facility supports the standard RS232 C interface with speeds up to 9600 baud.

Files are stored in true ACSII format for communication and compatibility with mainframe systems. The Micro-Mainframe allows the generation, testing, editing and debugging of program source files in the interpretive mode.

These files can then be executed on the MicroMainframe or transmitted and executed on the mainframe system utilizing the same language interpreters.

An extensive software package for the MicroMainframe has been developed by Waterloo Computing Systems Limited to meet the requirements of the University of Waterloo, Waterloo, Ontario. This portable software is particularly suited to microcomputers, but identical versions are available on medium and large scale systems. Thus a user is not limited by the capacity of the micro; the identical program will run without modification on many of the largest and fastest equipment available.

The package consists of interpreters for various langauges, an editor, an operating system (supervisor) and an assembly language development system.

The four language interpreters include Waterloo microBASIC, Waterloo MicroPascal, Waterloo microFORTRAN and Waterloo microAPL. COBOL is under deve-

lopment by Waterloo Computing Systems Limited. These language interpreters have been designed specifically for teaching computer programming. Their design emphasizes good error diagnosis and debugging capabilities which are useful in educational and other program development environments.

Waterloo microBASIC includes ANS Minimal BASIC, with certain minor exceptions, and several extensions such as structured programming control, long names for variables and other program entities, characterstring manipulation, callable procedures and multiline functions, sequential and relative file capabilities, integer arithemetic, debugging facilities, and convenient program entry and editing facilities.

Waterloo MicroPascal is an extensive implementation of Pascal, corresponding very closely to draft proposals being produced by the International Standards Organization (ISO) Pascal Committee. The ISO draft language is a refinement of the language originally defined by Wirth, varying only in minor aspects.

This implementation includes sophisticated features such as text file support, pointer variables, and multi-dimensioned arrays. A significant feature of Waterloo MicroPascal is its powerful interactive debugging facility.

Waterloo microFORTRAN is a special dialect designed for teaching purposes. It has many of the characteristics and much of the flavor of normal FORTRAN, but varies significantly from established standards for that language.

This language processor has many of the important characteristics of the WATFIV-S compiler, which is widely used on IBM computers, plus some features from the new FORTRAN-77 definition.

Examples of language features supported are FORMAT, subroutines and functions, multi-dimensioned arrays, extended character-string manipulation, structured programming control and file input/output. In addition, the interpreter provides a powerful interactive debugging facility.

Waterloo MicroAPL is intended to be a complete and faithful implementation of the IBM/ACM standard for APL with respect to the syntax and semantics of APL statements, operators and primitive functions, input/output forms, and defined functions.

System commands, system variables and system functions are those consistent with a single-user environment. There are no significant design limitations on the rank or shape of arrays or name length. The shared variable processor is omitted. Extensions include system functions supporting files of APL arrays. APL equivalent of BASIC features PEEK, POKE and SYS are included.

A text editor, known as Waterloo microEDITOR, is included which is suitable for creating and maintaining both program and source data files.Its a traditional line-oriented text editor with powerful text searching and substitution commands including global change. Full-screen support and special function keys allow text to be altered, inserted and deleted on the screen without entering commands. Facilities for repeating and editing previously issued commands further enhance the useability of this editor.

A library of functions and procedures is supplied for general use by other programs included in the Package. The Library includes support functions for input/output operations to the keyboard, screen and peripheral devices. Other elements of the Library provide floating-point arithmetic, fundamental trigonometric functions, and several general purpose utility functions.

A Serial Line Setup program is included which provides for the selection of programmable characteristics, such as baud rate. The program includes support for establishing communication with a host computer, through a serial line, for accessing the host's files and peripheral devices.

Reference manuals, textbooks and instructors' guides are available for each software component of the system.

The Micro-Mainframe can operate as a stand-alone system, with the wide variety of software available, as a mainframe system development tool with the available languages and upload/off-load capabilities, and in the educational environment for training in languages and system design.

Pricing details will follow and deliveries are scheduled for late 1981.

Disk-oriented Assembler and Linker programs, the Waterloo 6809 Assembler and Linker, are included which support development of general purpose Motorola 6809 machine-language programs.

The Assembler supports syntax and directives for Motorola 6809 assembly language and includes powerful macro capabilities. In addition, the Assembler supports pseudo opcodes for structured programming control, long names (labels) for meaningful identification of program segments and data, and the ability to include definitions from separate files. The Assembler produces both a listing and re-locatable object file.

The Linker allows the combination of an arbitrary number of relocatable object files to produce an absolute loadable and executable program file. Since it is disk-oriented, the Linker is capable of building programs which are larger than the RAM work space available. The Linker supports building of programs in segments or banks for operation in bank-switched RAM memory, as well as supporting building of programs for operation in normal RAM memory.

The Waterloo micro-SUPERVISOR is an operating system designed for single-user microcomputer environments. It includes a Monitor, Library and Serial Line Communication support.

The Monitor program supports loading of Linker-produced program files into bank-switched RAM memory or normal RAM memory. The Monitor also provides facilities which are useful for debugging machine-language programs. These include commands to display and alter RAM memory and 6809 microprocessor registers, utilizing full screen features for ease of use. In addition, a Monitor command permits disassembly of Motorola 6809 microprocessor instructions into assembly-language mnemonics.

VIC Zero page MEMORY MAP
Compiled by Jim Butterfield

Hex	Decimal	Description
0000-0002	0-2	USR jump
0003-0004	3-4	Float-Fixed vector
0005-0006	5-6	Fixed-Float vector
0007	7	Search character
0008	8	Scan-quotes flag
0009	9	TAB column save
000A	10	0=LOAD, 1=VERIFY
000B	11	Input buffer pointer/# subscrpt
000C	12	Default DIM flag
000D	13	Type: FF=string, 00=numeric
000E	14	Type: 80=integer, 00=floating point
000F	15	DATA scan/LIST quote/memry flag
0010	16	Subscript/FNx flag
0011	17	0=INPUT;$40=GET;$98=READ
0012	18	ATN sign/Comparison eval flag
0013	19	Current I/O prompt flag
0014-0015	20-21	Integer value
0016	22	Pointer: temporary strg stack
0017-0018	23-24	Last temp string vector
0019-0021	25-33	Stack for temporary strings
0022-0025	34-37	Utility pointer area
0026-002A	38-42	Product area for multiplication
002B-002C	43-44	Pointer: Start-of-Basic
002D-002E	45-46	Pointer: Start-of-Variables
002F-0030	47-48	Pointer: Start-of-Arrays
0031-0032	49-50	Pointer: End-of-Arrays
0033-0034	51-52	Pointer: String-storage(moving down)
0035-0036	53-54	Utility string pointer
0037-0038	55-56	Pointer: Limit-of-memory
0039-003A	57-58	Current Basic line number
003B-003C	59-60	Previous Basic line number
003D-003E	61-62	Pointer: Basic statement for CONT
003F-0040	63-64	Current DATA line number
0041-0042	65-66	Current DATA address
0043-0044	67-68	Input vector
0045-0046	69-70	Current variable name
0047-0048	71-72	Current variable address
0049-004A	73-74	Variable pointer for FOR/NEXT
004B-004C	75-76	Y-save; op-save; Basic pointer save
004D	77	Comparison symbol accumulator
004E-0053	78-83	Misc work area, pointers, etc
0054-0056	84-86	Jump vector for functions
0057-0060	87-96	Misc numeric work area
0061	97	Accum#1: Exponent
0062-0065	98-101	Accum#1: Mantissa
0066	102	Accum#1: Sign
0067	103	Series evaluation constant pointer
0068	104	Accum#1 hi-order (overflow)
0069-006E	105-110	Accum#2: Exponent, etc.
006F	111	Sign comparison, Acc#1 vs #2
0070	112	Accum#1 lo-order (rounding)
0071-0072	113-114	Cassette buff len/Series pointer

Continued over

0073-008A	115-138	CHRGET subroutine; get Basic char
007A-007B	122-123	Basic pointer (within subrtn)
008B-008F	139-143	RND seed value
0090	144	Status word ST
0091	145	Keyswitch PIA: STOP and RVS flags
0092	146	Timing constant for tape
0093	147	Load=0, Verify=1
0094	148	Serial output: deferred char flag
0095	149	Serial deferred character
0096	150	Tape EOT received
0097	151	Register save
0098	152	How many open files
0099	153	Input device, normally 0
009A	154	Output CMD device, normally 3
009B	155	Tape character parity
009C	156	Byte-received flag
009D	157	Direct=$80/RUN=0 output control
009E	158	Tp Pass 1 error log/char buffer
009F	159	Tp Pass 2 err log corrected
00A0-00A2	160-162	Jiffy Clock HML
00A3	163	Serial bit count/EOI flag
00A4	164	Cycle count
00A5	165	Countdown,tape write/bit count
00A6	166	Tape buffer pointer
00A7	167	Tp Wrt ldr count/Rd pass/inbit
00A8	168	Tp Wrt new byte/Rd error/inbit cnt
00A9	169	Wrt start bit/Rd bit err/stbit
00AA	170	Tp Scan;Cnt;Ld;End/byte assy
00AB	171	Wr lead length/Rd checksum/parity
00AC-00AD	172-173	Pointer: tape bufr, scrolling
00AE-00AF	174-175	Tape end adds/End of program
00B0-00B1	176-177	Tape timing constants
00B2-00B3	178-179	Pntr: start of tape buffer
00B4	180	1=Tp timer enabled; bit cnt
00B5	181	Tp EOT/RS232 next bit to send
00B6	182	Read character error/outbyte buf
00B7	183	# characters in file name
00B8	184	Current logical file
00B9	185	Current secndy address
00BA	186	Current device
00BB-00BC	187-188	Pointer to file name
00BD	189	Wr shift word/Rd input char
00BE	190	# blocks remaining to Wr/Rd
00BF	191	Serial word buffer
00C0	192	Tape motor interlock
00C1-00C2	193-194	I/O start adds
00C3-00C4	195-196	Kernel setup pointer
00C5	197	Last key pressed
00C6	198	# chars in keybd buffer
00C7	199	Screen reverse flag
00C8	200	End-of-line for input pointer
00C9-00CA	201-202	Input cursor log (row, column)
00CB	203	Which key: 64 if no key
00CC	204	0=flash cursor
00CD	205	Cursor timing countdown
00CE	206	Character under cursor
00CF	207	Cursor in blink phase
00D0	208	Input from screen/from keyboard
00D1-00D2	209-210	Pointer to screen line
00D3	211	Position of cursor on above line
00D4	212	0=direct cursor, else programmed
00D5	213	Current screen line length
00D6	214	Row where curosr lives
00D7	215	Last inkey/checksum/buffer
00D8	216	# of INSERTs outstanding
00D9-00F0	217-240	Screen line link table
00F1	241	Dummy screen link
00F2	242	Screen row marker
00F3-00F4	243-244	Screen color pointer

00F5-00F6	245-246	Keyboard pointer	
00F7-00F8	247-248	RS-232 Rcv pntr	
00F9-00FA	249-250	RS-232 Tx pntr	
00FF-010A	256-266	Floating to ASCII work area	
0100-103E	256-318	Tape error log	
0100-01FF	256-511	Processor stack area	
0200-0258	512-600	Basic input buffer	
0259-0262	601-610	Logical file table	
0263-026C	611-620	Device # table	
026D-0276	621-630	Sec Adds table	
0277-0280	631-640	Keybd buffer	
0285	645	Serial bus timeout flag	
0286	646	Current color code	
0287	647	Color under cursor	
0288	648	Screen memory page	
0289	649	Max size of keybd buffer	
028A	650	Repeat all keys	
028B	651	Repeat speed counter	
028C	652	Repeat delay counter	
028D	653	Keyboard Shift/Control flag	
028E	654	Last shift pattern	
028F-0290	655-656	Keyboard table settup pointer	
0291	657	Keymode (Kattacanna)	
0292	658	0=scroll enable	
0293	659	VIC chip control	
0294	660	VIC chip command	
0295-0296	661-662	Bit timing	
0297	663	RS-232 status	
0298	664	# bits to send	
0299-029A	665	RS-232 speed/code	
029B	667	RS232 receive pointer	
029C	668	RS232 input pointer	
029D	669	RS232 transmit pointer	
029E	670	RS232 output pointer	
029F-02A0	671-672	IRQ save during tape I/O	
0300-0301	768-769	Error message link	
0302-0303	770-771	Basic warm start link	
0304-0305	772-773	Crunch Basic tokens link	
0306-0307	774-775	Print tokens link	
0308-0309	776-777	Start new Basic code link	
030A-030B	778-779	Get arithmetic element link	
0314-0315	788-789	Hardware interrupt vector	(EABF)
0316-0317	790-791	Break interrupt vector	(FED2)
0318-0319	792-793	NMI interrupt vector	(FEAD)
031A-031B	794-795	OPEN vector	(F40A)
031C-031D	796-797	CLOSE vector	(F34A)
031E-031F	798-799	Set-input vector	(F2C7)
0320-0321	800-801	Set-output vector	(F309)
0322-0323	802-803	Restore I/O vector	(F3F3)
0324-0325	804-805	INPUT vector	(F20E)
0326-0327	806-807	Output vector	(F27A)
0328-0329	808-809	Test-STOP vector	(F770)
032A-032B	810-811	GET vector	(F1F5)
032C-032D	812-813	Abort I/O vector	(F3EF)
032E-032F	814-815	USR vector	(FED2)
0330-0331	816-817	LOAD link	
0332-0333	818-819	SAVE link	
033C-03FB	828-1019	Cassette buffer	
0400-0FFF	1024-4095	3K RAM expansion area	
1000-1FFF	4096-8191	Normal Basic memory	
2000-7FFF	8192-32767	Memory expansion area	
8000-8FFF	32768-36863	Character bit maps	
9000-900F	36864-36879	Video Interface Chip	
9110-912F	37136-37167	6522 Interface Chips	
9400-95FF	37888-38399	Alternate Colour Nybble area	
9600-97FF	38400-38911	Main Colour Nybble area	
A000-BFFF	40960-49151	Plug-in ROM area	
C000-FFFF	49152-65535	ROM: Basic and Operating System	

Supermon 4

I said a couple of newsletters ago that we'd be publishing the listing of Supermon Four, a program that adds commands to the existing PET monitor, and is very useful when writing machine code programs. Previously only published for Basic 2 machines, we now have the program available for Basic 4 40 and 80 column machines as well.

So, here it is, followed by a summary of the extra commands available in Supermon, as well as the existing commands in the resident PET monitor. It may take a long time typing it in, but you'll find it well worth it in the end.

Type in the small BASIC program first, and save a copy in the normal manner (i.e. just use SAVE). Then type in the machine code part, and save that via the monitor (i.e. enter the monitor with SYS4, and then type:–

s "O:supermon 4",08,0600,0d00

This will save the program onto drive 0 of device 8. If you use tapes, change the 08 to 01, and remove the 0: bit within the quotes.

Finally type RUN and all will be revealed – the PET is yours for the disassembling!

```
 100 PRINT"[CLR,2CD,RVS]    SUPERMON4!"
 110 PRINT"[CD]      DISSASSEMBLER [RVS]D[OFF] BY WOZNIAK/BAUM
 120 PRINT"         SINGLE STEP [RVS]I[OFF] BY JIM RUSSO
 130 PRINT"MOST OTHER STUFF [RVS],HALT[OFF] BY BILL SEILER
 150 PRINT"[CD]TIDIED & WRAPPED BY JIM BUTTERFIELD"
 170 L=PEEK(52)+PEEK(53)*256:SYS1536:M=PEEK(33):N=PEEK(34)
 180 POKE52,M:POKE53,N:POKE48,M:POKE49,N:N=M+N*256
 210 PRINT"[2CD]LINK TO MONITOR -- SYS";N
 220 SYS N
READY.
```

```
?
.
.: 0600 A9 CB 85 1F A9 0C 85 20
.: 0608 A5 34 85 21 A5 35 85 22
.: 0610 A0 00 20 38 06 D0 16 20
.: 0618 38 06 F0 11 85 23 20 38
.: 0620 06 18 65 34 AA A5 23 65
.: 0628 35 20 43 06 8A 20 43 06
.: 0630 20 50 06 90 DB 60 EA EA
.: 0638 A5 1F D0 02 C6 20 C6 1F
.: 0640 B1 1F 60 48 A5 21 D0 02
.: 0648 C6 22 C6 21 68 91 21 60
.: 0650 A9 80 C5 1F A9 06 E5 20
.: 0658 60 AA AA AA AA AA AA AA
.: 0660 AA AA AA AA AA AA AA AA
.: 0668 AA AA AA AA AA AA AA AA
.: 0670 AA AA AA AA AA AA AA AA
.: 0678 AA AA AA AA AA AA AA AA
.: 0680 AD FE FF 00 85 34 AD FF
.: 0688 FF 00 85 35 AD FC FF 00
.: 0690 8D FA 03 AD FD FF 00 8D
.: 0698 FB 03 00 00 A2 08 DD DE
.: 06A0 FF 00 D0 0E 86 B4 8A 0A
.: 06A8 AA BD E9 FF 00 48 BD E8
.: 06B0 FF 00 48 60 CA 10 EA 4C
.: 06B8 9A FA 00 A2 02 2C A2 00
.: 06C0 00 B4 FB D0 08 B4 FC D0
.: 06C8 02 E6 DE D6 FC D6 FB 60
.: 06D0 20 98 D7 C9 20 F0 F9 60
.: 06D8 A9 00 00 8D 00 00 01 20
.: 06E0 79 FA 00 20 6B D7 20 57
.: 06E8 D7 90 09 60 20 98 D7 20
.: 06F0 54 D7 B0 DE AE 06 02 9A
.: 06F8 4C A4 D7 20 31 D5 CA D0
.: 0700 FA 60 E6 FD D0 02 E6 FE
.: 0708 60 A2 02 B5 FA 48 BD 0A
.: 0710 02 95 FA 68 9D 0A 02 CA
.: 0718 D0 F1 60 AD 0B 02 AC 0C
.: 0720 02 4C CE FA 00 A5 FD A4
.: 0728 FE 38 E5 FB 8D 1B 02 98
.: 0730 E5 FC A8 0D 1B 02 60 20
.: 0738 81 FA 00 20 44 D7 20 92
.: 0740 FA 00 20 AF FA 00 20 92
.: 0748 FA 00 20 CA FA 00 20 44
.: 0750 D7 90 15 A6 DE D0 65 20
.: 0758 C1 FA 00 90 60 A1 FB 81
.: 0760 FD 20 A8 FA 00 20 39 D5
.: 0768 D0 EB 20 C1 FA 00 18 AD
.: 0770 1B 02 65 FD 85 FD 98 65
.: 0778 FE 85 FE 20 AF FA 00 A6
.: 0780 DE D0 3D A1 FB 81 FD 20
.: 0788 C1 FA 00 B0 34 20 65 FA
.: 0790 00 20 68 FA 00 4C 1B FB
.: 0798 00 20 81 FA 00 20 44 D7
.: 07A0 20 92 FA 00 20 44 D7 20
.: 07A8 98 D7 20 63 D7 90 14 85
.: 07B0 B5 A6 DE D0 11 20 CA FA
.: 07B8 00 90 0C A5 B5 81 FB 20
.: 07C0 39 D5 D0 EE 4C 9A FA 00
.: 07C8 4C BA D4 20 81 FA 00 20
.: 07D0 44 D7 20 92 FA 00 20 44
.: 07D8 D7 20 98 D7 A2 00 00 20
.: 07E0 98 D7 C9 27 D0 14 20 98
.: 07E8 D7 9D 10 02 E8 20 CF FF
.: 07F0 C9 0D F0 22 E0 20 D0 F1
.: 07F8 F0 1C 8E 00 00 01 20 6B
.: 0800 D7 90 C6 9D 10 02 E8 20
.: 0808 CF FF C9 0D F0 09 20 63
.: 0810 D7 90 B6 E0 20 D0 EC 86
.: 0818 B4 20 34 D5 A2 00 00 A0
.: 0820 00 00 B1 FB DD 10 02 D0
.: 0828 0C C8 E8 E4 B4 D0 F3 20
.: 0830 17 D7 20 31 D5 20 39 D5
.: 0838 A6 DE D0 92 20 CA FA 00
.: 0840 B0 DD 4C BA D4 20 81 FA
.: 0848 00 8D 0D 02 A5 FC 8D 0E
.: 0850 02 A9 04 A2 00 00 8D 09
.: 0858 02 8E 0A 02 A9 93 20 D2
.: 0860 FF A9 16 85 B5 20 06 FC
.: 0868 00 20 64 FC 00 85 FB 84
.: 0870 FC C6 B5 D0 F2 A9 91 20
.: 0878 D2 FF 4C BA D4 A0 2C 20
.: 0880 79 D5 20 17 D7 20 31 D5
.: 0888 A2 00 00 A1 FB 20 74 FC
.: 0890 00 48 20 BB FC 00 68 20
.: 0898 D3 FC 00 A2 06 E0 03 D0
.: 08A0 13 AC 1C 02 F0 0E A5 FF
.: 08A8 C9 E8 B1 FB B0 1C 20 5C
.: 08B0 FC 00 88 D0 F2 06 FF 90
.: 08B8 0E BD 51 FF 00 20 45 FD
.: 08C0 00 BD 57 FF 00 F0 03 20
.: 08C8 45 FD 00 CA D0 D4 60 20
.: 08D0 68 FC 00 AA E8 D0 01 C8
.: 08D8 98 20 5C FC 00 8A 86 B4
.: 08E0 20 22 D7 A6 B4 60 AD 1C
.: 08E8 02 38 A4 FC AA 10 01 88
.: 08F0 65 FB 90 01 C8 60 A8 4A
.: 08F8 90 0B 4A B0 17 C9 22 F0
.: 0900 13 29 07 09 80 4A AA BD
.: 0908 00 FF 00 B0 04 4A 4A 4A
.: 0910 4A 29 0F D0 04 A0 80 A9
.: 0918 00 00 AA BD 44 FF 00 85
.: 0920 FF 29 03 8D 1C 02 98 29
.: 0928 8F AA 98 A0 03 E0 8A F0
.: 0930 0B 4A 90 08 4A 4A 09 20
.: 0938 88 D0 FA C8 88 D0 F2 60
.: 0940 B1 FB 20 5C FC 00 A2 01
.: 0948 20 A1 FA 00 CC 1C 02 C8
.: 0950 90 F0 A2 03 CC 09 02 90
.: 0958 F0 60 A8 B9 5E FF 00 8D
.: 0960 0B 02 B9 9E FF 00 8D 0C
.: 0968 02 A9 00 00 A0 05 0E 0C
.: 0970 02 2E 0B 02 2A 88 D0 F6
.: 0978 69 3F 20 D2 FF CA D0 EA
.: 0980 4C 31 D5 20 81 FA 00 20
.: 0988 44 D7 20 92 FA 00 20 44
.: 0990 D7 A9 04 A2 00 00 8D 09
.: 0998 02 8E 0A 02 20 34 D5 20
.: 09A0 0B FC 00 20 64 FC 00 85
.: 09A8 FB 84 FC 20 35 F3 F0 05
.: 09B0 20 CA FA 00 B0 E9 4C BA
.: 09B8 D4 20 81 FA 00 A9 03 85
.: 09C0 B5 20 98 D7 20 0B D5 D0
.: 09C8 F8 AD 0D 02 85 FB AD 0E
.: 09D0 02 85 FC 4C E7 FB 00 CD
.: 09D8 0A 02 F0 03 20 D2 FF 60
.: 09E0 A9 03 A2 24 8D 09 02 8E
.: 09E8 0A 02 20 34 D5 78 AD FA
.: 09F0 FF 00 85 90 AD FB FF 00
.: 09F8 85 91 A9 A0 8D 4E E8 CE
.: 0A00 13 E8 A9 2E 8D 48 E8 A9
.: 0A08 00 00 8D 49 E8 AE 06 02
.: 0A10 9A 4C 55 D6 20 C0 FC 68
.: 0A18 8D 05 02 68 8D 04 02 68
.: 0A20 8D 03 02 68 8D 02 02 68
.: 0A28 8D 01 02 68 8D 00 00 02
.: 0A30 BA 8E 06 02 58 20 34 D5
.: 0A38 20 23 D5 85 B5 A0 00 00
.: 0A40 20 FE D4 20 31 D5 AD 00
.: 0A48 00 02 85 FC AD 01 02 85
.: 0A50 FB 20 17 D7 20 0E FC 00
```

```
.:  0A58 20 35 F3 C9 F7 F0 F9 20  .:  0B28 FE 00 D0 A9 20 DF FE 00  .:  0BF8 21 81 82 00 00 00 00 59
.:  0A60 35 F3 D0 03 4C BA D4 C9  .:  0B30 D0 A4 AD 0B 02 C5 B5 D0  .:  0C00 4D 91 92 86 4A 85 9D 2C
.:  0A68 FF F0 F4 4C 5B FD 00 20  .:  0B38 9D 20 44 D7 AC 1C 02 F0  .:  0C08 29 2C 23 28 24 59 00 00
.:  0A70 81 FA 00 20 44 D7 8E 11  .:  0B40 2F AD 0C 02 C9 9D D0 20  .:  0C10 58 24 24 00 00 1C 8A 1C
.:  0A78 02 A2 03 20 79 FA 00 48  .:  0B48 20 CA FA 00 90 0B 98 D0  .:  0C18 23 5D 8B 1B A1 9D 8A 1D
.:  0A80 CA D0 F9 A2 03 68 38 E9  .:  0B50 05 AE 1B 02 10 0B 4C 9A  .:  0C20 23 9D 8B 1D A1 00 00 29
.:  0A88 3F A0 05 4A 6E 11 02 6E  .:  0B58 FA 00 C8 D0 FA AE 1B 02  .:  0C28 19 AE 69 A8 19 23 24 53
.:  0A90 10 02 88 D0 F6 CA D0 ED  .:  0B60 10 F5 CA CA 8A AC 1C 02  .:  0C30 1B 23 24 53 19 A1 00 00
.:  0A98 A2 02 20 CF FF C9 0D F0  .:  0B68 D0 03 B9 FC 00 00 91 FB  .:  0C38 1A 5B 5B A5 69 24 24 AE
.:  0AA0 1E C9 20 F0 F5 20 F7 FE  .:  0B70 88 D0 F8 A5 DE 91 FB 20  .:  0C40 AE A8 AD 29 00 00 7C 00
.:  0AA8 00 B0 0F 20 78 D7 A4 FB  .:  0B78 64 FC 00 85 FB 84 FC A0  .:  0C48 00 15 9C 6D 9C A5 69 29
.:  0AB0 84 FC 85 FB A9 30 9D 10  .:  0B80 41 20 79 D5 20 17 D7 20  .:  0C50 53 84 13 34 11 A5 69 23
.:  0AB8 02 E8 9D 10 02 E8 D0 DB  .:  0B88 31 D5 4C D8 FD 00 A8 20  .:  0C58 A0 D8 62 5A 48 26 62 94
.:  0AC0 8E 0B 02 A2 00 00 86 DE  .:  0B90 E6 FE 00 D0 11 98 F0 0E  .:  0C60 88 54 44 C8 54 68 44 E8
.:  0AC8 F0 04 E6 DE F0 7B A2 00  .:  0B98 86 B4 A6 B5 DD 10 02 08  .:  0C68 94 00 00 B4 08 84 74 B4
.:  0AD0 00 86 B5 A5 DE 20 74 FC  .:  0BA0 E8 86 B5 A6 B4 28 60 C9  .:  0C70 28 6E 74 F4 CC 4A 72 F2
.:  0AD8 00 A6 FF 8E 0C 02 AA BC  .:  0BA8 30 90 03 C9 47 60 38 60  .:  0C78 A4 8A 00 00 AA A2 A2 74
.:  0AE0 5E FF 00 BD 9E FF 00 20  .:  0BB0 40 02 45 03 D0 08 40 09  .:  0C80 74 74 72 44 68 B2 32 B2
.:  0AE8 E0 FE 00 D0 E2 A2 06 E0  .:  0BB8 30 22 45 33 D0 08 40 09  .:  0C88 00 00 22 00 00 1A 1A 26
.:  0AF0 03 D0 1A AC 1C 02 F0 15  .:  0BC0 40 02 45 33 D0 08 40 09  .:  0C90 26 72 72 88 C8 C4 CA 26
.:  0AF8 A5 FF C9 E8 A9 30 B0 21  .:  0BC8 40 02 45 B3 D0 08 40 09  .:  0C98 48 44 44 A2 C8 54 46 48
.:  0B00 20 E6 FE 00 D0 CA 20 E8  .:  0BD0 00 00 22 44 33 D0 8C 44  .:  0CA0 44 50 2C 41 49 4E 00 00
.:  0B08 FE 00 D0 C5 88 D0 EB 06  .:  0BD8 00 00 11 22 44 33 D0 8C  .:  0CA8 DB FA 00 30 FB 00 5E FB
.:  0B10 FF 90 0B BC 57 FF 00 BD  .:  0BE0 44 9A 10 22 44 33 D0 08  .:  0CB0 00 D1 FB 00 F8 FC 00 28
.:  0B18 51 FF 00 20 E0 FE 00 D0  .:  0BE8 40 09 10 22 44 33 D0 08  .:  0CB8 FD 00 D4 FD 00 4D FD 00
.:  0B20 B3 CA D0 D0 F0 0A 20 DF  .:  0BF0 40 09 62 13 78 A9 00 00  .:  0CC0 B9 D4 7F FD 00 4A FA 00
                                                                    .:  0CC8 33 FA 00 AA AA AA AA AA
```

SUPERMON COMMANDS

SIMPLE ASSEMBLER

.A 2000 LDA #$12
.A 2002 STA $8000,X
A. 2005 (RETURN)

In the above example the user started assembly at 2000 hex. The first instruction was load a register with immediate 12 hex. In the second line the user did not need to type the 'A' and address. The simple assembler retyped the last entered line and prompts with the next address. To exist the assembler, type 'RETURN' after the address prompt. Syntax is the same as the disassembler output.

DISASSEMBLER

.D 2000
(screen clears)
., 2000 A9 12 LDA #$12
., 2002 9D 00 80 STA $8000,X
., 2005 AA TAX
., 2006 AA TAX
(full page of instructions)

Disassembles 22 instructions starting at 2000 hex. The three bytes following the address may be modified by using the cursor keys to move and modify the bytes. Then hit 'RETURN' and the bytes in memory will be changed. Supermon will then disassemble that page again.

PRINTING DISASSEMBLER

.P 2000, 2040
2000 A9 12 LDA #$12
2002 9D 00 80 STA $8000,XY.
2005 AA TAX
....
203F A2 00 LDX #$00

To engage printer, set up before hand with OPEN4,4:CMD4. Access the monitor via a call of SYS 54386 (*NOT* a break).

SINGLE STEP

.I

Allows a machine language program to be run step by step. Call register display with .R and set the PC address to the desired first instruction for single stepping. The .I will cause a single step to execute and will disassemble the next line.
Controls:–
< for single step
RVS for slow step
SPACE for fast stepping
STOP to return to monitor
(On business keyboards use 8, back arrow, 6 and STOP)

FILL MEMORY

.F 1000 1100 FF

This fills the memory from 1000 hex to 1100 hex with the byte FF hex.

GO RUN

.G

Go to the address in the PC register display and begin run code. All the registers will be replaced with the displayed values.
.G 1000

Go to address 1000 hex and begin running code.

HUNT MEMORY

.H C000 D000 'READ

This hunts through memory for C000 hex to D000 hex for the ascii string 'READ' and prints the address where it's found. A maximum of 32 characters may be used.
.H C000 D000 20 D2 FF

Hunt memory from C000 hex to D000 hex for the sequence of bytes 20 D2 FF and print the address where it is found. A maximum of 32 bytes may be used.

LOAD FROM TAPE

.L

Load any program from tape #1
.L "RAM TEST"

Load from tape #1 the program named "RAM TEST"
.L "RAM TEST",02

Load from tape #2 the program named "RAM TEST".

MEMORY DISPLAY

.M 0000 0080
.: 0000 00 01 02 03 04 05 06 07
.: 0008 08 09 0A 0B 0C 0D 0E 0F

This displays memory from 0000 hex to 0008 hex. The bytes following the address may be modified by editing and then typing a return.

REGISTER DISPLAY

.R

```
      PC SR  AC  XR  YR  SP
.;  000001  02  03  04  05
```

This displays the register values saved when supermon was entered.

Continued on page 25

WHAT REALLY HAPPENS TO A PET WHEN IT ARRIVES IN THE UK.

The same as happens to all PETs when they arrive in the UK – QC – Quarrantine Control.

Before shipment to the UK whether manufactured in the USA, Germany or Japan each individual sub assembly is 100% tested on burn in racks at high temperatures for several hours before assembly into its housing and then tested again. But after the rigours of a trip by air or sea across the atlantic, road freight from Europe or sea from the far east we feel we ought to test them before delivery to our Dealers.

Ideally we would like to burn in each individual unit for 4 hours and this we did in the puppy stage of PET but with average sales of 3000 units/ month that would be 12,000 hours and sales demands just would not allow the time or any additional costs.

Currently we 100% test all new products before release until reliability is proven and then batch test, but to no set formula. In the future we will continue to 100% test new, or particularly 'sensitive' models, but for proven product we will test to British Standard 6001. For those uninitiated people this sets guide lines as to the sample quantity to test from a given batch size and dictates the pass and fail level at a pre-determined 'AQL' – Acceptable Quality Level for that industry. If the sample fails then the batch has to be 100% tested – yes more delays dealer but hopefully a quality to the required standard in the electronics industry. This of course only dictates the pass or fail level, the tests carried out are as follows:

Many a slip...

Just one of the many excitements at the second International PET Show.

CPUs

Processors are subjected to a boot-loader test – no not a 'kick up the khyber' but a device which has an inbuilt test program. This is clipped on to the 6502 processor and initialises and runs itself thus removing the necessity for test disks and cassettes. Whilst any automatic test procedure is very effective it is only as good as the man who wrote it. There may well be routines which have been omitted and that is why we are keen for dealers to pass on end user comments in this sphere.

DISK DRIVES

Disk drives are 100% performance tested. The test is loaded into the PET from the floppy disk being tested. The test diskette is replaced with a blank diskette in each drive and the PET then formats, writes and reads, scratches tracks and sectors across the whole of the disks surface on both drive units. For 8050 drives we have found through experience that it is far more reliable to use certified 77 track diskettes for this purpose. As a result of the inherent sensitivity of adjustment on 77 track drives it is possible that head alignment is effected due to the vibration during transit. However, our dealers are well aware of this and accept that they may have to re-align disks before sale to the end user.

PRINTERS

Printers fall into two categories – those with an inbuilt self test facility and those without.

1. Those with an inbuilt self test are first tested without connection to a computer to ensure that the head drive and printing mechanism are working correctly, this is usually triggered by switching on with the paper feed button depressed. They are then tested through a computer to ensure that the interface circuitry is operating.

2. Both the above tests are run through a computer.

VICs

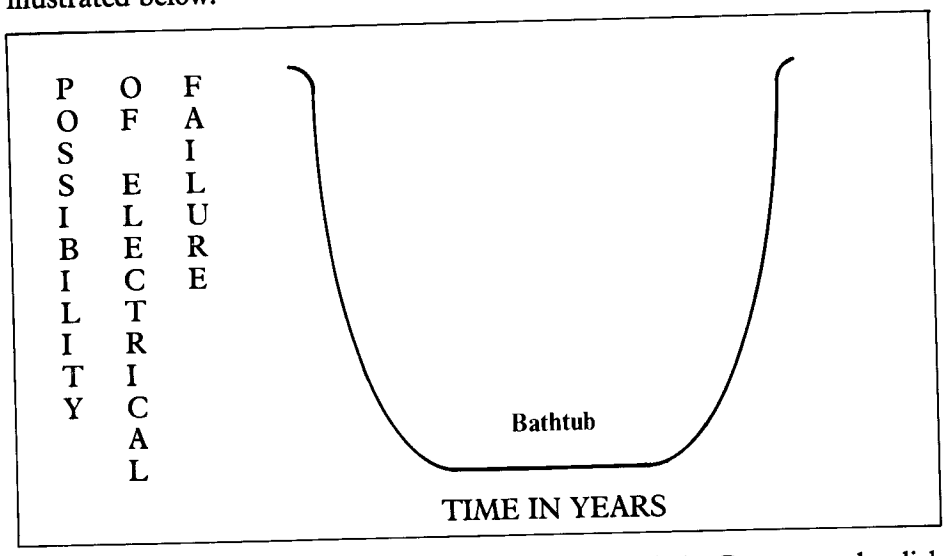
VICs will be 100% tested, initially using a burn in RAM ROM test cartridge. Eventually these will have a bootloader test similar to that used on other processors.

In addition all units have a check list for such things as documentation, cables, cosmetic condition, labels, switches, fuses, screws and nuts, transformer security, earth leads, PCB, keyboard, harness connectors, VDU, LEDs and uncle Tom Cobbeley and all.

Your dealer will also have similar tests to combat the rigours of UK transit and handling.

RELIABILITY

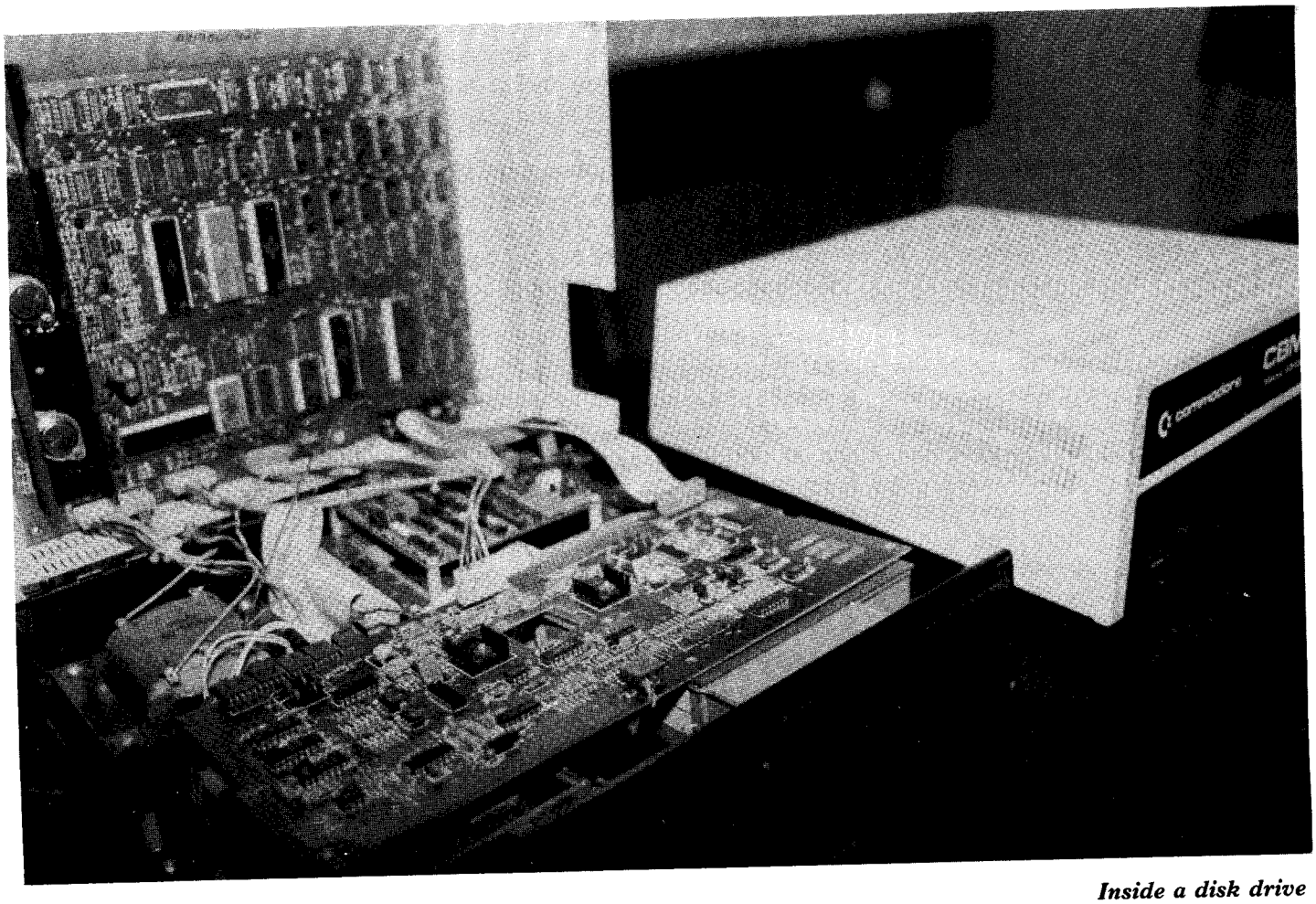
Even after all this your unit may still fail on power up. We are sure you electronic freaks will understand and know of the 'bath tub effect' which is common in the electronics industry. For those of you who do not this is illustrated below:



This means that the likelihood of electrical failure is highest immediately after commissioning, will stabilise for the normal working lifespan of the equipment and then will rise again as the components age. The timescale involved is entirely dependant upon the amount of usage.

INCOMPATABILITY

Each unit is therefore tested hopefully to an acceptable quality level. If you buy a system and it is not compatible initially, just bear with it for a while until each unit learns to speak the same language, after all the processor is made in Germany, the disk drive USA and the printer Japan, Austria or Germany. What do you honestly expect!!!!!



Inside a disk drive

Reviews

VISICALC – visible calculator

A well known oil company claims that it saves more than $27,000 p.a. in outside computer bureau costs with VISICALC. and that's not a bad recommendation for any piece of software. Without a shadow of doubt this program is both practical and versatile. It also makes full use of the potential of the machine.

VISICALC comes with a ROM and an excellent tutorial manual with a command summary card, and all are contained in a ring binder.

What can it do? It can plan budgets, modify projections, cope with financial analysis, predict tax consequences, develop financial reports and generate proformas and analyse surveys etc. You will however, need at least 32K memory. Hard copy is available for a range of printers; I wish this latter feature appeared in more programs. Software writers please note that not everyone owns a CBM printer!

Electronic Sheet

The manual describes the program as an "electronic sheet" suitable for anyone who uses a calculator, a sheet of paper and a pencil (and probably a rubber as well!). The sheet is organised as a grid of rows and columns; the top left position is designated A1 whilst the bottom right is BK254. Each cell may be addressed as a co-ordinate which may contain a label or prompt, a number or a formula. PET's screen acts as a window on this matrix and you may scroll this window in any of the four directions or even split the screen to see any two parts of the sheet at the same time. I mentioned earlier that you would need 32K memory; even this is a limitation and there is no way in which you will be able to use the full matrix. Nevertheless, there is a memory indicator at the top right-hand corner of the screen and this displays 10K bytes free after loading the program from disk.

What If?

So far, so good. Nothing very special yet! The great strength of VISICALC is that you can change any parameter in that matrix and watch all the other columns or rows change with it. Let's say you have a model set up which is based on your household budget. It will contain your income and expenditure. The expenses column contains gas and electricity charges, mortgage repayments, food and clothing etc. Expenses are summed and deducted from income to give us a total profit (or loss!), and then we want to calculate a percentage of our profit to be placed in a Building Society. Fine! Now let's imagine that our mortage payments are going up by 3%. What will the effect be in our balance of account? What change will we now make to the cash allocated to savings? By simply changing the figure in the mortage column you are instantly aware of the repercussions right through the model.

It is this recalculation feature that makes VISICALC a powerful planning and forecasting tool. What could have taken days or weeks will now only take hours or even minutes, and that has to be goods because you'll be right up to date. Full editing facilities are also available and you can insert, change, move or even delete lines, titles or formulae.

Let me say that the above example is rather trite and doesn't reflect the full capabilities of the software. I somehow imagine that our oil company had other things in mind!

Take Time

The first temptation when confronted with an extensive manual is to get the program loaded and then prod around with a few keys to see what happens. Resist the temptation. The manual is well thought out and will lead you and the computer through the intricacies of the program. You may need about five or six hours with the manual, but there are plenty of examples and they all work too.

It's impossible to do justice in a short review to all the features of this package and so I'll split the more important into three sections:-

1. Disk Storage
2. Printout
3. Screen: input and format commands.

Disk Storage

In one sense this is putting the proverbial cart before the horse, but it's probably the easiest to tackle. Once the sheet is set up, it can be stored on diskette, with all the associated data and formatting. A modest sized sheet is going to take around 12 blocks or 3K bytes. File names are permitted on SAVE or LOAD. A very helpful feature on LOADing (especially if you can't remember the file name) is to be able to press Cursor-up and look one by one at the directory entries. When you decide which file you need, simply press RETURN to load it. Mention must be made at this stage of the Data Interchange Format or DIF. This is well documentated at the end of the manual, and basically it is a conversion program which allows VISICALC to talk to other programs and vice versa. VISICALC seems to be avoiding one of the major piitfalls of some other Commodore software, namely insularity.

Initialisation of the disk is also possible from the program but users should be aware that this really means formatting and the loss of previous contents . . . anyway we all make back up copies, don't we?

Printout

All functions are called by pressing a control key ("/") and another key, so at this stage we press control and "P" for printout. A legend appears at the top of the screen:

PRINT: ASCII PET FILE

In fact, the print command will also access the disk unit or any addressable device. Since the matrix will usually be wider than your printer, you may copy the sheet in strips. In any case you must first set up the start and end point, either by cursoring to the desired co-ordinates or simply entering them from the keyboard. Paper can be fed through the printer by holding down the "+" key. A nice little touch of finesse there.

Selecting the FILE option will save the sheet to disk (default Drive 0). The file is then stored in print file format which may then be incorporated into another BASIC program but VISICALC cannot reload data in this format.

Screen Input

VISICALC automatically senses whether you are inputting alphabetic data (labels) or numerical data (values). A value entry can be a simple number, an arithmetic expression or may contain entry position co-ordinates, which are termed value references. Thus 1.5 + (D2/B3) will use the current values stored at D2 and B3 as its calculation of the final result.

The normal arithmetic operators are available (addition, subtraction, division, multiplication and exponention) but the package performs calculations in the order it finds them, from left to right. You can, however, change priority with brackets.

When a value entry has been made, the program displays the calculated value in the entry position on the sheet, but stores the actual formula in memory. There are some useful function arguments as well:–

@SUM (list)	adds value of all entry positions in the list
@MIN (list)	chooses smallest value in list
@MAX (list)	chooses largest value in list
@COUNT (list)	returns number of non-blank entries in list.
@AVERAGE (list)	mean of entries making up arguments

Other features include ABS, INT, SQRT and transcendental functions such as @EXP, @LN (natural log) and @SIN. Where the sheet must be set up before data is available, @NA is used... if they are merely left blank they would otherwise be evaluated as zero.

Graphic drawing facilities are available in the form of simple bar graphs which are returned from integer values in the coordinates that are to be graphed.

The actual format for numbers will depend on the column width and the format chosen (see next section), but the software will switch between conventional and scientific notation as required to display the calculated value to the greatest precision.

While developing a VISICALC sheet I often found that I needed to insert additional rows of columns. Thank goodness this is available with the INSERT command. The sheet will be opened up from the current cursor position and entry references are automatically adjusted for you.

THE INCREDIBLE SHRINKING SHEET

Once I was finished, I must admit that my first efforts looked rather scrappy with quite a few gaps here and there. Saving the sheet to disk at this stage, clearing the screen, and then re-loading, I was amazed to find the sheet had shrunk. Everything was there, but only those labels and numbers actually saved. This is a real bonus and should clear out memory to give you a little more space to play around with.

We'll end this section by looking at an especially useful feature which should save more time and slog. REPLICATION permits an entry or formula to be repeated in a specified location or locations across or down the sheet.

To use the replication facility you must specify two, and sometimes three parameters:–

a. The source range
b. The target range
c. Relationship of value reference. This tells VISICALC whether to simply copy each value into a new position or change it relative to its new location.

This feature is very powerful and the manual gives a full and clear description along with worked examples. Again, it's worth spending some time on this section because if you have not grasped this feature, it will impede the rest of your work.

SCREEN – FORMATTING

Even with the new 80 column screen, it is sometimes difficult to fit all your sheet onto one screen. It's a fair bet that every sheet will eventually exceed the limits of the screen. To see all the entries you will need to scroll up or down or left/right and this in turn means that some rows and columns will be lost from view. In a small sheet this may not matter because you might be able to remember your entry locations. But on a large sheet this can make life very difficult. Thoughtfully, Personal Software have catered for this; titles may be fixed vertically or horizontally or indeed in both directions.

However, you might also want to make a comparison between columns that are physically too far apart to be viewed simultaneously, and in this case the WINDOW function comes to the rescue. Each window may be scrolled independently or together and you may split the screen horizontally or vertically. When the program has been loaded from disk the standard entry position is defined as 9 characters wide.

This means that data above that maximum will be truncated in the entry position, but not in memory. The top line of the display will still show the full entry of that "cell". But you need not live with this limitation – the column width can be changed with the GLOBAL command.

Layout of the screen in the FORMAT command will also aid presentation, so that columns of figures line up and look tidy.

The INTEGER command will truncate the figures, but rounding only takes place on screen; memory retains the figure to eleven significant places. The "$" command displays figures in dollars and cents, whilst the "*" key will display values in a bargraph format. Entries may also be right or left justified so that they fit neatly under a lable. In addition, the GLOBAL feature will format to the above standards in the current window. It also permits recalculation, either by row or column, or automatically over the whole sheet, or only locally in the highlighted area. Finally, you may clear the screen of all entries or get out of the program; in each case confirmation is requested in case of accidents.

CONCLUSIONS

A very well thought-out program that is very user friendly. I am sure this program is going to be a very useful tool in many businesses, both small and large. And it's really good to see this degree of excellence and professionalism in material developed for Commodore. It has the added attraction that no knowledge of programming languages is needed for the operator. All this is available at a very modest price and must represent good value for money. The facilities available are very good and it is seemingly impossible to crash the system.

Parameters in BASIC subroutines and functions

This article, from the Department of Mathematics at Liverpool Polytechnic describes the various methods of communicating between BASIC programs and functions and subroutines, and shows how a simple machine-code extension to BASIC can provide extra facilities.

The value of subroutine and function facilities in a language is their capacity to be used, or called, from various parts of the program, and in association with this, their capacity to operate on different values at different calls. This makes it unnecessary to have copies of essentially the same sequence of program in various places. Communication between a subroutine or function, and the program which calls it, is achieved either by means of parameters, or by the use of global variables, i.e. variables which are accessible both to the subroutine and to the program calling it. The name of a function is itself a parameter, used to pass a value back to the calling program. In BASIC, the standard functions have parameters: User-defined functions can have one parameter, and can use global variables: but subroutines can only use global variables – all variables in BASIC are global.

There are several different classes of parameter required for program-subroutine communication. From the point of view of the program, these are:

a. A value, e.g. a number or string. This is passed to the subroutine for it to use. It can be the value of any expression, e.g. a numeric constant, simple or subscripted variable, function, or arithmetic expression.
b. A variable (simple or subscripted). This has an initial value, which may be used by the subroutine, and a final value,- which the subroutine can set. It provides a means of passing single-value results back to the program.
c. A data structure, e.g. an array. This can be used like a variable, except that the whole sets of values can be passed to and fro, e.g. for matrix operations.
d. A process, e.g. a function or subroutine name. This is used in the subroutine to generate values as required, depending on parameters supplied to the process by the subroutine, e.g. to generate values for numerical integration, or graph plotting.

BASIC standard functions (e.g. SIN, PEEK, LEFT$) all restrict themselves to accepting values as parameters, of the type appropriate to the function. Different calls of a function can use different parameters, e.g. SIN(X), SIN(Y). The result is a single value, passed back as the name of the function. There are no standard functions which have arrays or processes as parameters.

User-defined functions

These can accept one parameter, which must be a numeric value. The value is copied to the variable specified in the definition of the function, which can then be used in the expression which calculates the (numeric) value of the function. The expression can also use any other BASIC variable: it is the responsibility of the program to ensure that such variables have the correct values e.g. a function to translate and scale an X-coordinate of a point on a graph.

```
DEFFN X1(X) = A*X+B
X is the parameter, A and B are
  global variables
FOR I = 1 TON: X(I) = FNX1
  (X(I)): NEXT I
```

will alter all the values of the array X, by copying each value in turn to the simple variable X, and calculating the value A*X+B which is then assigned to the array element. The expression may use other functions, but these are fixed when the function is defined – they cannot be passed as parameters.

BASIC subroutines

In the sense, these do not exist! A subroutine is any part of the program called by a GOSUB instruction: it is the call which identifies the piece of program as a subroutine. Different values can be used by copying into an agreed global variable. Results are produced in fixed variables. Arrays can be used, but are restricted to an agreed array with a fixed number of dimensions and a fixed size, into which other arrays must be copied. A function can be used, by redefining it before each entry to the subroutine. A subroutine can be called from within a subroutine, but cannot be changed between calls, i.e. cannot be specified as a parameter. As an example, conconsider the following BASIC subroutine. It loads the elements 0-C of an array X with the values of a function F(Y) for Y = A to B, incrementing Y so that F(A) goes into X(O), and F(B) goes into X(C). The results are scaled by a factor S, and a constant M added to each one.

```
  100 H = (B-A)/C: REM INCREMENT
    BETWEEN VALUES OF Y
110 FOR I = O TO C
120 X(I) = S*FNF(A+H*I) +M
130 NEXT I
140 RETURN
```

To use this, X must be dimensioned to be as big as the biggest array to be generated. The function F must be defined, and appropriate values assigned to A, B, C, S and M. After the subroutine is called the result must be copied into the actual array where it is wanted, e.g.

```
10 DIMX(100), P(39), Q(79)
20 A=0 : B=2*π : C=39 : S = 12:M = 12
30 DEFFNF(Y) = SIN(Y)
40 GOSUB 100
50 FOR I = 0 TO 39: P(I) = X(I):NEXT
60 A = 1: B = 10: C = 79: S = 48: M = 0
70 DEFFNR(Y) = 1/Y
80 GOSUB 100
90 FOR I=OTO 79: Q(I)=X(I) :NEXT
```

These calls fill arrays P and Q with values for plotting SIN(X) on the screen in characters, and 1/X in pixels (¼-characters)

It is apparent that we could achieve the same effect in the subroutine if, instead of assigning values to A,B,C etcetera, we could replace A,B,C whenever they occur in the subroutine by the corresponding values. Sine BASIC is an interpretive language, this is fairly easy to do, by means of a machine-code extension, which persuades BASIC to switch backwards and forwards between processing the text of the subroutine,

and processing the text which we want to inset for a particular call. With this extension, the previous example can be rewritten as follows:

```
100 H =(C!)/E!
110 FOR I =O TO E!
120 D!(I) = F!*A!(B!+H*I)+G!
130 NEXT I
140 RETURN
```

The rest of the program becomes

```
10 DIM P(39), Q(79)
40 GOSUB 100! SIN! O! 2*π ! P!39!12!12!
70 DEFFNR(Y) = 1/Y
80 GOSUB 100!FNR!1!10!Q!79!48!0!
```

The array X, and the variables A,B,C,S and M are no longer required, it is not necessary to define SIN as a user function, or to copy the array X to the required destination array. In the subroutine, whenever a single-letter identifier followed by an exclamation mark is found by the BASIC interpreter, the sequence of characters between exclamation marks, corresponding to the letter, which follow the last GOSUB are processed instread. Thus after the GOSUB in line 40, A! will be replaced by SIN, B! by O, C! by 2* and so on as the subroutine is interpreted. After the GOSUB in line 80, A! is replaced by FNR, B! by 1, C! by 10 and so on. Note that it is the characters which are replaced, not the values: this can cause unexpected results if you are not careful, e.g.

```
10 GOSUB 100!2!3!X!
20 GOSUB 100!1+1!1+2!Y!
30 PRINT X,Y
40 STOP
100 C!=A!*B!
110 RETURN
```

RUN this program, and it will print 6, 4.

The reason is that if we expand line 100 by inserting the text of the parameters, we obtain

(a) X = 2*3 (as we might expect)
(b) Y = 1+1*1+2 (which produces the answer 4)

To guard against this, expressions to be used as parameters may be enclosed in brackets, e.g.

20 GOSUB 100! (1+1) ! (1+2) !Y!

which gives

(c) Y = (1+1)*(1+2) (which will give the correct answer).

The identifiers A!, B! etcetera appearing in the subroutine are known as *dummy parameters.* If a dummy parameter appears where an expression would be valid, the corresponding actual parameter may be an expression: anywhere else, it must be an appropriate identifier, e.g. simple or subscripted variable, or function name.

Actual parameters may be used which depend for their values on other actual parameters. This can be used to produce quite versatile routines, e.g. consider the following subroutine:

```
200 T=O:FOR C!=O TO D!:T=T+A!*B!:NEXT
210 E!=T: RETURN
```

This looks like a complicated way of evaluating A!*B!*(D!+1).

However it may be used in the program

```
10 S=5 : M=2:N=3:P=4
20 DIM A(S),B(S),T(M,N),U(N,P),V(M,P)
30 GOSUB 200!A(I)!B(I)!I!S!X!
40 FOR I=OTOM:FORJ=O TO P
50 GOSUB 200!T(I,K)!U(K,J)!K!N!V(I,J)!
60 NEXT J,I:REM MATRIX MULTIPLICATION
```

If you replace the dummy parameters in the first call, the resulting routine gives the inner product of the 1-dimensional arrays A and B in X, i.e. the sum of the products of corresponding terms. The second call produces the inner product of row I of array T with column J of array U, placing the result in element I,J of array V.

VARIABLE NAME

A facility is included, which was suggested by the articles by Danny Doyle on DIMP (CPUCN, Vol.2 No.8 and volume 3 number 2). It sometimes happens that it would be useful to be able to accept a variable name or an expression as data when the program is running e.g. to allow the user to specify a function which he wants plotting or, (for program testing) the names of variables to be displayed. In this case, the characters which are the actual parameter are held in a string variable, or are the result of a string expression. To deal with this case, if an actual parameter starts with a dollar sign, the remainder of the parameter is evaluated as a string expression. The string is then converted to BASIC symbols, and treated as the actual parameter, e.g.

```
10 INPUT A$, B$, C$
20 GOSUB 100!A$!$A$!
30 GOTO 10
100 PRINT A!,B!
110 RETURN
```

RUN the program

INPUT	RESULT
B$,222,333	B$ 222
B,C$, 333	B 333
SIN(π /4),222,	SIN(π /4)
333	.707106781
$AS,222,333	A [Program will hang up- press STOP]

In the last case, the actual corresponding to A is itself A, so that the parameter decoder will loop indefinitely.

The remaining facility makes it convenient to produce a subroutine which operates a variable number of parameters. If Z! is used as a dummy parameter in the subroutine, it must be followed by an integer expression in brackets. This will be treated as the parameter number i.e.

Z!(1) is equivalent to A!
Z!(2) is equivalent to B! etcetera

The number of actual parameters would normally be supplied as a parameter, e.g.

```
10 GOSUB 100!3!A!B!C!
100 T=O : REM SUBROUTINE TO SUM PARAMETERS
110 FOR I = 3 TO A!+1 : REM A! = NUMBER OF PARAMETERS FROM B!
120 T = T+Z!(I) : REM SUM C! ONWARD
130 NEXT
140 B!=T : REM RESULT TO B!
150 RETURN
```

The machine code extension to BASIC is supplied as a program which is LOADed and RUN in the normal way. It automatically relocates itself to the top of available memory, and adjusts HIMEM accordingly. It occupies about 512 bytes of store.

It is compatible with DOS 4.0. It is available from the Department of Mathematics, Liverpool Polytechnic on cassette tape with some demonstration programs, and a user specification for a handling charge of £5.

Extensions to BASIC syntax

Actual parameters may appear in a list following a line number in a GOSUB or ON-GOSUB statement. Dummy parameters may appear anywhere in a subroutine where an identifier is valid.

Parameter = any sequence of characters not commencing with comma or colon, terminated by exclamation mark other than within quotes.

Parameter-list = exclamation mark followed by one or more parameters, terminated by comma, colon or EOL immediately after a parameter.

Extended line-number = line-number followed by parameter-list.

Continued over

from page 21.

Extended line-number may replace line-number in GOSUB and ON-GOSUB statements. Spaces are ignored in and between parameters and terminators, except within strings of characters bounded by quotes.
If the last character of a parameter is a closing quote, it *must* be followed by at least one space before the terminating exclamation-mark.
If an EOL (end of line) occurs anywhere in a parameter list, the list is terminated. Any characters between the last exclamation mark and the EOL will be ignored.
Dummy-parameter = letter followed by exclamation mark, or Z! (expression).

Parameter action

When a GOSUB or ON-GOSUB followed by a parameter-list is interpreted, the location of the parameter-list is noted as the current parameter-list. A GOSUB not followed by a parameter-list has no effect – any existing current parameter-list remains in effect.
When a dummy-parameter is interpreted, the text of the paramater in the current parameter-list corresponding to the dummy-parameter is interpreted in place of the dummy parameter. Dummy-parameter A! corresponds to the first actual parameter, B! to the second, and so on. If there is no corresponding actual parameter (i.e. the list is too short or there is no current parameter-list), a SYNTAX ERROR occurs. If the first character of the actual parameter is $, the remainder of the parameter is interpreted as a string expression, the string is converted to BASIC symbols, and the result treated as the actual parameter. If it starts with $, the process is repeated. The STOP key is tested during this process, in case it forms a loop.

Errors are reported if the expression does not use up the whole of the actual parameter, or if the expression occupies more than 78 characters, or if the dummy parameter occurs in an INPUT statement. If the dummy-paramater is Z!, a numeic expression in brackets is expected immediately following it. The value of the expression is used as the parameter number, thus Z!(1) is equivalent to A! etcetera. When a RETURN corresponding to a GOSUB with a parameter-list is interpreted, the GOSUB stack is searched for the last preceding GOSUB with parameter list for which the corresponding RETURN has not been encountered. If one is found, the corresponding parameter-list is reinstated as the current parameter-list: if not, it is noted that there is no current parameter list. When a RUN command is obeyed, it is noted that there is no current parameter list.

The effct of this is that the last parameter list is always the one available, but as soon as the subroutine entered with this parameter-list terminates, the previous parameter list (if any) becomes accessible again. Subroutines entered by GOSUBS without parameter lists retain access to the one current at the time. An actual parameter cannot contain a reference to a dummy parameter: an exclamation-mark would be treated as the end of the actual parameter before it could be interpeted as the end of a dummy parameter.

Operation

a. Reset the PET.
b. Load any routines which provide extra instructions, such as KRAM or DOS.
c. LOAD the program PARAM, and RUN.
d. PARAM relocates itself to the current top of memory and adjusts HIMEM to leave room for itself. It links itself into the BASIC interpreter, and does a NEW to delete itself.
e. Parameter facilities are now in effect, and will remain so until the PET is reset.

Errors

The following errors are explicitly checked within PARAM:
a. Z! not followed by (expression) – SYNTAX ERROR
b. Expression value not in the range 0-255 – ILLEGAL QUANTITY
c. Expression not numeric – TYPE MISMATCH
d. Character string after actual parameter starting with $ does not form a valid expression – SYNTAX ERROR.
e. Character string after actual parameter starting with $ too long – more than 78 characters – STRING TOO LONG
f. Dummy parameter has no corresponding actual parameter – SYNTAX ERROR
g. Actual parameter starts with $ when dummy parameter is used in an INPUT statement – ILLEGAL DIRECT

Letters

Dear Sir,

The following is a way of linking Basic programs from disk, which is of course not possible using the toolkit.

It is assumed that the programs to be joined do not overlap on line numbers – GOTO's and GOSOB's will not work if they do. The two program files are then concentrated on the disk using the COPY COMMAND. This file is then loaded into the PET. Although both programmes are loaded a LIST command will display only the first program as the end of program markers are still present. To delete these proceed as follows. Type as a direct command :-
For A = 1024 To 8000: If PEEK (A) < > 0 or PEEK (A + 1) < > 0 THEN NEXT.

If the length of the first program is known then it will be quicker to put in a higher figure than 1024.

When the screen returns READY then type For B = A to A + 3: POKE B, 32: NEXT: POKE B, 0

The two programs are now joined but before they can be run the chaining of the second program must be corrected. To do this simply type 0 followed by a carriage return. All that now remains is to list the last line of the first program and carriage return over it. This is to remove the blank spaces which were inserted in joining the two programs together. This may not be necessary but the blanks may cause a SYNTAX ERROR if not removed.

The two programs are now joinned and ready for use.

Richard Brand
Chaldon, Surrey,

Dear Sir,

Thanks very much for your prompt reply (29.5.81). Ephraim is at this time in England, I do hope he brings "Basic Aid". In his Tel Aviv office they did not have it.

Enclosed I am sending you a small program which might be of interest to readers of the Newsletter.

I have converted my 3032/3040 set to Basic 4 and Dos 2.2, this is really

Communications Supplement

Communications is the name of the game at the moment - everyone seems to be getting involved in producing some kind of communications device. Commodore recognized the growing importance of this some while ago, and hot on the heels of releasing Communicator One and Commlink, appointed Rod Wellburn to be their communications manager.

As well as these two Commodore products there are a number of others on the market, and those of you who went to the PET Show will have seen the communications stands overflowing with weird and wonderful devices. Some of these will be mentioned in a while, but in the main body of the newsletter you can read a rundown on Commodore's new micro-mainframe, a microcomputer with many mainframe capabilities, including the capacity to communicate with other devices at speeds of up to 9,600 baud.

Perhaps the most significant "off the shelf" advance to date has been the BEE from B & B Computer in Bolton, Lancashire (tel. 0204 26644 for further information), which is a Commodore Approved Product - we get everywhere! The BEE revolves around Prestel, an excellent service in itself, providing more than 150,000 pages of information. Prestel, for the unfamiliar, is a visual link via a normal telephone to a central computer database, so that up to the minute information is available on-line whenever its required - it's as easy to operate as a hand-held remote control T.V. unit.

The BEE is the most recently approved Prestel adaptor, designed to interconnect any PET to a standard television set and British Telecomm Prestel. When the BEE is connected to a PET a retrieval command allows for page memory in the BEE (i.e. the contents of the Prestel page being looked at) to be dumped to an area of computer memory and then saved to disk or tape. Thus data can be accessed and re-displayed at any time, at no extra cost for each further viewing. This information can also be output to a printer if a hard copy is required.

The latest enhancement to both Prestel and the BEE is Telesoftware. This allows the down-loading and running of software stored in the Prestel data banks, and includes such things as business programs, games, educational and many other packages.

Packages exist for communicating with IBM, Univac, Honeywell, Burroughs, NCR, ICL, and many others. This is the area where the mysterious word "modem" often appears. So, what is a modem?

Often referred to as "Black Boxes" due to their technical nature, modems are designed and built by the local telephone authority (and others) so that the computer or terminal signal can be transmitted over the telephone authority's facilities. The word modem actually comes from the words MOdulator-DEModulator, and its basic function is as follows:- pulses are modulated onto a carrier and sent to the receiving modem, where they are demodulated and sent to the receiving equipment.

Commodore have produced their own modem, designated as the 8010. This is a 300 baud communications device, utilising standard telephone lines and handsets. The unit conforms to the CCITT standard, and comes complete with power pack and instruction manual. It is, of course, British Telecomm approved. By dialing the required telephone number, communication can be established by verbally setting up each end of the telephone connection and then placing the receiver in the coupler, or by using the auto-answer dial up modem.

The modem will operate in either ANSWER or ORIGINATE mode, allowing the flexibility of connection to most existing modems. Once communication is established, the carrier light will indicate that transmission and reception can begin. As an IEEE device connected via standard cables, the 8010 is easily programmed and is set up as a standard device, with an address number of 5 (in the same way that the printer is device4, the disk drive usually 8, and so on).

A number of programs are already on the market for the 8010, and it is expected that many more will be available shortly.

So, as you've probably gathered by now, this is a communications special. The main feature is an article from Dr. Philip G. Barker, principal lecturer in the department of Computer Science at the Teeside Polytechnic in Cleveland, to whom many thanks. It describes the problems associated with using a microcomputer as an interactive terminal, and outlines techniques for over coming these problems and gives summaries of some potential applicaton areas. Whatever your level of PET and communications knowledge, this is definitely worth reading. For the expert, Dr. Barker goes into great detail in many areas -for the beginner, he starts you off gently by describing computers in general and takes you through until the detailed areas appear, by which time you'll have a good grasp of what is going on, and will realise, as Mike Shingler of Kingston Computers put it, why 1981 is "The Year of Communications".

Using a Microcomputer as an Interactive Terminal

Dr Philip G. Barker

INTRODUCTION

A microcomputer consists of an input unit (usually a keyboard), a display system (such as a CRT screen), memory elements (RAM and ROM) and processing logic (one or more CPUs). These components are interconnected by means of an appropriate bus structure. Provided suitable interfaces are available this bus also permits the attachments of a variety of external devices. The most commonly used add-on peripheral are document printers, disk units, tape cassettes and modems.

Microcomputers are most often used as stand-alone computing facilities. However, because they have all the

necessary hardware elements available, they may also be employed to function as interactive terminal devices. If a microcomputer is to be used in this way some form of interface unit will be needed. The nature of the interface will depend upon the distances that separate the micro and the host to which it is to be attached. When the distances involved are relatively short, as might be the case when a local mainframe is used, very simple interfaces can be employed. However, when significant distances separate the devices - as in the case of connection to a remote mainframe -some form of modem is needed.

The advantages that accrue from being able to perform this type of computer inter-connection are many-fold. Amongst the more important of these are:

(1) the ability to access a far wider range of software than is available on a single isolated microcomputer.

(2) the ability to use the attached host as a powerful back-end data base processor enabling the archival, retrieval and sharing of both programs and data.

(3) The availability of more convenient program development tools such as editors, cross-assemblers, simulators and loaders. These permit software developed on the mainframe to be cross-loaded into an attached microcomputer for subsequent autonomous operation.

The ease with which these benefits may be derived will vary considerably with the nature of the computers involved and the kind of link that is to be implemented. This paper describes some of the problems that were encountered when a Commodore PET microcomputer was used as an interactive terminal device attached to a remote mainframe. It also outlines how these problems were overcome and summarises some of those application areas likely to benefit from this type of facility.

THE MICROCOMPUTER SYSTEM

In order to effect the operation of a microcomputer as a terminal device both hardware and software enhancements will usually be necessary. For the PET system, an arrangement of equipment similar to that shown in figure 1 is most often used. Communication with the remote computer installation is achieved over the Public Switched Network (PSN) using a conventional telephone receiver. This is connected via an acoustically coupled modem and a programmable bidirectional interface to the micrcomputer system -a 32K Commodore PET (Don80, CBM79). The modem (Tra8) supports transmission speeds up to 300 baud and operates via standard EIA RS-232 (or CCIT V.24) signals.

The Programmable Bidirectional Interface

The external bus struture of the PET is based upon two kinds of interface:

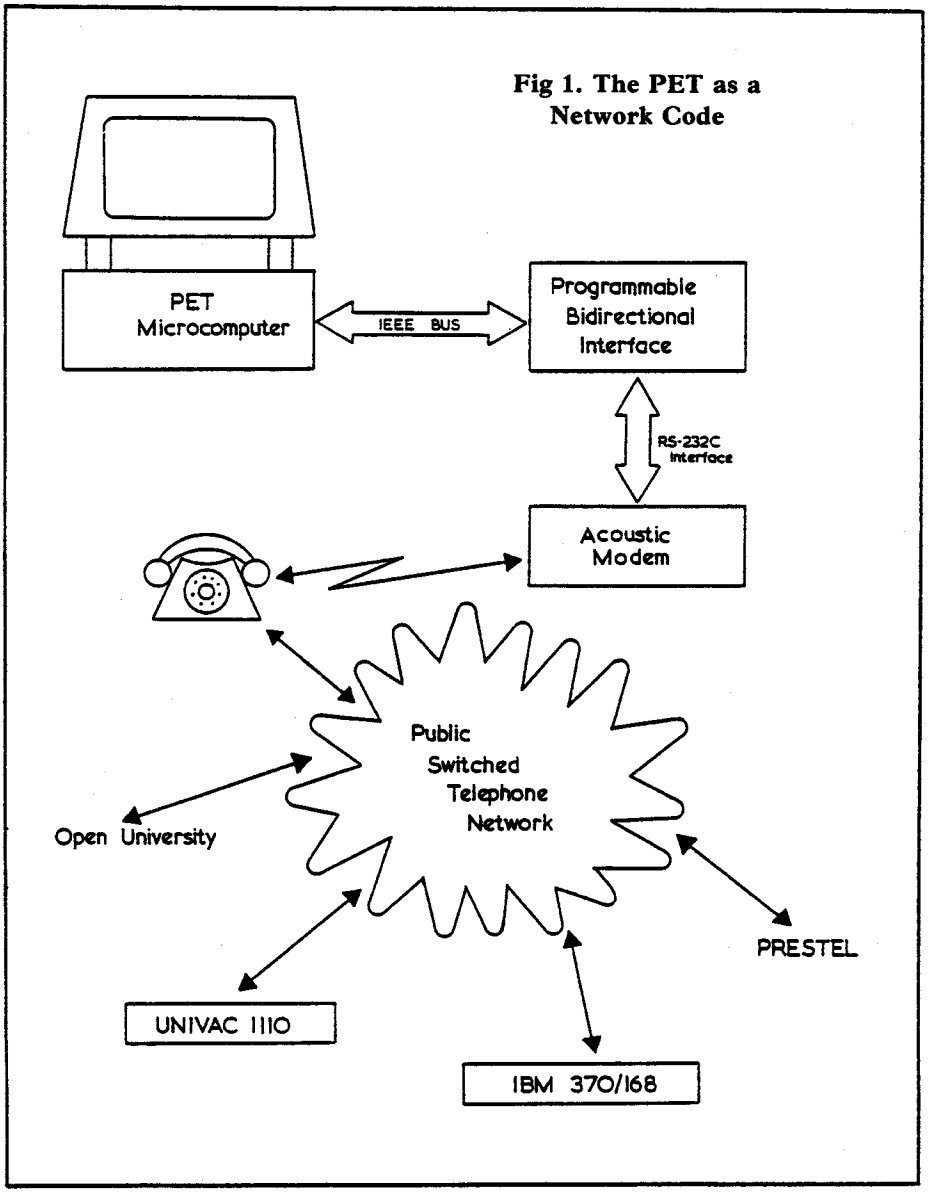
(a) an implementation of the IEEE-48 standard (Fis80), and

(b) a non-standard User-Port connection (CBM79).

Most peripheral devices are attached to the computer via the IEEE-4888 port. This procedure is adopted becauce of the availability of a variety of firmware (ROM) routines that service the I/O instructions written in a programming language (BASIC, PASCAL or Assembler). These routines produce synchronised bus signals that control the I/O activity of the devices that are connected to the IEEE Port. Many commonly available peripherals, including modems for terminals, operate acording to the the RS-232 standard. Consequently, before such devices can be connected to the PET, some means of interconverting between IEEE-488 and RS-232 conventions is necessary. A variety of hardware circuits exist for achieving this (HAM80). We have used a buffered 80 character programmable bidirectional interface constructed by Small Systems Engineering Ltd. (Sma80).

The interface is described as programmable since its communication characteristics can be set up and changed under program control. This is effected be sending it a control

Fig 1. The PET as a Network Code



have/lack certain special features. Their relationship to a computational process running on the mainframe is shown in figure 3. Effectively, the device support routines 'smooth out' the differences between terminal devices so that, as far as the computational process is concerned, they all look alike.

Given that such routines exist, there are two ways in which the mainframe facilities can be used to 'personalise' a PET terminal. One involves issuing commands to the device support routines manually, while the other arranges for these to be isued automatically when the user logs onto the remote computer.

As an example, consider communication with the remote IBM 370/168. The PET establishes a connection with the remote host. Then, the user types a sequence of commands that gives the associated device support routine information about the terminal it is currently servicing. A typical sequence of commmands is shown in the box below - appropriately annotated on the right.

```
%ATC = ]    - defines the attention interrupt character,
%DPC = ←    - specifies the delete previous character code,
%DLC = ↑    - specifies the delete line character,
%EFC = \    - defines the end-of-file character.
```

The effect of these commands is quite straightforward. They associate special function with certain of the PET keys. Thus, each time the PET user presses) key on his keyboard an attention interrupt will be generated in the mainframe. Similarly, depression of the / key will cause an end-of-file sequence to be produced. During a terminal session the user is free to allocate these special functions to other keys if he needs to. For example, having 'personalised' his PET with the commands listed above if he now types the command &ATC = @ this causes the @ key to become the attention interrupt key thereby releasing) for its normal function.

An alternative method of personalising the remote PET might involve creating a 'Personality File' in the file store of the remote host. This file would contain a series of commands (similar to those listed above) which would automatically be invoked as soon as the PET logged onto the system as a remote terminal. This approach could be used to make the existance of device support routines in the host totally transparent to the user.

Limitations imposed by Processing Speed

Using the scheme outlined above the PET becomes a more usable remote terminal. However, there are still some problems that need to be overcome. The program presented in figure 2 works reasonably well provided that lower case characters are not present in the messages sent to or received from the remote mainframe. Should this not be the case, code conversion problems can arise. Suppose the following messages sequence was sent from the host:

```
>"SOURCE*TO DOES NOT EXIST.        ..... (A)
>Enter replacement or "CANCEL".    ..... (B)
```

This would appear on the screen of the PET in the following way:

```
>"SOURCE*TO" DOES NOT EXIST.       ..... (C)
>E.4%2 2%0,1#%-%.4 /2 "CANCEL".    ..... (D)
```

In order to understand the somewhat cryptic appearance of line D the ASCII code values for the character of the message need to be examined. A comparison of the codes for the transmitted and received characters in the first word message B is shown in the following table:

Message Character	ASCII Value (Octal)	Ignore High Order Bit	Resultant Character
E	105	05	E
n	156	56	.
t	164	64	4
e	145	45	%
r	162	62	2

It can be seen that the mysterious appearance of line D arises because the high order bit of the ASCII code for lower case characters appears to have been lost. This phenomenon arises because of the special way in which the screen memory of the PET microcomputer drops bit 6 of the standard ASCII value in order to produce a six bit code for its keyboard characters (CBM79). The problem can be overcome by adding some additional statements to the program presented in figure 2. Modifications similar to those shown below can be used to provide the lower case capability needed to overcome the encryption problem outlined above.

```
135  POKE 59468, 14
       o
       o
315  IF A$ = "" THEN : RETURN
316  C=ASC(A$)
317  IF (C>64) AND (C<91) THEN A=128
318  IF (C>96) AND (C<128) THEN A=-32
319  A$ = CHR$(C+A)
       o
       o
```

The POKE statement (line 135) sets the PET into alternate character set mode - upper/lower case rather than upper case/graphics. The extra statements at lines 315 through 319 are included in order to compensate for the effect of the POKE statement on the way in which the ASCII code values are interpreted by the PET.

In POKE 12 mode (normal) the decimal value 65 represents the letter "A". However, in POKE 14 mode this value corresponds to "a" while "A" is produced by the value 193 (that is, 128+65). The modification shown in lines 315 through 319 makes it possible for the PET to interpret both upper and lower cases characters correctly.

Unfortunately, the additional computational overhead associated with these extra statements introduces a further problem. When long message strings are sent from the host (for example, when listing a file), the speed of the modified program becomes too slow to handle them. Communication between the mainframe and remote terminal is asynchronous (Heb75). Each character transmitted consists of a start bit, seven data bits, a parity bit and one stop bit - that is, ten bits in total. Thus, at 300 baud the mainframe transmits 1 character every 33.33 millisec. If the remote terminal cannot process data quickly enough, then information is likely to be lost unless some form of buffering and/or mechanism for delaying mainframe transmission (flow control) is used.

The programmable interface between the modem and the PET has the capability of buffering 80 characters. Furthermore, when the buffer becomes full the interface should pass a signal to the mainframe which stops it transmitting - thereby preventing loss of information. However, if the mainframe chooses to ignore this signal, the interface fails to send it, or, if it is allowed to 'float', then information will become lost through buffer overflow. This phenomenon has been observed when the equipment shown in fiqure 1 is used in conjunction with the modified BASIC program described above.

Suppose R2 is the rate of transmission from the mainframe and R1 the rate at which the PET processes the data in the buffer. An indication of

the message size at which information loss will start to occur can be gained from the expression:

$$\text{Message Size} = \frac{80.R2}{R2 - R1}$$

The results of some simple timing experiments are shown in fiqure 4. The basic time for the original subroutine (lines 200 through 330 in figure 2) to service a character sent from the mainframe is about 31 millisec. The additional overhead added to this routine by the code conversion statements is about 41 millisec/character. It is easy to see that providing a lower case capability more than doubles the time it takes to process each character received from the mainframe.

On the basis that transmission from the mainframe (R2) takes place at a rate of 30 characters/sec and the modified basic program is able to process characters from the buffer at a rate (R1) of 13.83 characters/sec, it is easy to compute an upper limit to the message size that can be handled without information loss. This works out to be about 150 characters. Actual experiment based upon transferring messages of known size indicate that information loss starts to take place for messages in excess of 160 characters. Bearing in mind the 'crudeness' of the timing experiments, the agreement between prediction and observation is reasonably good. Indeed, in order to compute the 'correct' answer the value of R1 only needs to be increased to 15 characters/sec, which means that the difference between estimated and actual values is within 10%. This problem can be easily overcome by speeding up the rate of processing in the PET. Using machine code programs is one way of acomplishing this. Indeed, when the modified program of figure 2 is replaced by an equivalent machine code program, no problems are experienced.

Providing a Cursor

A cursor is commonly used to indicate the position on the screen at which the next character - typed by the user or received from the mainframe - is to appear. There are four basic types of cursor. Different types are distinguished by their shape and whether they are opaque or transparent, static or flashing, hardware or software controlled. A transparent flashing hardware controlled (that is, interrupt driven) cur-

Fig 3. Role of device support routines

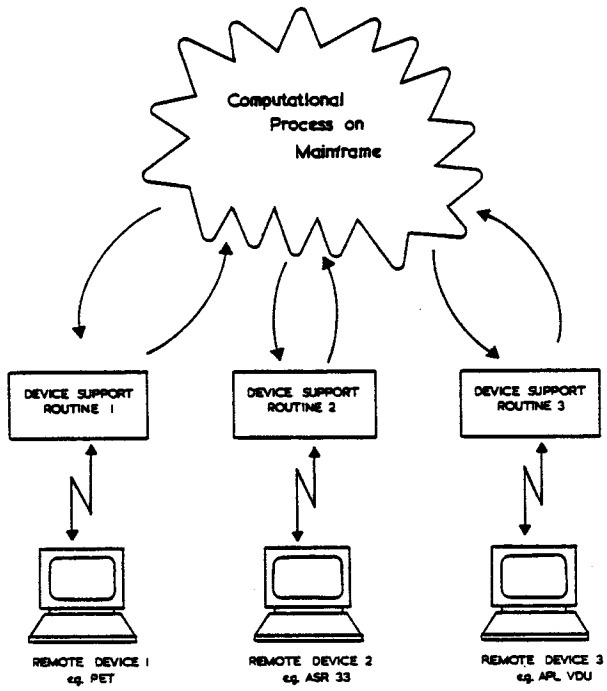


Fig 4. Results of timing experiments.

(1) Driver Routine Blank

```
100  POKE 59468, 14
110  K=TI
120  FOR I = 1 TO 100
130  A$ = CHR$(65)
140  NEXT I
150  PRINT TI-K
160  PRINT A$
170  END
```

Time = 39 Jiffies

(2) Subroutine Linkage Timing

```
100  POKE 59468, 14
110  K=TI
120  FOR I = 1 TO 100
130  A$ = CHR$(65)
135  GOSUB 300
140  NEXT I
150  PRINT TI-K
160  PRINT A$
170  END
300  REM *** GET MAINFRAME CHAR ***
310  RETURN
```

Time = 57 Jiffies

(3) Print Statement Overhead

Lines 100 through 170 of (2)

```
300  REM *** GET MAINFRAME CHAR ***
310  PRINT A$
320  RETURN
```

Time = 200 Jiffies

(4) GET Timing

Lines 100 through 170 of (2)

```
300  REM *** GET MAINFRAME CHAR ***
310  GET A$ : IF ST=2 THEN : RETURN
320  PRINT A$
330  RETURN
```

Time = 203 Jiffies (no character input)
= 252 Jiffies (with character input)
Average = 227 Jiffies

(5) Code Conversion Timing

Lines 100 through 170 of (2)

```
300  REM *** GET MAINFRAME CHAR ***
310  IF A$ = "" THEN : RETURN
320  C = ASC(A$)
330  IF (C>64) AND (C<91) THEN A = 128
340  IF (C>96) AND (C<128) THEN A = -32
350  A$ = CHR$(C+A)
360  RETURN
```

Time = 303 Jiffies

Basic time to service a character from mainframe = $\frac{(227-39)}{100} \cdot \frac{1}{60}$ = 31.3 millisec
(Results 4 and 1)

Additional overhead in performing code conversion = $\frac{(303-57)}{100} \cdot \frac{1}{60}$ = 41.0 millisec
(Results 5 and 2)

sor is the type which is normally implemented on the PET when it is in direct command or INPUT modes.

Unfortunately, when the PET is programmed to obtain input from the keyboard with a GET statement, no flashing cursor is displayed. However, it is possible to turn it on for this mode of data entry by executing a POKE 167, 0 statement prior to the input transaction (Ham80, Don80). The program listed in figure 2 can thus be modified quite easily in order to provide a flashing cursor facility. Inclusion of the statement

21 POKE 167,0

is all that necessary. If such a modification is made, a cursor now appears, but, as user-computer dialogue proceeds, the appearance of the microcomputer screen becomes ergonomically unacceptable. Static images of the cursor ("blobs") remain deposited at what would seem to be randomly selected positions on the screen. In fact, these appear at some of the cursor location corresponding to the receipt of a carriage return character - from the keyboard or from the mainframe. The particular points at which they occur correspond to instants at which synchronisation between the BASIC program and the cursor handling system is lost. There is an easy way of removing the blobs by the addition of some extra BASIC statements that ensure a space character is deposited at the cursor position when a carriage return code is received. However, like the code conversion routines described in the previous section, the computational overhead of employing such code is prohibitive.

OVERCOMING THE PROBLEMS

The easiest solution to overcoming the problems outlined in the previous sections is to write the software - that listed in figure 2 and the various amendments - in assembler. Bearing in mind what has been said, the basic alogorithm to be implemented is as follows:

```
Begin:  Set up the non-maskable interrupt vector to handle
        the RESET button.
Step1:  Get a character from the keyboard.
Step2:  If no character then jump to Step4.
        If a cursor control character
           then ignore it (Jump to Step4).
Step3:  Send character to mainframe.
Step4:  Get character from mainframe.
Step5:  If STATUS = 2 then jump to Step1.
Step6:  Perform code conversion (upper/lower case).
Step7:  If carriage return character (hexadecimal OD)
           then over-write the "blob".
        Print the character on the screen.
Step8:  Jump to Step1 and repeat cycle.
Last:   Reset default input/Output device codes.
        Jump back to BASIC interpreter.
```

The machine code implementation of this algorithm was developed on a cross-assembler for the MCS650X range of microcomputers (Lub80). This was available on one of the back-end mainframe machines (IBM 370/168). The development system that was used is similar to that depicted in figure 5. Assembler source language statements were stored in a mainframe file called INPUT. The contents of this file could be modified in various ways by means of the system editor During an assembly, the cross-assembler read the statements contained in INPUT, checked their validity and generated appropriate object code which was stored in the file OUTPUT1. At the same time, a listing of the source file (and appropriate diagnostics) was sent to the file OUTPUT2. This could later be listed on a system printer or on a local print device. Alternatively, this output could be produced directly on the screen of the PET. A typical listing of the final version of the assembler program (produced on a local printer) is shown in figure 6.

In order to use this program it is necessary to provide a simple prologue routine written in BASIC. An example of such a routine is given below.

```
10  OPEN 1,4    : REM OUTPUT CHANNEL
20  OPEN 2,6    : REM INPUT CHANNEL
21  POKE 167,0  : REM TURN ON CURSOR
30  PRINT#1, CHR$(255)"FXXGA" : REM SET UP INTERFACE
40  POKE 59468,14 : REM TURN ON LOWER CASE
50  SYS 8192    : REM JUMP TO ASSEMBLER ROUTINE
```

Use of this combination of programs enables all the previously described problems to be overcome. The prologue code is written in BASIC rather than assembler so that the end-user can easily modify those parts of it which he is likely to want to change -external device addresses, cursor on/off status and interface details. The assembler routine disables all the cursor control keys in order to avoid spurious side affects. The PET's **RUN STOP** and **RVS** keys are not treated in this way. In the system which we use the **RUN STOP** key is used to produce an attention interrupt in the mainframe, while **RVS** is assigned the task of generating and end-of-file character for data entry from the terminal. The **RESET** button on the PET produces a local interrupt causing control to be passed back to BASIC. When this happens, the PET can be made to function as a stand-alone microcomputer and thereby run BASIC (or assembler) programs - provided they do not interfere with the prologue code listed above. As an example of this, suppose the PET contained the following program

```
1000  FOR I = 1 TO 100
2000    PRINT I, I*I, I↑3
3000  NEXT I
4000  STOP
```

in addition to the original prologue routine (lines 10 through 50). The effect of the following user directives,

(1) Type: RUN
(2) Press RESET button
(3) Type: RUN 1000
(4) Type: RUN

would be to set up communication with the mainframe (step 1), then, at a later stage, logically sever the link (step 2), initiate the execution of a local application program running of the PET (step 3) and then re-establish communication with the mainframe (step 4).

The assembler routine is 181 bytes long and could thus easily fit into one of the tape cassette buffers, therby leaving the whole of the remaining memory space available for other purposes. Although developed for use on a 40-column PET, the programs will also work on the newer 80-column (8000 series) machines (CBM80a CBM80b). However, in the latter case the return address to BASIC (warm start) would need to be changed from $C389 (BASIC 2.0) to $B3FF (BASIC 4.0). In addition, it would be desirable (although not necessary) to modify the assembler routine to handle the additional keys present on the extended keyboard. When the software outlined above is used in conjunction with the 8000 series PET a powerful interactive terminal facility can be constructed.

SOME APPLICATIONS

There are many application areas where the ability to combine a local microprocessing capability with a remote back-end mainframe facility can be usefully employed. The desirability, cost and amount of effort involved in implementing this mode

of computing will depend critically on the nature of the link between the mainframe and the micro. In the work that has been described here, the Public Switched Network has been employed. Direct links or private dedicated lines invariably provide a better quality communication facility but are often expensive unless line utilisation is high. The use of lower quality "dial-up' links puts an extra overhead on the processing sofware -the need to perform appropriate error checks in order to ensure that data transmitted over the link is not peturbed by extraneous noise. Consequently, in each of the following application examples, although it is not explicitly described, extensive error checking procedures would need to be implemented.

The examples that are described in this section represent perfectly general techniques - that is, they could be employed with any microcomputer system. However, for the purpose of illustration, they are described in terms of a particular machine - the Commodore PET.

Cross-loading of Machine Code Programs

Applications involving this type of operation fall into two general categories:

(a) those in which the 'intelligent' terminal functions as the target machine, and

(b) situations where it acts as intermediate link in a hierarchical chain.

The first of these involves straightforward cross-loading of machine code programs from the mainframe to the microcomputer terminal. This facility can be used to substantially improve its functionality, thereby making it far more useful than a conventional interactive terminal. In the second type of application the intelligent micro-terminal acts as a buffer between the mainframe and the target machine. Code to be cross-loaded is first passed to the micro-terminal where it can be checked for transmission errors and stored. Then, in a subsequent step, it can be sent from the intelligent station across to the target machine.

As far as the Commodore PET is concerned, each of these categories of cross-loading is fairly easy to perform. Consider the case of direct loading. Programs developed using a system similar to that shown in figure 5 are easily transferred from the mainframe. The object code produced by the cross-assembler is stored in a file containing a series of fixed format records. Each record contains

(a) a value, N, that gives the number of program bytes in the record,

(b) a load address that specifies where these bytes are to be loaded in the memory of the target machine,

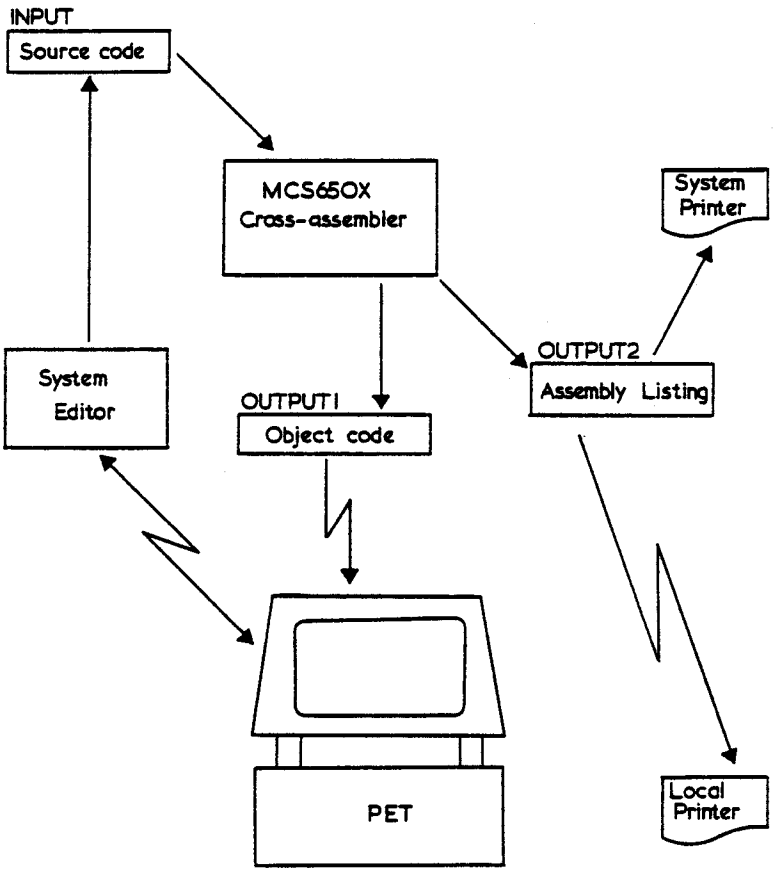
(c) the N consecutive program bytes, and

(d) a two byte checksum value.

Knowing the format of these records, it is a simple matter to formulate and implement an algorithm to perform the cross-loading operation (Bar81).

Similarly, it is equally easy to implement cross-loading in situations where the PET-based intelligent terminal acts as a mode in hierarchical link. In this context we have been experimenting with systems in which code that is transferred to the PET is subsequently cross-loaded into other PETS, a Sinclair ZX81, a Texas Instruments TMS 9980A 16-bit machine, and a variety of National Semiconductor SC/MP based systems.

Fig 5 Cross-assembler Development System



In-house Viewdata Systems

The simplest form of viewdata system consists of a series of 'pages' of information stored in the form of a computer data base. In response to queries made by users, appropriate pages of information are transmitted to viewdata terminals distributed over some particular locality or region. Both national and in-house systems currently exist. As business tools the attractiveness of these systems stems from the ease with which information can be retrieved and the colourful and graphic way in which it can be presented.

Adaptors are available commercially (Bee80) that enable the PET to be used as a terminal to the UK's national viewdata system (PRESTEL). Typically, such adaptors connect the microcomputer to a standard domestic colour TV and an auto dialer built-in modem. Unfortunate-

Fig 6 Continued

```
206A 40 B220      69        CMP STEP7
206D C9 5B        70 TEST1  CMP #£60      ; IS CHAR LESS THEN BL?
206F 80 10        71        BCC ADD       ; YES THEN ADD ON 123
2071 C9 6C        72        CMP #£60      ; IS CHAR GREATER THAN 96
2073 F0 1D        73        BEQ STEP7
2075 B0 03        74        BCS TEST2
2077 4C 9220      75        JMP STEP7
207A C9 90        76 TEST2  CMP #£SC      ; IS CHAR LESS THAN 12S?
207C 90 0D        77        BCC SUB       ; YES THEN DEDUCT 32
207E 4C 9220      78        JMP STEP 7
2081 D3           79 ADD    OLD
2082 16           80        OLD
2083 69 80        81        ADC #£20      ; ADD ON 12S
2085 8D AB20      82        STA CHAR      ; STORE RESULT BACK
2088 4C 9220      83        JMP STEP7
208B D8           84 SUB    OLD
208C              85        SEC
208D E9 20        86        SBC #£20      ; DEDUCT 32
206F 8D AB20      87        STA CHAR      ; STORE RESULT BACK AGAIN
                  88 ;
2092 20 CCFF      89 STEP7  JSR RFILES    ; SET DEFAULT DEVICES
2095 AD AB20      90        LDA CHAR      ; GET CHARACTER TO BE PRINTED
2098 CS 0D        91        CMP #£0D      ; IS IT RETURN?
209A DP 09        92        BNE PRINT     ; NO ITS NOT
209C A4 CS        93        LDY £CS       ; STORE A SPACE
209E A9 20        94        LDA #£20      ; IN POSITION OF CURSOR
20A0 B1 C4        95        STA (£C4),Y   ; TO AVOID THE BLCB
20A2 AD AB20      96        LDA CHAR      ; GET CHAR TO BE PRINTED
20A5 20 D2FF      97 PRINT  JSR PRTCHR    ; GO PRINT CHAR IN ACCUMULATOR
                  98 ;
20A8 4C 0D20      99 STEPS  JMP STEP1     ; GO BACK AND START LOOP AGAIN
                 100 ;
20AB 00          101 CHAR   DB  £0        ; PLACE TO STORE CHARACTER
20AC AE20        102 LAST   ADDR  *+2     ; DEFNE ADDRESS OF NMI HANDLER
                 103 ;
20AE 20 00FF     104        JSR RFILES    ; SET DEFAULT DEVICES
20B1 4C B9C3     105        JMP BASIC     ; GO BACK TO BASIC WITH "READY"
                 106        END
```

```
ADD      2081     79     71
BASIC    0389     10    105
BEGIN    2000     14
CHAR     20AB    101     25    58    64    82    87    90    96
GETCHR   FFE4      6     22    57
INFILE   FFC6      8     36
LAST     20AC    102     15    18
NOKEY    2043     48     27    29    31    33    35    37    39    41
                         43    45
OTFILE   FFC9      7     52
PRINT    20A5     97     92
PRTCHR   FFD2      9     53    97
RFILES   FFCC      5     21    89   104
STEP1    200D     21     62    65    89
STEP2    2013     24
STEP3    2046     51     46
STEP4    204E     55     24    48
STEP5    2059     60
STEP6    205F     64
STEP7    2092     89     37    39    73    75    79    83

STEP8    20AB     99
SUB      2088     84     77
TEST1    206D     70     66

TEST2    207A     76     74

MOS Technology MCS650X Assembler (AN240) done at 16:07:41 on 05-07-91.

0 error(s) detected.

Cards: 106   Symbols:        23   Cost:     £0.09
Punch :  0   References:     46   CPU Time:  0.73
Print : 137  Storage:         6
```

excellent for file-handling. But after a very short time, it became apparent, that there should be a possibility to inspect and edit relative files directly on disk – the result was the enclosed program.

Although my PET is of the graphic type, I "Poke"-d it into lower case (line 7000), printed once to secondary address 7 on the Printer, and listed the program. The printer evidently did not like this and printed brackets instead of vertical lines in lines 210-250! By the way, the spaces in lines 500, 510, 620, 630 are shifted.

P.R. Gabor
Gabotron Ltd
Israel

```
ready.

 10 rem ***********************************
 20 rem *                                 *
 30 rem *          f i x i t              *
 40 rem *                                 *
 50 rem *  editor for relative files       *
 60 rem *             1/6/81              *
 70 rem *                                 *
 80 rem ***********************************
 90 rem
100 z=1/254
200 s$(0)="   ■┌──┬──┬──┬──┬──┬──┬──┬──┬──┬──┐
210 s$(1)="   ■]   ] 0]1]2]3]4]5]6]7]8]9]
220 s$(2)="   ■├──[──[─[─[─[─[─[─[─[─[─┤
230 s$(3)="   ■]   ]■   ] ] ] ] ] ] ] ] ]
250 sm$="   ■├──[■──[─[─[─[─[─[─[─[─[─┤
260 sb$="   ■└──┴■──┴──┴──┴──┴──┴──┴──┴──┘
300 ac$="■■":d$="■■■■":sp$="     ":ad$="■■■■"
310 fork=1to5:sp$=sp$+sp$:ad$=ad$+d$:ac$=ac$+ac$:next
320 c7$=left$(ac$,7):sp$=sp$+left$(sp$,100)
330 pp$="____________________":tm$=chr$(13)
450 gosub7000
500 input"■input filename    ■■■■";a$
510 ifa$=" "thenprint"■";:end:          rem     just hit return to exit
520 ifa$<>"*"thennf$=a$
570 input"■length of record";rl
580 dopen#1,(nf$),l(rl),d0
590 ifrl>90then2000
600 fb=1:lb=rl:o$="■─"
610 gosub5000:                          rem     print grid
620 gosub950:input"record #    ■■■■";r$
630 gosub1000:ifr$=" "thendclose:goto500
640 rec=val(r$)
650 record#1,(rec):gosub950
660 gosub5400:                          rem     read record
680 print"record # "rec;
690 gosub5200:poke158,0:                rem     write on screen
700 geta$:ifa$=""then700:               rem     edit
710 ifa$="■"then4000
720 ifa$=chr$(20)then4050
730 ifa$="■"then4100
740 ifa$="■"then4150
750 ifa$="■"then4200
760 ifa$="■"then4250
770 ifa$="@"thenprinto$:goto620
780 ifa$="■"thenrec=rec+1:printo$:goto650
790 ifa$=chr$(34)thena$="@":goto4300
800 ifa$=chr$(141)then4800
810 ifa$=chr$(13)thena$="■":goto4300
850 ifa$="■"then590
860 ifa$="■"thendclose:print"■";:end
890 goto4300
930 rem
935 rem ******************************
940 rem * cursor to line # 24        *
945 rem ******************************
```

Continued over

```
948 rem
950 print"■"ad$left$(sp$,30)"■"ad$:return
955 rem
985 rem ******************************
990 rem *     check errors             *
995 rem ******************************
998 rem
1000 ifds<20thenreturn
1010 print"■"ds$:dclose:end
1970 rem
1975 rem ******************************
1980 rem * set range for inspection *
1985 rem ******************************
1988 rem
2000 input"■first byte to be
                        viewed   ■■■■";a$
2010 ifa$=" "thendclose:goto500
2020 fb=val(a$)
2030 ifrl-fb<81thenlb=rl:goto610
2040 input"nr of bytes to be viewed";a$
2050 a=val(a$):ifa>80thenprint"range too
                              large!":goto2040
2060 iffb+a>rlthenlb=rl:goto610
2070 lb=fb+a-1:goto610
3980 rem
3985 rem ******************************
3990 rem *         insert               *
3995 rem ******************************
3998 rem
4000 b2$=left$(" "+b2$,len(b2$))
4010 gosub5200:goto700
4030 rem
4035 rem ******************************
4040 rem *         delete               *
4045 rem ******************************
4048 rem
4050 b2$=mid$(b2$,2)+" "
4060 gosub5200:goto700
4080 rem
4085 rem ******************************
4090 rem *        crsr left             *
4095 rem ******************************
4098 rem
4100 iflen(b1$)=0orbn=fbthen700:        rem
                        cursor in first pos
4110 b=1:gosub4700
4120 goto700
4130 rem
4135 rem ******************************
4140 rem *        crsr right            *
4145 rem ******************************
4148 rem
4150 iflen(b2$)=0orbn=lbthen700:        rem
                        cursor in last pos
4160 b=1:gosub4750
4170 goto700
4180 rem
4185 rem ******************************
4190 rem *        cursor up             *
4195 rem ******************************
4198 rem
4200 iflen(b1$)<10orbn<fb+10then700:    rem
                    no higher line available
```

```
4210 b=10:gosub4700
4220 goto700
4230 rem
4235 rem ******************************
4240 rem *        crsr down             *
4245 rem ******************************
4248 rem
4250 iflen(b2$)<10orbn>lb-10then700:    rem
                    no lower line available
4260 b=10:gosub4750
4270 goto700
4280 rem
4285 rem ******************************
4290 rem * print a$ & move marker  *
4295 rem ******************************
4296 rem
4300 printa$"■";:b2$=a$+mid$(b2$,2)
4310 goto4150
4680 rem
4685 rem ******************************
4690 rem *  sbr for crsr left         *
4695 rem ******************************
4698 rem
4700 printo$:bn=bn-b:gosub5500
4710 b2$=right$(b1$,b)+b2$
4720 b1$=left$(b1$,len(b1$)-b)
4730 return
4735 rem
4740 rem ******************************
4745 rem *  sbr for crsr right        *
4748 rem ******************************
4749 rem
4750 printo$:bn=bn+b:gosub5500
4760 b1$=b1$+left$(b2$,b)
4770 b2$=mid$(b2$,1+b)
4780 return
4785 rem
4790 rem ******************************
4795 rem * replace record (enter)    *
4798 rem ******************************
4799 rem
4800 printo$:gosub5600
4810 rec=rec+1:goto650
4980 rem
4985 rem ******************************
4990 rem *  print grid on screen      *
4995 rem ******************************
4998 rem
5000 print"■"
5010 fork=0to3
5020 printa$(k):next
5030 fork=int(fb/10)tolb/10
5040 n$=right$(str$(k),2)
5050 gn$="  ■]"+n$+"]■  ] ] ] ] ] ] ] ] ] ]
5060 printgn$:printam$:next
5070 print"■"gb$:return
5180 rem
5185 rem ******************************
5190 rem *    put b2$ on screen       *
5195 rem ******************************
5198 rem
5200 bn=len(b1$)+1:gosub5500:           rem
                            position cursor
5210 print"■■■";:fork=1tolb-bn+1
```

Pub Bit

In mentioning the Oak and Saw, my local, last time around, I appear to have missed something out. Two things actually, so to redress the record, here we are again. The first thing I did was to miss out the landlord's name, so Mike Pepper, consider yourself in print. This next one's under duress, becuase although she may be small she carries a mean punch. Sue Wilkinson, the barmaid who has served me "n" million pints of John Courage over the last year or so, is alas leaving for the wilds of Birmingham shortly. So, lovely Sue, all the best, and I'll buy you a drink on Saturday, okay ?!

```
5220   by$=mid$(b2$,k,1)
5230   printby$"■";:ps=ps+1
5240   ifps=10thenps=0:printtm$tm$c7$;:rem      new line
5250   next
5260 gosub5500
5270 return
5375 rem
5380 rem ****************************
5385 rem * read record into a1$      *
5390 rem * divide into b1$ & b2$     *
5395 rem ****************************
5398 rem
5400 gosub1000:a1$="":fork=1tor1
5410   get#1,b$
5420   ifb$=chr$(13)thenb$="ℕ"
5430 ifb$=chr$(34)thenb$="@"
5440 a1$=a1$+b$:ifst=64thenk=r1
5450 next:b1$="":a=len(a1$)
5455 ifa<r1-1thena1$=a1$+left$(sp$,r1-a)
5460 iffb>1thenb1$=left$(a1$,fb-1)
5470 b2$=(mid$(a1$,fb))
5475 return
5479 rem
5480 rem ****************************
5485 rem * position cursor to bn &   *
5490 rem * print "⊥" on grid         *
5495 rem ****************************
5498 rem
5500 a=int(bn/10+z):r=a-int(fb/10+z):  rem      row number
5510 ps=int((bn/10-a)*10+z):          rem      col number (position)
5520 r=6+2*r:c=7+2*ps
5530 print"■"left$(ad$,r)left$(ac$,c);
5540 print"■■";
5550 return
5575 rem
5580 rem ****************************
5585 rem * read b1$&b2$ into record  *
5590 rem ****************************
5595 rem
5600 record#1,(rec):gosub1000:b$=""
5605 a1$=b1$+b2$
5610 fork=1tolb
5620   a$=left$(a1$,1):a1$=mid$(a1$,2)
5630   ifa$="ℕ"thena$=chr$(13)
5640   ifa$="S"thena$=chr$(34)
5650   b$=b$+a$
5660   next
5670 print#1,b$;:gosub1000
5680 return
6975 rem
6980 rem ****************************
6985 rem *       instructions       *
6990 rem ****************************
6995 rem
7000 poke59468,14:print"■"tab(12)left$(pp$,9)tm$tab(12)"■* FIXIT *"
7010 printtm$"This is an Editor for Relative Files.
7020 print"Files having more than 90 bytes can be
7030 print"edited in sections of up to 80 bytes."tm$
7040 print"The byte being edited is marked by '⊥'
7060 print"and this position can be moved by the
7070 print"regular cursol control characters."tm$
7080 print"Following commands are also operative:-"
7090 print"  @    -   change record #
7100 print"  &    -   go to next record, without
7110 print"           entering screen
7120 print" 'HOME'-   change byte range
7130 print" 'CLR' -   exit from Editor"tm$
7140 print"To enter record, hit shifted Carr.Ret.
7150 print"To change Filename, hit '@' and 'CR'"tm$
7160 print"Carriage return is displayed by the
7170 print"editor as 'ℕ', and Quotes as '@'
7180 print"■■■■■■■■Hit any key to continue!■";:
7190 poke158,0:wait158,1
7200 poke158,0:print"■":return
ready.
```

Continued from page 15

The values may be changed with the edit followed by RETURN. Use this instruction to set up the PC value before single stepping with '.I'.

SAVE TO TAPE

.S "PROGAM NAME",01,0800,0C80

Save to tape #1 from 0800 hex up to but not including 0C80 hex and call the program "PROGRAM NAME".

TRANSFER MEMORY

.T 1000 1100 5000

This transfers memory in the range 1000 hex to 1100 hex, and start storing it at address 5000 hex.

EXIT TO BASIC

.X

This returns to Basic ready mode. The stack value saved when entered will be restored. Care should be taken that this value is the same as when the monitor was entered. A 'CLR' in Basic will fix any stack problems.

MONITOR INSTRUCTIONS

G – Go run
L – Load from tape
M – Memory display
R – Register display
S – Save to tape
X – Exit to Basic

EXTRAMON INSTRUCTIONS

A – Simple assembler
D – Disassembler
F – Fill memory
H – Hunt memory
I – Single Instruction
P – Printing Disassembler
T – Transfer memory

Supermon will load itself into the top of memory, wherever that happens to be on your machine. You may then save the machine code for faster loading in the future. Be sure to note the SYS command which links Supermon to the Commodore monitor.

Dear Sir,

I am writing to send you a copy of the simple program for storing data in the floppy which I mentioned, in a talk given by Jim Butterfield recently.

It is very simple and merely writes a set of characters into the floppy then on continue reads them back out again. It reads back in even if the PET has been switched off and on in the meanwhile.

Yours sincerely Philip Deakin

FLOPPY MEMORY POKER

```
100 PRINT"■":OPEN15,8,15
105 FORI=33TO70
110 PRINT#15,"M-W"CHR$(I-33)CHR$(18)CHR$(1)CHR$(I)
120 NEXTI:CLOSE15
140 REM
150 STOP
160 REM
170 REM
180 OPEN15,8,15:FORI=0TO36:PRINT#15,"M-R"CHR$(I);CHR$(18)
190 GET#15,X$:PRINTX$;:NEXTI:CLOSE15
READY.
```

Basic Machine Code & Assembly Language - an introduction *Emily Berk*

The following article presents an introduction to machine and assembly language code - what they are and why they are used. Included is a program that will demonstrate the speed advantage of machine language programming, and also, the difficulty of programming in numeric code.

A computer is an intricate maze of on-off switches. As such, it can do only two things in response to commands – it can turn switches off and it can turn switches on. The only way the first computer programmers could command the earliest computers was, in fact, to turn some of those switches on and others off. These days, the programmer has a variety of languages in which to tell the computer what to do. But at some level, in order for the computer to understand what is desired, the programmer's language is still converted to impulses that turn switches on and off.

High Level vs. Low Level Languages

High level languages – languages such as FORTRAN, BASIC, and PASCAL – are programming languages that have been developed to mimic human language. They are, therefore, easy for humans to use. In order for the computer to understand such a high level language however, it must be translated into the 1's and 0's (binary representation of numbers) that the computer understands. This translation is accomplished through the use of a translating program, a compiler or interpreter. It is the BASIC interpreter that comes hardwired into your Commodore computer that allows you to program your computer in BASIC. Additional compilers and interpreters can be purchased and loaded into your computer allowing you to program your computer in the high level language of your choice. Using high level languages simplifies the programmer's task, but he pays a price for this convenience.

Because of this translation process, and the inefficiencies it induces, high-level language programs run more slowly and take up more space in memory than do programs written in the computer's own language – machine language.

The machine language of a particular computer is determined by its microprocessor. (Commodore computers use 6502 machine language because they are driven by a 6502 microprocessor.) Machine language is transmitted to the computer as ones and zeroes, which are then interpreted by it as high and low voltages on its wires. Assembly language is a way of expressing these same binary values using mnemonics (symbolic representations of an instruction). Although nearly every assembly language command corresponds one-to-one with a machine language instruction, the computer will not understand them until they too are converted to binary code. The program that translates assembly language programs into machine language is known as an assembler. Assemblers are much simpler and more efficient that the compilers and interpreters that translate high-level languages into machine language because of the one-to-one relationship between assembly language instructions and machine language instructions.

Machine & Assembly: What They Look Like

The machine language representation of the 6502 instruction to "load" information into the accumulator is:

10101001 (binary)

The machine language instruction to move the value 21 (decimal) to the accumulator is:

10101001 10101 (10101 is the binary notation for the 21 decimal)

An assembly language mnemonic equivalent to the BASIC command "load" or machine command "10101001" is:

LDA

Thus the assembly language instruction that will move the value 21 (decimal) to the accumulator might be written as:

LDA #21

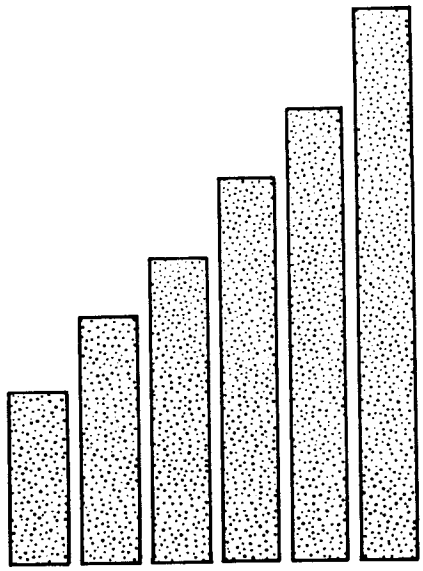
Not only is the mnemonic, LDA, easier for a programmer to remember than the number 10101001, but it also looks like what the command is meant to do.

A Programming Example

The following program will graphically illustrate the speed advantage of a program written in machine code. Imagine that the computer is controlling an experiment and periodically reads five results that can range between 5 and 35 from an instrument. It then logs them into memory. A simple horizontal bar chart is required so that the user can keep an eye on progress, but the computer must be "off-line" to the experiment for as short a time as possible.

Lines 10 through 80 contain the BASIC routine that reads the results and draws the bar chart. Lines 90-160 allow BASIC to store a machine language program (starting at location 826) that can then be executed by typing SYS826. Lines 140-160 contain the machine code equivalent of lines 10 through 80. The machine code used here is written in decimal because the BASIC interpreter will convert the decimal to binary before storing the numbers in memory.

```
10 S=32768
20 FOR X = 5 TO 0 STEP - 1
30 A = PEEK (870+X)
40 FOR Y=ITOA
50 POKE S+Y,102
60 NEXT Y
70 S=S+80
80 NEXT X
90 AD = 826
100 READ Q$
105 IF Q$ = "*"THEN STOP
110 POKE AD, VAL (Q$)
120 AD=AD+1
130 GOTO 100
140 DATA 169,0,141,76,3,169,128
142 DATA 141,77,3,160,5,190,102,3
144 DATA 169,102,157,200,128,202
150 DATA 208,250,136,48,17,173
152 DATA 76,3,24,105,80,141,76,3
154 DATA 144,231,238,77,3,76,70,3
160 DATA 96,10,15,20,25,30,35,*
READY.
```



Type RUN and see the BASIC program draw the bar chart. Now clear the screen, move the cursor down several places, and type SYS826. The speed advantage of machine code is quite apparent. You can also see that the machine language commands used to generate the chart are incomprehensible to a human being who does not have the key – a knowledge of machine language commands. This program is also difficult to debug for the same reason. There is nothing in the machine language part of the code like a Basic REM statement that tells the programmer what it was that he intended to do at any particular point. So, three months after he writes this program, when he decides to make a slight change, the programmer will have to rethink the entire algorithm to figure out what each command does and how it can be changed.

SYS & USR: A Compromise

The compromise many programmers decide on is to write as much of the code as they can in Basic, but to branch to machine language subroutines to perform time-critical portions of code. This type of compromise is facilitated by two Basic commands, USR and SYS.

The SYS command is analogous to the BASIC GOSUB command. Its format is: SYS (addr), where addr is an integer or integer expression between 0 and 65535 that specifies the location of a machine language subroutine the programmer has stored previously. Return is effected by a Return to Subroutine machine language command.

The USR command is analogous to a function call – with USR is passed an integer or integer-expression parameter, which can then be used by the machine language routine in its calculations. USR causes a jump to a subroutine whose address has been previously poked by the programmer into memory locations 1 and 2. The machine language program that runs because of the USR command may cause the value of the parameter that was passed to change. For more information about the SYS and USR commands, refer to the PET User Manual, or the PET Revealed, by Nick Hampshire.

Next issue, we will continue this series with more about machine language and assembly language programming.

Education

D. Milnes

TEC records on the PET.....

Over the last 4 years due to the introduction of Technician Education Council courses into the Further Education system a great deal of administrative work has been placed firmly on the shoulders of the teaching staff, one in particular – the TEC coordinator of the department.

The majority of the work involved is keeping track of students and units studied, exemptions etc. This involves a tremendous amount of time and paper work particularly at the beginning of the college year – sorting out new enrolments and their qualifications and the making up of class lists.

The program was devised by the author to minimise this work and to ensure that students were not 'lost' in the system.

After loading the programme and typing 'run' the operator is presented with a menu which gives the main options available within the programme:-

Option required

1) Inputting of student data
2) Deleting of students
3) Updating
4) Changing name
5) Printout or Display
6) Terminate
7) Printout of student names
8) Appending student number

We will look at the above 8 options in turn and see in detail the facilities available.

1) Inputting of student data

There are two options available under this heading –

a) data from tape
b) data from keyboard

Option (a) is for loading existing students into the PET in order to update, printout etc or before adding in new students under option (b).

New students are brought into the TEC records via the keyboard of the PET. Two types of student intake are catered for – students new to the college who have just left school and students who are entering TEC from existing college courses eg CGLI 203 craft.

First the students new to the college –

The first prompt is for the name of the student, then his AH4 score and maths score (these are tests which are taken by all new students to the department).

The next prompt is for any GCE's or CSE's and if yes then the essential subject grades are asked for:–

Maths
Physics
Technical Drawing
Workshop Theory & Practice
English
Number of other GCE's

In any of the above 5 subjects, if the GCE grade is A, B or C or the CSE grade is 1 then the student is automatically placed in the appropriate level 2 unit and given an exeption in the corresponding level 1 unit.

The other type of new student on TEC is the student who enters from one of the existing courses for example City and Guilds 203 – Craft.

This student is typed in under the dummy name "spz". The computer prompts for the students name and course of study previously taken, the exemptions and units studies are then typed in from the keyboard.

It might at this stage be of interest to look at how the information is stored in the computer memory.

A) NEW STUDENTS

The student name is stored in one element of a 120 element array and this would appear as follows:

Name: A;1043632mapbtawle2000 0000xx

The delimiter ";" of the name is searched for by the computer and when this is found we have the following:

3 digits after the delimiter – 104 is the AH4 score.

The next 2 digits give the maths score – 36.

The next two digits give the number of GCE 'O' levels and CSE's respectively in this case 3 and 2.

m – maths ..... a – grade
p – physics ..... b – grade
t – tech drg ..... a – grade
w – w/shop th. ... 1 – grade
e – english ..... 2 – grade

The digits '0000000' give the space for the student TEC number when registered and the final two 'xx' give the option when we get a floppy disc unit for a two figure reference for the students firm and training officer.

B) STUDENT from EXISTING COURSE

Name.B;*CGL12030000000xx

The '*' after the name delimiter identifies the student as one who has attended on a different course in previous years. This is reflected in any subsequent printout of information regarding this student. The rest of the name string is the same as that of the new enroller to the TEC course – namely student TEC number and firm identity.

2) Deleting of names

As the name suggests the facilityfor deleting a student name and record from the course.

3) Updating

The first prompt is for the student name to be updated. If this cannot be found because of incorrect spelling then an error message is generated which then prompts for the correct spelling.

On finding the required student a table is given showing the current unit status for the student and a prompt for the unit number and new status (see Figure 7). The table is updated on the PET screen at the same time so that the operator can see which units have been updated. When updating is finished 'n' is typed to the unit prompt and as a double check the amended unit status is given with the option of correcting if any mistakes have been made. When the operator is satisfied that everything is okay the prompt 'more students to update' is given.

If 'yes' the programme loops back to the name prompt and if 'no' the original menu of options is displayed.

The PET keeps track of the total number of units obtained by each student in previous years and if this total plus the units currently being studied is equal to the number of units required for a certificate at the end of the current year the name of the student is printed out. In this way a list of students is obtained who have to pay their final TEC registration fee.

Also at the end of the year any student going into HTC units is printed out and a separate data cassette file set up for these who are at the same time deleted from the TEC file.

A programme will be developed in the near future for the Higher Technician certificate.

It must be emphasised at this point that the programme is easy to follow – the PET prompts for all information and then processes the data into a readable form. Output can be to the screen or a printer so if a printer is not available the programme can still be used. If one inadvertently drops out of the programme say by pressing the 'run/stop' key by mistake typing 'goto 260' will bring the system back to the first option menu without losing data.

At the end of each option session a simple command takes the operator back to the original 8 options when the next option can be chosen.

4) Changing name

Occasionally a students name will be mispelt – option 4 changes the name.

5) Display or printout

Information can be directed either to the screen or printer with a change your mind option if you have chosen the printer and then realise that you hit the wrong key.

When the printout device has been opened there are three choices of output

a) Single student
b) All students
c) Subject lists

On the choice of either (a) or (b) we can look at one of the following options:-

i) Units currently attended
ii) Units exempt
iii) Units passed with the state of pass
iv) Units referred
v) Summary of above

If the subject lists option is chosen then it is obvious that a printout is obtained of all students currently attending the particular subject hosen in the TEC programme.

6) To terminate

When all changes have been carried out on the data file this option will record the amended data on cassette for future reference.

7) Printout of students names

Gives a printout of all students currently enrolled on the TEC course.

8) Students numbers

Students who enroll on TEC courses for the first time do not have a TEC number. Thus the dummy number '0000000' is placed in the name string ready for the official number to be typed in under this option.

Every student name who has this dummy 0000000 is printed out by the computer and the prompt 'Student number' is given. On typing the official number the '0000000' is replaced by this number and the next student name given by the computer.

This continues until all the newly registered students have been given their TEC numbers.

Conclusion

To conclude a programme is at present being developed at the Holly Bank annexe of Huddersfield Polytechnic by Mr Peter Lazenby which will keep records of students marks in homework, phase tests etc and issue pass lists of students at the end of the year.

It is hoped to use that programme in conjunction with the one outlined in this article. This will completely computerise all TEC student records from the start on TEC to finish of the Higher Technician Certificate.

D. Milnes
TEC co-ordinator
Eng: Dept:
DABTAC
Dewsbury
West Yorks

Where next?

Where to Start

PET - where do I start

There have been a number of articles in this and other newsletters aimed at the beginner to the PET scene - programming hints, beginners guide to disk programming and so on. What there hasn't been is an article for someone who's thinking of buying, or perhaps already has, would like to expand but isn't sure in which way to go. Do you upgrade ROMS, which program do you aim at next, the sort of questions that must get asked many times over.

To quote Jim Butterfield "I would strongly recommend upgrading". The logic behind this is clear - as the newer and newer equipment comes out, so more and more powerful programs appear, superceeding what has gone before. So, are there any problems involved in upgrading ?

The following ROM family tree is for those with the original 8k PETs, and is a guide to what can be done, and what to order from your local dealer.

No Problems

What about those of you with the BASIC 2.0 (sometimes referred to as BASIC 3.0) ROM sets? Here there are no problems involved, as it is a straightforward upgrade to BASIC 4.0. (If you're beginning to get confused over all these ROM numbers, refer to Jim Butterfield's article on What Software System ?). With the final upgrade to BASIC 4.0 we can, as yet, go no further, although I would imagine that before too long some enterprising company comes up with something - don't hold your breath. The 80 column machine thus becomes out of reach to present 40 column machine users, although the possibility exists of exchanging such equipment via the "second-hand" column elsewhere in the newsletter. But there is no reason that one shouldn't buy an 80 column PET to begin with!

If you started off using cassette decks, all Commodore computers (with the one exception of the micro-mainframe) can use cassette decks, so that you would have no problem in transferring programs from cassette to disk if you go for the use of disk drives. As the cassette market has few business programs for it, it is really essential that the serious businessman equips himself with a disk drive of some kind.

So, what are the upgrade possibilities here? The original disk drive, the 2040, and its successor the 3040, can both be upgraded to the current model, the 4040, although it is somewhat difficult to achieve this on the original model. At the 4040 we come to a halt, the 8050 that goes with the 80 column PETS is beyond our reach.

Again, for the benefit of those of you getting totally confused, we have the family tree :-

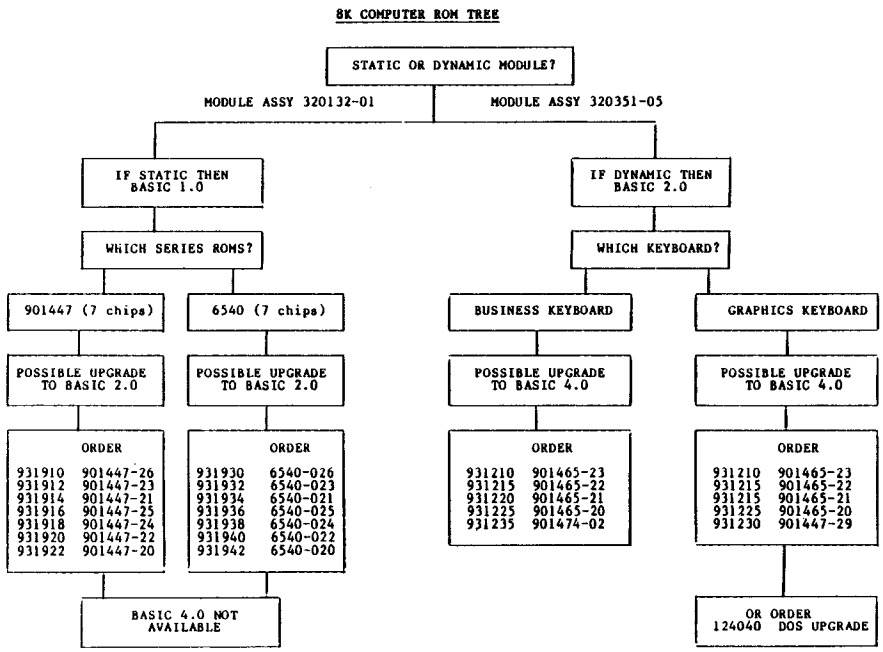
PET	BASIC	DISK DRIVE	DOS
2001/8	1.0	-	-
3000 series	2.0	2040/3040	1.0/1.2
4000 series	4.0	4040	2.1
8032	4.0	8050	2.5

Now we come to the question of "what works with what ?" Apart from the 2001/8, anything will work with anything, but you will find certain restraints - for instance, all of the disk commands that were an 'add-on' when we go to BASIC 4.0, will simply produce a syntax error when you attempt to use them on BASIC 2.0 machines, and these commands are very useful (as we'll see later). Similarly, some of the commands that the 4040 and 8050 understand cannot be used with the 2040/3040.

Where do you start?

Now we know what levels you can start at and what levels you can go to. The question remains - where do you start ? That obviously depends on what you want to do - stock control in a 14,000 square feet factory would be rather difficult with a 2001/8, but becomes somewhat easier with an 8032 and 8050. This is something we'll go into in more detail next time, along with a guide to some of the programs that are around, but for now I'll leave you with an article on the pros and cons of upgrading your system, followed by a guide to using relative records on your disk unit -only possible if you do upgrade. You may at first find this difficult to follow, but persevere - consult David Pocock's disk drive series, past, present and future.

Reproduced from the Australian Commodore Magazine.



BASIC 4.0 - DOS 2.0 AND THE RELATIVE RECORD SYSTEM

NOTE: Throughout article // =#

Commodore is now distributing computers and disks with upgraded operating systems. These are of course, BASIC 4.0 and DOS 2.0 But many users that have BASIC 2.0 and DOS 1.0 are asking themselves, "Should I upgrade?".

The new operating systems offer many advantages over the old, but there are cases where upgrading may hurt more than help. This would refere to those who 1) have a working system performing with mishap, and 2) don't do any programming of their own. More specifically this would be businesses that have aquired equipment and a custom program(s) to perform special tasks. There are subtle differences in the new systems that may cause discrepencies once upgraded. However, this does not rule out the possibility of upgrading. Higher capacity may be necessary to maintain your system efficiently. This would mean a "forced" upgrade to the 8050 disk, which contains the new DOS, and program modification may be required.

Serious programmers, on the other hand, should consider upgrading as seriously as their programs. Some new features are:

BASIC 4.0

1. garbage collection time has been reduced to negligible.
2. Shifted RUN/STOP loads and runs first **disk** file.
3. Disk error channel read automatically into DS and DS$, same as TI and TI$ read the clock. These new variables are one reason programs may require modifications.
4. PRINT// command omits line feed after carriage return on files OPENed with a logical file number less than 128; 128 or greater still sends CR/LF.
5. Disk commands now included in the BASIC. Although BASIC 2.0 could handle the disk, PRINT//ing to the command channel was somewhat clumsy.

Direct access disk commands do not change in BASIC 4.0 (i.e. format is still PRINT//15, "ul",b-a, b-p, etc) but do change in DOS 2.0 (see DOS 2.0 below). Also not that the INITIALIZE command does not get keyword priviledges in BASIC 4.0 BASIC 4.0 was designed to work best with DOS 2.0 which does automatic initializes.

Basic 4.0 also has other commands that work only with DOS 2.0:

```
APPEND//2, "file", dl
CONCAT"more than data", dO TO"existing data",dl
RECORD//2, 3000, 5
```

The APPEND// command OPENs an existing file for writing. DOS 2 positions to the end of the file such that data can be "appended".

The CONCAT command concatenates one file "TO" another existing file (SEQ type files only). Concatenating was possible with the DOS 1.0 'C' copy command, but an extra sequence of scratch and rename commands would be necessary to accomplish the above:

```
DOS 2 CONCAT "more data", dO TO "existing data"dl

DOS 1 PRINT//15,"C1:temporary=1:existing data, 0:more data"
      PRINT//15,"S1:existing data"
      PRINT//15,"R1:existing data=1:temporary"
      PRINT//15,"SO:temporary"
```

Thanks to DOS 2.0, a single BASIC 4.0 command does it all! But remember, DOS 2.0 does the work; BASIC 4 only sends the command string to the disk command channel.

RECORD// works the DOS 2 Relative Record System. This feature of the new DOS makes it virtually indispensable!

Although the above three commands belong to BASIC 4.0, they can be simulated with BASIC 2.0, however, DOS 2.0 must be in the disk for them to work. (See article on DOS 2.0 command from BASIC 4.0)

DOS 2.0

1. Automatic initializing.
2. "@" SAVE with replace fixed.
3. Formatting and Duplicating approximately 5 times faster.
4. Directory track and 6 other tracks have 1 less sector for 144 directory entries max and 664 blocks free max. It was felt that the recording density for DOS 1.0 diskette middle tracks was too high for reliability. DOS 1.0 diskettes will require converting to work on DOS 2.0 (see COPY command below). Although both diskette types can be read on either DOS, writing DOS 2 diskettes with DOS 1 is fatal. DOS 2 doesn't allow writing to DOS 1 disks.
5. RENAME command fixed.
6. COPY command now allows default characters. (e.g. COPY i*", dO to "*",dl would copy all files starting with "if" on dO to the same name on dl. Also COPY dO TO dl copies all files over... good for converting DOS 1.0 diskettes to DOS 2.0 diskettes).
7. "B-W" direct access commands removed; use "U2" instead. All others remain the same.
8. Sector byte zero now accessible from B-P command.

See Figure 1 (below)

BASIC 2.0	BASIC 4.0	
LOAD"prog",8	DLOAD"prog"	
SAVE"1:prog",8	DSAVE"prog",dl	;defaults to DO
VERIFY"1:prog",8	VERIFY"1:prog",8	;no change
OPEN 2,8,6,"1:file,s,w"	DOPEN	;defaults unit 8, omit w for read
CLOSE 2	DCLOSE#2,dl ON u8	no change for USR file ;omit "#2" and "dl" and close all files ON u8
LOAD"$1",8:LIST	DIRECTORY dl or CATALOG dl	
PRINT#15,"N1:title,xx"	HEADER"title",dl,ixx	
PRINT#15,"S1:prog"	SCRATCH"prog",dl	
PRINT#15,"V1"	COLLECT dl	
PRINT#15,"D1=0"	BACKUP dO TO dl	
PRINT#15,"R1:file=1:prog"	RENAME "prog",dl TO "file",dl	
PRINT#15,"C1:prog=0:prog"	COPY "prog",dO TO "prog",dl	

Figure 1.

9. Error channel cleared on receiving correct command syntax. DOS 1 left the error light on until completion of a successful command (excluding LOAD"$O",8).

THE RELATIVE RECORD FILE SYSTEM

Built in to the new DOS 2.0 is a filing system known as The Relative Record System. It's called Relative Record because each record is relative to another.

When a relative file (shown as REL on directory) is created, each record will have the same byte length. The length of the records are chosen by the user and can be any length between 1 and 254. No bytes are wasted which means, in most cases, records will span sector boundaries.

Essentially, a REL file is like an SEQ file with entry points. These entry points are stored in "side sectors" which take up space on the disk, but are transparent to the user.

Each side sector can handle up to 30K with a maximum of 6 side sectors. This limits REL files to 180K, but since 4040 diskettes are 170K, a REL file could use up the whole disk. The 180K limit also applies to the 8050.

The speed of the system is incredible; maximum 3 block reads to access any record, regardless of file size.

A maximum of three REL files can be open on the disk simultaneously provided no other files are open.

The command set associated with REL files is:

```
DOPEN//
RECORD//
INPUT//
GET//
DCLOSE//
```

REL files can be COPYd, SCRATCHed, RENAMEd, etc., just like any other file. Treat them no differently than any other file, but with the same amount of respect. REL files must be DOPENd and DCLOSEd properly, using ST and DS/DS$ for file status interrogation.

Example Set-Up

First you must decide how many bytes maximum your information will need. This will be the number of bytes maximum per field plus one byte for a carriage return at the end of each field. You could save on bytes by not using carriage returns but then you must know how to split up the record into fields using MID$ upon retrieval. Once again, no more than 80 characters without a carriage return.

Once you've chosen a length or Record Size, put it in a variable, say RS. Choose a logical file number, a filename and a drive and:

```
DOPEN//6, "FILENAME",DO,L(RS)
```

You can write or read a REL file once opened. When DOPENing for the first time, the record size (RS) must be specified. After that the length need not be given. If it is, it must be the same as before else a disk error will occur and the disk will abort the open attempt.

On creating the file, the disk procedes to build records in disk RAM. These will be empty until you fill them with data. An empty record starts with CHR$(225) followed by RS-1 CHR$(0)'s. (see note 1 at the end)

You are now ready to store data. The DOPEN automatically positions to record number 1. After a PRINT//, the DOS will position to record 2. This means that placing multiple strings into a single record must be done using one PRINT// statement, else the strings will go into successive record numbers.

Assuming R$=CHR$(13)...

```
DO
1 0 0
PRINT//6,"HELLO"R$;A$;R$;B$;R$;X%,R$;
DON'T!
100 PRINT//6,"HELLO"R$;
110PRINT//6,A$;R$;
120 PRINT//6,B$;R$;
130 PRINT//6,X%;R$;
```

This would put "HELLO" in record //1, A$ in record 2, B$ in records 3 and X% in record //4.

This could be a drawback, especially if your variables are in an array and you wish to use a loop to output all to the same record //. This brings us to the RECORD// command.

```
RECORD//LF,(RR),(PN)
```

RECORD// tells the file (LF) to position to record number RR at byte position PN within the record. The variable PN can be from 1 to 254. Variables in the RECORD// command must be enclosed in brackets. Output using a loop might look like:

```
100 PN=1
110 FRO J=1 TO NF (NF=number of fields)
120 RECORD//6, (RR),(PN)
130 PRINT//6, FL$(J);R$;
140 PN=PN+LEN(FL$(J))+1 (+1 for C/R)
150 NEXT
```

The ";RS;" in line 130 could be left off since this would be handled by BASIC.

Another method would be to concatenate the fields into one string and output:

```
100 FL$=""
110 FOR J=1 TO NF
110 FL$ = FL$+FL$(J)+RS
120 NEXT
130 PRINT//6,FL$
```

Remember....strings in memory can be length 255 max. Max REL record length is 254. If you print a string to a REL record that is longer than the record lengh, an OVERFLOW IN RECORD error will occur in the error channel. BUT, the first RS characters of the string will make it into the record; the rest will be lost. Should this happen, there probably won't be a carriage return at the end of the record. That doesn't matter. You will still be able to retrieve this data. As a matter of fact, carriage returns are not necessary at the end of a record, even if the data doesn't fill the record! "But why?", you ask....

REL Record Retrieval

As mentioned earlier, an empty record starts with CH$(255) followed by RS-1 CHR$(0)'s. This is done by the DOS.

Let's say our record size is 50. If we take the characters H, E, L, L, and O, and send them into REL REC //1 starting at position 1 without a carriage return, (i.e. PRINT//6,"HELLO";) the DOS would do as it's told and put "HELLO" into REL REC //1 with no carriage return. Not too surprising, eh. However, once that's done, the DOS procedes to "pad" the remainder of the record with CHR$(0)'s; in this case 45 of them. The DOS is now positioned at REL REC //2.

Now let's say we position back to REL REC //1 with a Record//6, 1 command.

The INPUT// command stops on carriage return or EOI. ST is set to 64 on EOI, otherwise ST=O. (see note 2 for details).

If we now execute an INPUT//, the DOS sends the H, E, L, L, and O. But when the DOS sees the CHR$(O) it also sends EOI which is just as

good as carriage return. ST is set to 64 and the DOS positions automatically to the next record; REL REC //2.

The DOS would also send EOI if the character being sent was from the last position in the record. In this case the record is not full, but this means that the character in the last position doesn't have to be a CHR$(13). You can save 1 byte per record this way. For 2500 records that's almost 10 fullblocks!

Back to our example, INPUT// terminated when the DOS saw CHR$(O) and sent EOI. This has further ramifications. Suppose you were to execute something like:

```
100 RECORD//6, 1, 1
110 PRINT//6, "HELLO";  ;or HELLO";R$;
120 RECORD//6, 1, 20
130 PRINT//6,"JIM";
```

there would be CHR$(O)'s left in between "HELLO" and "JIM" would be lost forever to INPUT//, unless you position back to it using RECORD// before INPUT//ing. Otherwise, only GET// could get it back. The DOS does not send EOI with CHR$(O) when using GET//.

Therefore, if you're anticipating blanks between data, or blank fields representing no data, it's best to construct the record in RAM first using spaces field padding. Remember though, leading spaces will PRINT// to the disk but INPUT// (as with INPUT) ignores them. Leading spaces include spaces at the beginning of a record and spaces immediately following a carriage return within the record.

PRINTOVER

Recall that the PRINT// command sends the characters into the record and then pads to the end of the record with CHR$(O)'s. This can be hazardous, especially if valid data exists beyond the data being sent into the record. This data would be wiped out with zeros. One more reason why you should construct the record in Ram first. You could get around this by putting the new data into the disk buffer with a "Memory-Write" routine, but that's fairly advanced and we won't cover that here.

END OF FILE DETECTION

The following routine could be used to read the entire contents of a REL file:

```
10 DOPEN//8,"FILE NAME"
20 INPUT//8, A$
30 PRINT A$
40 IF DS=50 THEN DCLOSE//8 : END
50 GOTO 20
```

On DOPENing, the file positions to record 1 and automatically positions to successive records after INPUT//ing each records' valid data. This would continue until reaching a record that hadn't yet been formated. DS DS$ would read 50, RECORD NOT PRESENT. But the last record used isn't necessarily the last record formatted (see note 1.) Storing the number of the last record used would take care of that. Give it a SEQ file of it's own and update it every time it changes using "@" write replace.

Empty files start with CHR$(255). This gets done by the DOS initially, but if a record DELETE is done, this "empty" flag should be replaced (i.e. PRINT//If, CHR$(255)). This available file space can then be detected for future use.

ONE MINOR GOTCHA

When a REL file is DOPENed for the first time, only one sector is allocated for data. If the file is aborted (i.e. no DCLOSE, DIRECTORY display, reset, etc.) before the DOS allocates a second data sector, the side sector information doesn't get written to the disk. That second data sector allocation forces the side sector onto the disk, but DCLOSING properly will always prevent this.

To be absolutely sure, although probably unnecessary, the following routine could be used:

```
50000 DOPEN//If"FILE NAME", DO,L (RS)
50010 RECORD// If,(INT(254/RS)+1)
50020 PRINT//If, CHR$(255);
50030 DCLOSE//IF
50040 RETURN
```

The fix actually defeats its own purpose as the file is properly DCLOSED in line 50030!

This would only have to be done once and your file is ready for I/O. Once again, the record size (RS) need only be given the first DOPEN.

NOTE 1

When a REL file is created, the DOS goes looking for some RAM to use inside the disk unit: a 256 byte buffer. The first two bytes are used to store the track and sector numbers of the **next** sector in the REL file just like SEQ files. The remaining 254 bytes are for record space, hence the 254 byte maximum record size. At this point the DOS fills the record space with CHR$(O)'s and puts a CHR$(255) "marker" in the first byte of each record. This byte would be a multiple of the record size. If the record sizer were 50, there would be CHR$(255) at bytes, 2, 52, 102, 152, 202, and 252 (offset by 2 due to track & sector bytes at 0 and 1).

If REL REC //1 were currently being written to or read from, you could procede to read or write REL RECs 2, 3, 4, and 5 without any mechanical disk activity. Requesting record //6 (i.e. RECORD//If,6,1) would return an error //50, RECORD NOT PRESENT because disk space for a 6th record hasn't yet been formatted. But 5 records don't fill the buffer completely; there are still 4 bytes left (252-255). These belong to record //6. The next PRINT// would start putting characters into these 4 bytes, at which point the DOS would find another available sector, stick it's coordinates into bytes 0 and 1, and write the buffer contents onto the diskette. Now the buffer is reformatted with the first 46 bytes of the record space belonging to record//6. A DCLOSE would write the rest of the data to disk. Requesting record //3000 would force the DOS to format all records in between before allowing access to the record.

NOTE 2

1. INPUT// continues to input characters from the disk until it sees a carriage return (,comma or a colon but we'll ignore these here). The next line of your program should be a check of ST. If there is more data, ST will be 0; if not, ST will be 64.

2. INPUT// also terminates on the receiving EOI (End or Identify). EOI has a line of it's own on the IEEE bus. INPUT// checks this line. If it turns on, then no matter what character INPUT// has just received, inputting stops **and** ST is set to 64.

That all sounds like a lot but it really isn't. The Relative Record System is really quite easy to work. Being new, it'll take some getting used to. Once you're storing data in REL RECS, you'll hate to think how you did it any other way!

Disk Use for Beginners

David J. Pocock

Hello again! Welcome to Part 4. This month I will be covering the use of the DOS support program (also called UNIVERSAL WEDGE) and for users of BASIC 4.0 machines a comparison list of disk commands. Also I will be starting on SEQuential files.

Unfortunately I must start this article with an apology (again). In my second article I said that if you want to upgrade a 2040 to DOS 2.1 (4040) it involved changing the 'Main Board'. This however is not the case and was due to a misunderstanding on my part of some information given to me (just goes to show that we are human!). It is possible to upgrade a 2040 to 4040 by removing all the ROMs (which are in sockets) and replacing them with a new set (which contains 1 or more there is a spare socket). Another chip has to be added on the board, which involves soldering and altering some links on the board (cutting printed circuit board tracks and soldering wires across others) to make the chip work– I suggest you see your local dealer!

DOS SUPPORT

The DOS SUPPORT program supplied on the utility disk (packed with the disk unit) enables disk commands to be sent without having to OPEN a file and without the use of PRINT #. It also enables you to read the 'ERROR' message without a BASIC program and to look at the DIRECTORY without destroying any BASIC program in memory.

The DOS SUPPORT commands are preceded by the "greater than" symbol '>' or the "at" symbol '@' depending on the keyboard in use (both work but '@' is easier to get to than '>' on an 8032 keyboard).

eg @10 Initialise the disk in drive 0.
>V1 Validate the disk in drive 1.

To read the ERROR message just type the '>' or '@' and press RETURN.

To examine the DIRECTORY type:-

>$0	For drive 0
>$1	For drive 1
>$	For both drives
>$0:A*	Any files on drive 0 beginning with the letter A.

N.B. The '>' or '@' must be in the first column (far left) of the screen, otherwise a SYNTAX ERROR will be generated. Also the DOS SUPPORT facilities do not function from within a BASIC program and you must use the full method (see: there WAS a reason for all that hard work).

BASIC 4.0

For those of you with BASIC 4.0 (8032 or 4032), and those thinking about buying or upgrading, here is a list of the disk commands, their DOS SUPPORT abbreviations and the BASIC 4.0 commands.

The following will be used throughout for examples:-

id	Disk Identity
dr	Drive number (0 or 1)
sdr	Source Drive (copy duplicate)
ddr	Destination Drive
name	Name of a disk file
onme	Old Name (copy)
nnme	New Name

Lowercase characters are filled in by you, everything else is as it appears.

New a blank disk		
DOS SUPPORT command	N	Ndr:name,id
BASIC 4.0 command	* HEADER	HEADER"name",Iid,Ddr
Duplicate a Disk		
DOS SUPPORT command	D	Dddr=sdr
BASIC 4.0 command	BACKUP	BACKUP Dsdr TO Dddr
Copy a file		
DOS SUPPORT command	C	Cddr:nnme=sdr:onme
BASIC 4.0 command	COPY	COPY "onme",Dsdr TO "nnme",Dddr
Validate a disk		
DOS SUPPORT command	V	Vdr
BASIC 4.0 command	COLLECT	COLLECT Ddr
Rename a file		
DOS SUPPORT command	R	Rdr:nnme=onme
BASIC 4.0 command	RENAME	RENAME Ddr,"onme" TO "nnme"
Scratch a File		
DOS SUPPORT command	S	Sdr:name
BASIC 4.0 command	* SCRATCH	SCRATCH "name",Ddr
Initialise a disk		
DOS SUPPORT command	I	Idr
BASIC 4.0 command	Not implemented as 4040 and 8050 perform automatic initialisation (provided id of disk is different - readers comments ?)	
Directory Listings		
DOS SUPPORT command	$	@$dr
BASIC 4.0 command	DIRECTORY	DIRECTORY Ddr
	CATALOG	CATALOG Ddr
Reading the 'error' channel		
DOS SUPPORT command		@
BASIC 4.0 command	DS , DS$	PRINT DS or PRINT DS$
DS gives the error number		
DS$ gives the complete message		

* This indicates that BASIC 4.0 asks 'ARE YOU SURE' before it performs the operation. The BASIC 4.0 commands can be used from within a BASIC program ; however certain rules have to be obeyed (these are in the BASIC manual) and I will give them in a later article.

(Editors Note : We had to cut the Sequential file bit to leave room for the rest of the newsletter - see you next time).

Disk Drive Programming

Four more programs for those of you getting rapidly into disk drives, all by the inimitable Jim Butterfield. They will all, with the exception of reading relative files, run on any PET and disk drive – that particular program requires BASIC 4.0. Beginners by now will be getting a fair grasp of what is going on from David Pocock's articles – experts will hopefully find these routines useful anyway, either as incorporated into your own programs, or used on their own. I'll try and keep this series of programs (preferably going onto other peripherals as well in due course – for those of you with 2001s and/or cassette decks I'll attempt to do a similar thing for you next month) continuing as the months go by.

If there's any routine you've written, for whatever device, that you think might save others an amount of head scratching, I'd be delighted to reproduce it here. It you can send me a clear listing that will be a great help.

Anyway, program on .....

COPY ALL - THE ULTIMATE COPYING PROGRAM

```
100 PRINT"[CLR] DISK.COPY.ALL      JIM BUTTERFIELD"
110 DIM L2(232),L1%(232),N$(232),T%(232),T$(4)
120 DATA XXX,SEQ,PRG,USR,REL
130 FORJ=0TO4:READT$(J):NEXTJ
140 INPUT"FROM UNIT  8[3CL]";F
150 GOSUB800
160 F$=D$
170 INPUT"TO UNIT  9[3CL]";T
180 GOSUB800
190 T$=D$
200 IFF=T ANDF$=T$THENRUN
210 GOSUB860
220 CLOSE1:CLOSE15:OPEN 15,F,15:PRINT#15,"I"+F$
230 GOSUB830:IF E THEN STOP:GOTO220
240 INPUT"PATTERN  *[3CL]";P$
250 FORJ=1TOLEN(P$):IFMID$(P$,J,1)<>"*"THENNEXTJ
260 P1=J-1:IFP1>0THENP$=LEFT$(P$,P1)
270 PRINT"HOLD DOWN 'Y' OR 'N' KEY TO SELECT"
280 PRINT"PROGRAMS TO BE COPIED..."
290 OPEN 1,F,3,"$"+F$
300 GOSUB830:IFETHENSTOP:GOTO220
310 GET#1,A$:A=ASC(A$+" ")
320 IFA=1ORA=65THENL1=253:GOTO350
330 IFA=67THENL1=761:GOTO350
340 CLOSE1:PRINT"?????":STOP
350 FORJ=1TOL1:GET#1,X$:NEXTJ
360 N=0:N1=0:R=255:Z=89
370 N$="":GOSUB1020:T9=Y-128:GOSUB1000
380 J1=1:Z$=CHR$(160):FORJ=1TO16:GET#1,X$:N$=N$+X$
390 IFX$<>Z$THENJ1=J
400 NEXTJ
410 GOSUB990:L1%=Y
420 GOSUB980:GOSUB980
430 L2=X+256*Y
440 IFT9<1ORT9>4GOTO540
450 IFP1>0THENIFP$<>LEFT$(N$,P1)GOTO540
460 PRINTN$;" ";T$(T9)
470 P=PEEK(151)ANDR
480 GETZ$:IFZ$=""ANDP<255GOTO520
490 IFZ$="Y"ORZ$="N"THENZ=ASC(Z$):R=255:GOTO520
500 IFZ$=CHR$(13)THENR=0:GOTO520
510 GOTO480
520 IFZ<80THENPRINT"[CU]                              [CU]":GOTO540
530 N=N+1:L2(N)=L2:N$(N)=LEFT$(N$,J1):T%(N)=T9:L1%(N)=L1%
540 IFST>0GOTO570
550 N1=N1+1:IFN1=8THENN1=0:GOTO370
560 GOSUB1000:GOTO370
570 CLOSE1:CLOSE15:PRINT" * * * * *"
580 FORJ=1TON
```

Continued

```
 590 L2=L2(J):T%=T%(J):IFL>L2GOTO640
 600 PRINT"*** OUTPUT DISK FULL"
 610 INPUT"DO YOU HAVE A NEW ONE";Z$
 620 IFASC(Z$)<>89THENEND
 630 GOSUB860:GOTO590
 640 OPEN14,F,15:OPEN15,T,15
 650 PRINTN$(J);LEFT$("                    ",17-LEN(N$(J)));
 660 OPEN3,F,3,F$+":"+N$(J)+","+T$(T%)
 670 INPUT#14,E,E$,E1,E2:GOSUB840:IFETHENPRINT"** ";E$;E:GOTO750
 680 IFT%=4THENOPEN4,T,4,T$+":"+N$(J)+",L,"+CHR$(L1%(J)):GOTO700
 690 OPEN4,T,4,T$+":"+N$(J)+","+T$(T%)+",W"
 700 L=L-L2:GOSUB830:IFETHENPRINT"*** ";E$;E:GOTO750
 710 IFT%=4THENSYS3312:GOTO730
 720 SYS3272
 730 N$(J)="":GOSUB830:IFETHENPRINT"**** ";E$;E:GOTO750
 740 PRINT"[CU]"
 750 CLOSE4:CLOSE3:CLOSE15:CLOSE14
 760 NEXTJ
 770 X=FRE(0):INPUT"ANOTHER INPUT DISK READY";Z$
 780 IFASC(Z$)=89GOTO220
 790 END
 800 INPUT"DRIVE  0[3CL]";D
 810 IFD*D<>DGOTO800
 820 D$=CHR$(D+48):RETURN
 830 INPUT#15,E,E$,E1,E2
 840 IFE=0THENE=(ST AND 191):E$="*ST*"
 850 RETURN
 860 OPEN15,T,15:INPUT"WANT TO NEW THE OUTPUT DISK  N[3CL]";Z$
 870 IFASC(Z$)<>89GOTO910
 880 INPUT"DISK NAME,ID";X$,Y$
 890 PRINT#15,"N"+T$+":"+X$+","+Y$
 900 GOSUB830:IFETHENSTOP:GOTO860
 910 PRINT#15,"I"+T$:OPEN1,T,0,"$"+T$+":!#$%&"
 920 GOSUB830:IFETHENSTOP:GOTO860
 930 GOSUB980:GOSUB980
 940 Q=Q+1:GET#1,X$:IFX$<>""GOTO940
 950 GOSUB980
 960 L=X+Y*256:PRINT"(";L;"BLOCKS FREE )"
 970 CLOSE1:CLOSE15:RETURN
 980 GET#1,X$
 990 GET#1,X$
1000 GET#1,X$
1010 X=LEN(X$):IFXTHENX=ASC(X$)
1020 GET#1,X$:Y=LEN(X$):IFYTHENY=ASC(X$)
1030 RETURN
READY.
```

READING RELATIVE FILES - JIM BUTTERFIELD

```
 100 INPUT "REL FILE NAME";F$
 110 DOPEN#1,(F$)
 120 INPUT "RECORD #";R
 130 IF R=0 GOTO 200
 140 RECORD#1,(R)
 150 IF DS>0 THEN PRINT DS$:GOTO 120
 160 INPUT#1,R$
 170 PRINTR$
 180 IF ST=0 GOTO 160
 190 GOTO 120
 200 DCLOSE#1
READY.
```

DISK ID CORRECTOR - JUM BUTTERFIELD

```
 1 REM DISK ID CORRECTOR
 7 DV=1:REM DR #
 10 OPEN9,0,0:OPEN15,8,15
 20 PK=PEEK(59468):POKE59468,12
 30 MD$="[HOME]":FORI=1TO20:MD$=MD$+"[CD]":NEXT
 40 FORI=1TO39:BL$=BL$+" ":NEXT
 50 P0$="[CLR,2CD]          ID CHECKER/CORRECTOR
 60 RE$="PRESS [RVS]RETURN[OFF] TO CONTINUE
 99 GOTO1000
 100 INPUT#15,ER:IFER=0THENRETURN
 110 INPUT#15,ER,EM$,ET$,ES$
 120 PRINTMD$"[RVS]DISK ERROR! [OFF]#"ER" "EM$" "ET$", "ES$
 130 END
 200 INPUT#9,Q$:PRINT:Q1$=LEFT$(Q$,1):RETURN
 300 CLOSE15:POKE59468,PK:PRINT"[CLR]":END
 1000 PRINTP0$:PRINT"[CD]PLACE DISKETTE TO BE CHECKED IN DRIVE"DV"[2CD]
 1010 PRINTRE$:GOSUB200
 1020 AD$="":ID$=""
 1030 PRINT#15,"I"+STR$(DV):GOSUB100
 1040 OPEN2,8,2,"#0":GOSUB100
 1050 PRINT#15,"U1:2";DV;",18,0":GOSUB100
 1060 FORJ=33TO34
 1070 PRINT#15,"M-R";CHR$(J);CHR$(16):GET#15,Z$
 1080 AD$=AD$+Z$:NEXTJ
 1090 PRINT#15,"B-P:2,162":GET#2,A$,A1$:ID$=A$+A1$
 1100 PRINT"[2CD]ACTUAL ID RECORDED ON SECTORS IS: "AD$
 1110 PRINT"[CD]FILE ID IS: "ID$
 1120 IFAD$<>ID$THEN1200
 1130 PRINT"[CD,8CR,RVS]THIS DISK IS OK!
 1140 CLOSE2
 1150 PRINTMD$BL$:PRINTBL$MD$"DO YOU WISH TO CHECK
 1160 PRINT"ANOTHER DISKETTE? (Y/N) [RVS]";:GOSUB200
 1170 IFQ1$="Y"THEN1000
 1180 IFQ1$<>"N"THEN1150
 1190 GOTO300
 1200 PRINT"[2CD]ACTUAL ID AND FILE ID DO NOT MATCH!
 1210 PRINTMD$"SHALL I PROCEED TO CORRECT? (Y/N) [RVS]";:GOSUB200
 1220 IFQ1$<>"Y"THEN1140
 1230 PRINTMD$BL$MD$"BE PATIENT...
 1240 PRINT#15,"B-P:2,162":GOSUB100
 1250 PRINT#2,AD$;:GOSUB100
 1260 PRINT#15,"U2:2";DV;",18,0":GOSUB100
 1270 PRINT#15,"I"+STR$(DV):CLOSE2
 1280 PRINTMD$BL$"[CLR]"MD$"CHECKING DISK
 1290 PRINTLEFT$(MD$,8);
 1300 GOTO1020
READY.
```

DISK MEMORY DISPLAY - JIM BUTTERFIELD

```
 100 PRINT"[CLR]DISK MEMORY DISPLAY
 105 PRINT"BY JIM BUTTERFIELD
 110 DATA77,45,87,0,18,16,162,0,189
 120 DATA157,64,6,232,224,16,208,245
 125 DATA76,193,254
 130 FORJ=1TO9:READX:C$=C$+CHR$(X):NEXT
 140 FORJ=1TO11:READX:D$=D$+CHR$(X):NEXT
 150 PRINT"[CD]THERE ARE TWO PROCESSORS:
 160 PRINT"  1) THE IEEE PROCESSOR
 170 PRINT"  2) THE DISK PROCESSOR
 180 INPUT"WHICH DO YOU WANT TO PEEK (1/2)    [3CL]";D
 190 OPEN1,8,15:P$=CHR$(4)+CHR$(16):R$=CHR$(224)
 200 PRINT"INPUT MEMORY ADDRESS":PRINT"IN HEXADECIMAL:
 210 PRINT"  [4CL]                                [CU]"
 220 INPUTZ$
 240 PRINT"[CU]";:IFLEN(Z$)<>4THEN220
 250 FORJ=1TO4:Y=ASC(MID$(Z$,J))
 260 IFY<58THENY=Y-48
 270 IFY>64THENY=Y-55
 280 IFY<0ORY>16THEN220
 290 Y(J)=Y:NEXT:K=0:PRINT"[6CR]";
 300 U=Y(3)*16+Y(4):V=Y(1)*16+Y(2)
 310 IFD<>2THEN360
 320 PRINT#1,C$;CHR$(U)+CHR$(V);D$
 330 PRINT#1,"M-W";P$;CHR$(1);R$
 340 PRINT#1,"M-R";P$:GET#1,X$
 345 IFX$=R$THEN340
 350 U=64:V=18
 360 PRINT#1,"M-R";CHR$(U);CHR$(V)
 370 GET#1,X$:IFX$=""THENX$=CHR$(0)
 380 PRINT" ";:X=ASC(X$)/16
 390 FORJ=1TO2:X%=X:X=(X-X%)*16
 395 IFX%>9THENX%=X%+7
 400 PRINTCHR$(X%+48);:NEXT
 410 U=U+1:IFU=256THENU=0:V=V+1
 420 K=K+1:IFK<8THEN360
 430 Y(0)=0:Y(4)=Y(4)+8:J=4
 440 IFY(J)<=15THEN450
 441 Y(J)=Y(J)-16:J=J-1:Y(J)=Y(J)+1
 442 GOTO440
 450 PRINT:PRINT"  ";
 455 FORJ=1TO4:Y=Y(J):IFY>9THENY=Y+7
 460 PRINTCHR$(Y+48);:NEXT
 465 PRINT"[CU]":GOTO220
 480 REM IT MAY BE NECESSARY TO CHANGE LINE 120 TO RESET SEQUENCE (108,252,255
 490 REM ON SOME DISKS. WAS ORIGINALLY (76,193,254)
READY.
```

Butterfield Speaks to the World

Users used to tell their PET/CBM machines apart by the size and shape of the keyboards, or by the message that was displayed when power was turned on. That doesn't work too well any more. New keyboards can be fitted to old machines; new ROM sets can be plugged in; and even the green screen/white isn't much of a hint any more.

Key Concepts – Jim Butterfield

Although I'm not a hardware man, I often get calls from users who want to know about some aspect of their machine. I try to establish which machine they are talking about. Commodore may have a much more official checklist for their hardware configurations: but here's the set I use. Perhaps readers can suggest other differences that are worth while knowing.

Items 1 and 2 are pretty obvious: how many columns on the screen – 80 or 40 – and what type of keyboard. I'm not concerned with tiny versus full-sized keyboards; rather, is it a simple graphics keyboard or a full-scale business ASCII layout? The easy way is to look at the top row, above the alphabetic letters: are there numbers along the top or just symbols?

The next question: does your PET have original architecture or the more recent layout? The tipoff here is the connector on the right-hand side of your machine. If you see an edge-connector – a connector with copper "fingers" extending from the board inside – you have the original machine board. On the other hand, if you can see a series of upright pins when you peer through the right hand slot, you have a more recent board. Another way of telling the same thing is to type POKE 59409,52. If the screen goes blank when you press RETURN, you have a unit with original architecture. Here's another characteristic of the early machines: if you type FOR J=1 TO 1000:POKE 32768,0:NEXT J there will be a lot of "snow" on the screen for a few moments; newer machines don't have this problem.

The next thing to test is the screen writing speed. Clear the screen, and type on the top line or two:
TI$="000000":FOR J=1 TO 600: PRINT "A"; : NEXT J: PRINT TI$
and press RETURN. You'll get a lot of letter A's across the screen, followed by a number. If the number is 000002 or less, you have a fast screen number. If it's bigger, you have a slower screen unit. Don't worry – the speed different is only seen when writing characters to the screen, and you can't read that fast, anyway.

There are a couple of minor cosmetics that are sometimes worth noting. Do you know which cassette unit plugs into the back connector? Is it cassette number 1 or 2? There's a difference in board wiring either way. And finally, if you like rummaging around the inside of your PET, how many pins do you have on your ROM chips? If you don't know how to spot a ROM chip, you don't need to know the answer to this one. While you have the lid open, count the number of ROM sockets.

There are other hardware differences that you'll find in various PET/CBM machines, but the above are the ones I ask for most often. Technical tyros will be glad to add to the list: what kind of RAM is fitted? How are the ROM sockets decoded? . . . and so on.

Many other differences that we notice between machines seem to be hardware, but they're really software. It's quite surprising how different logic can make the machine appear physically different. We'll talk about software, or ROM sets, a little later.

Some people claim that cosmetics make a great deal of difference to a computer's usefulness. I suppose it's part of your state of mind. I've seen PETS with racing stripes and others with pink polka-dots. If the owners feel that they can write better programs that way, good. I'm thinking of painting one of mine puce-coloured in the hopes that it will keep the cat away . . .

As a footnote to the article, we have now got the new 12 inch 4032's. These are easy enough to distinguish because of the size of the screen and the fact that they have 40 columns across as opposed to the standard 80 of the 80 column PETS.

Butterfield speaks to. . . . .

Wordpro 4 Plus

Because word processing is such a common use for Commodore computers in business this months column looks at Wordpro 4 Plus, from Professional Software.

This is the latest in a long line of top quality word processing programs for the Commodore machines. It works only on the 8032 model CBM computer.

The program is supplied in both 8050, 4040 and 3040 disk format, (although it would be inadvisable to run the diskette on both machines, since it spins in opposite directions in the sleeve, which is said to be bad for the diskette surface). The first action on receiving the program should be to take a security copy of the versions for the unit(s) you are going to use. Of course the 4040 version will read on a 3040 or 2040 unit. A security ROM must be installed in the frontmost spare socket of the computer.

The manual is a well-bound three-ring binder, and the contents very well laid out in tutorial style. Programmer's Notes in an appendix cater for the advanced user. Also provided are transparent self-adhesive stickers for fixing to the front of the keys on your keyboard, which are most helpful in reducing the number of times you must consult the manual!! Finally the package provides two copies of a small wallet-sized quick reference guide in a binder.

The manual contains instructions as to modifying various printers for most effective use with the program.

On loading the program, using the familiar "SHIFT RUN" command, you meet a display, asking you to state how many lines of main text you require, with a maximum of 116. This implies a maximum document length of between 1000 and 1500 words residing in the memory at a time. As the Wordpro series of programs has been developed, there has been a progressive trade-off of program length and complexity of facilities, against space available for text, and it is arguable that the limit has been reached, or even exceeded! More about this later!

You now must specify the type of printer, and disk unit device number, and there you are ready to enter text. Let me make it clear straight away that the program is full of useful facilities, and competes quite successfully with custom built word processing machines costing several times the cost of the configuration required for this package. To cover the facilities fully require a book as large as the instruction manual, so I will confine my remarks to giving the flavour of what it is like to work with the package, and to detailing features which are particularly powerful and useful .

The particular characteristic immediately noticeable is that the text goes right across the screen, filling a full 80-column line before dropping to the next. That is to say, the text is typed without any formatting being carried out at that time. The formatting is set up by various control commands, given by prefixing with a checkmark (tick), obtained by hitting OFF/RVS (which is designated "CONTROL" by Wordpro 4 plus) followed by a question mark. Such commands may be placed several on a line, and may be followed by text, separated by a semi-colon. There are those who say that failure to format as you go means that a program does not qualify for the description of a "True word processor". I cannot agree. The need to examine the formatted output is satisfactorily fulfilled if, as in the case of Wordpro 4 Plus, you can format to the screen at any time to see what the output will look like. Many will find the availability of the formatting commands for examination at any time to be a convenience, and you can see if you have left any behind when moving blocks of text around from one place to another.

Whilst you are typing the text you may use the Shift key for capitals just as you do when using an ordinary typewriter, or alternatively you may use a specially-designated key to do so: thus all characters are in capitals, save for those on the top line, which remain lower case unless SHIFTED, using the SHIFT key. Also you may go into INSERT MODE, during which you will be able to type as much text as you like, which is automatically inserted at any point, between any two letters chosen by you. This is extremely convenient, especially when editing the document. A STATUS line at the top of the screen tells you which of the various modes you are operating within, including Numeric Mode, in which you can automatically align the decimal points in numbers. This is very useful in financial work, and is even more advantagous in Wordpro 4 Plus than in previous versions, because now you can set up any number of numeric tab points, and as soon as you TAB to them, you are automatically put into Numeric Mode, and furthermore, on leaving that location, you leave Numeric Mode.

Another attractive feature, is that if you arrive at one of the numeric tab positions by means of typing or use of the cursor controls, you do **not** go into numeric mode.

The STATUS line also indicates whether you are in MAIN TEXT or EXTRA TEXT. The extra text area is used for various functions the most striking of which is the setting of information to be used in customising documents which are otherwise standardised. This is particularly useful for form letters, in which the main body of the letter is identical for each recipient, but the salutation, address block, and other key portions are different of each addressee.

The first major criticism emerges at this point: one of the more necessary moves when operating a Word processor is to be able to check the contents of the directory of the diskettes in the drives. Unfortunately in order to do this with Wordpro 4 Plus you must be prepared to lose your text! You may avoid this by calling up the directory after transferring into EXTRA TEXT, but to have that area large enough to call up a directory makes the main text area much too small for practical work. Earlier versions allowed text loss to happen as soon as you pressed CONTROL followed by the drive number. Wordpro 4 Plus requires a RETURN, but even so, there will be times when users will accidentally eliminate all their hard earned text. Also, the 8050 disk unit encourages the development of long directories, and it is easy for it

of computing will depend critically on the nature of the link between the mainframe and the micro. In the work that has been described here, the Public Switched Network has been employed. Direct links or private dedicated lines invariably provide a better quality communication facility but are often expensive unless line utilisation is high. The use of lower quality "dial-up' links puts an extra overhead on the processing sofware -the need to perform appropriate error checks in order to ensure that data transmitted over the link is not peturbed by extraneous noise. Consequently, in each of the following application examples, although it is not explicitly described, extensive error checking procedures would need to be implemented.

The examples that are described in this section represent perfectly general techniques - that is, they could be employed with any microcomputer system. However, for the purpose of illustration, they are described in terms of a particular machine - the Commodore PET.

Cross-loading of Machine Code Programs

Applications involving this type of operation fall into two general categories:

(a) those in which the 'intelligent' terminal functions as the target machine, and

(b) situations where it acts as intermediate link in a hierarchical chain.

The first of these involves straightforward cross-loading of machine code programs from the mainframe to the microcomputer terminal. This facility can be used to substantially improve its functionality, thereby making it far more useful than a conventional interactive terminal. In the second type of application the intelligent micro-terminal acts as a buffer between the mainframe and the target machine. Code to be cross-loaded is first passed to the micro-terminal where it can be checked for transmission errors and stored. Then, in a subsequent step, it can be sent from the intelligent station across to the target machine.

As far as the Commodore PET is concerned, each of these categories of cross-loading is fairly easy to perform. Consider the case of direct loading. Programs developed using a system similar to that shown in figure 5 are easily transferred from the mainframe. The object code produced by the cross-assembler is stored in a file containing a series of fixed format records. Each record contains

(a) a value, N, that gives the number of program bytes in the record,

(b) a load address that specifies where these bytes are to be loaded in the memory of the target machine,

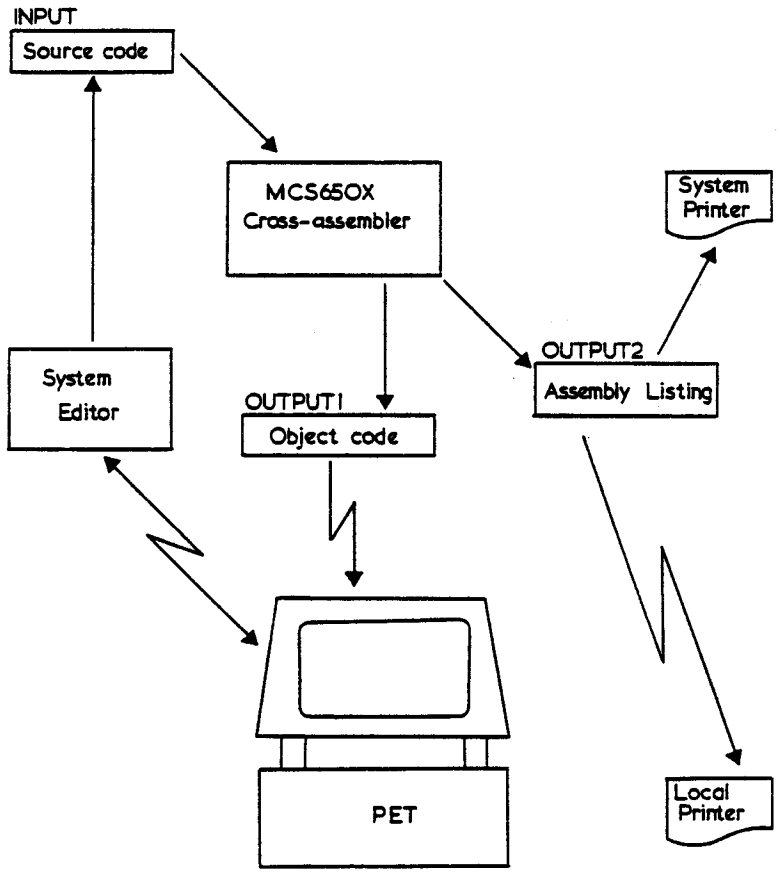
(c) the N consecutive program bytes, and

(d) a two byte checksum value.

Knowing the format of these records, it is a simple matter to formulate and implement an algorithm to perform the cross-loading operation (Bar81).

Similarly, it is equally easy to implement cross-loading in situations where the PET-based intelligent terminal acts as a mode in hierarchical link. In this context we have been experimenting with systems in which code that is transferred to the PET is subsequently cross-loaded into other PETS, a Sinclair ZX81, a Texas Instruments TMS 9980A 16-bit machine, and a variety of National Semiconductor SC/MP based systems.

Fig 5 Cross-assembler Development System



In-house Viewdata Systems

The simplest form of viewdata system consists of a series of 'pages' of information stored in the form of a computer data base. In response to queries made by users, appropriate pages of information are transmitted to viewdata terminals distributed over some particular locality or region. Both national and in-house systems currently exist. As business tools the attractiveness of these systems stems from the ease with which information can be retrieved and the colourful and graphic way in which it can be presented.

Adaptors are available commercially (Bee80) that enable the PET to be used as a terminal to the UK's national viewdata system (PRESTEL). Typically, such adaptors connect the microcomputer to a standard domestic colour TV and an auto dialer built-in modem. Unfortunate-

to become impossible to list the entire directory, because it is longer than 116 lines!! You can overcome this by using the SELECTIVE DIRECTORY command, and calling up files beginning with certain letters only, but this is a somewhat clumsy way out of the problem particularly as this command requires that there is a disk in each drive. Two points arise out of this. When designing Basic 4.0 and DOS 2.5, Commodore should have taken the opportunity to display the Directory in four columns as is done on other machines. Since they did not, I think that once the number of lines in Wordpro 4 Plus became less than the maximum required for display of a directory, it became necessary for the program to be revised so that the directory was displayed in columns, or could be called up in stages. Furthermore, since the "Universal Wedge" has shown the way to display a directory without disturbing the contents of memory, was it not possible to use this method to protect against data loss?

There are other ways of losing text of which the user should be aware : First you may exit to the beginning of the program by pressing CONTROL SHIFT RUN STOP. This has its advantages, because you can specify a different disk unit number or printer type, without switching the machine on and off, with the attendant need to remove the diskettes from the disk unit. However, it is a fact that after considerable use of the program, the use of many keys becomes automatic, and in particular, the CONTROL key becomes second nature, and because many of the operations require its use, and more importantly, some of these exit the CONTROL mode after they are carried out and some do not, it follows that the user somtimes is in CONTROL mode without realising it! The STATUS LINE illuiminates a "C" but you may fail to notice this, especially if you are not a touch typist. Now, the vital point is that the INSERT key is just below the STOP key, and the use of SHIFT AND INSERT together is frequent, especially for users who are also accustomed to programming the computer. I picture the scene: someone has been happily typing a long document, and just before they save it on the disk, they hit SHIFT RUN, instead of SHIFT INSERT, and since they are accidentally in CONTROL mode, they lose all their text!!

Furthermore, the program also permits exit to BASIC. This is achieved by hitting CONTROL SHIFT Q. Now Q is just below the inverted commas on the keyboard, and inverted commas are frequently used and require the SHIFT KEY! Need I say more? The point is that losing your text accidentally should never be allowed to happen by keyboard operation in the use of a word processor, and the acceptance of a word processor into an office, where it is to be used purely as a utilitarian device, is not the same proposition as setting an enthusiast to work using a word processor as a further extension of his computing interests. The loss of one's text, just once, is likely to create a credibility gap, which will take months to heal. Whilst I am at it, may I make some further constructive criticisms. Obviously the exits just described are a danger to users, and I recommend strongly that they should be removed. Furthermore, the program itself now takes up too much space, and text is too restricted. Now the additional features incorporated in this latest version of the program represent a considerable improvement over earlier models. However, that does not mean that all users will want all the functions, all the time. Why not include on the diskette various versions of the program, with progressively more features, so that the user can choose his own memory partitioning between functions and text space? The use of overlays is possible, although I am not in favour of this, if it can be avoided, because it is very convenient to have both drives available for data disks, so that then you may call up a variety of earlier documents in order to review texts prepared earlier. Also, what about the security ROM? I have sympathy with the wish to protect prgrams against illicit copying, as being the only way to avoid the programmer's and suppliers incomes' being damaged. However, now that EPROM programmers are proliferating, and the cost of even 4k EPROMS coming down, I would question the value of the ROM as protection. Since the purchaser of the program is paying for the ROM, and may even have to buy a SPACMAKER, Rom Pager or SOCK IT 2 ME in order to accommodate it, why not give him some value for his money? I understand from people who know about these things, that the security ROM performs a security function alone. This seems very wasteful. It may be that these ROMS are just rejects bought very cheaply, but I feel most users would be much more pleased if the 4K were used for part of the word processor program, this releasing 4k RAM for text. If it were necessary to raise the price to accomplish this, and to give recompense for the fact that part of the program would be frozen, and would therefore be costly to change, then I feel this would be acceptable to owners.

After those criticisms, what are the facilities which make me congratulate the author, Steve Punter, for the excellence of his program?

Well to start with, it enables us to use the machine as a very intelligent typewriter, and to edit what we have typed very conveniently. The cursor repeats, and even moving blocks of text about in memory is very fast. You can insert blocks of text off the diskette at any point in your text, and can find the occurrence of any string within the text. You can delete words and sentences with ease. This process is aided by the fact that the words or sentences you are contemplating deleting are highlighted in reverse field, so that you can decide whether to proceed after you have seen the potential effect. Repetitive pressing of "W" or "S" causes sucessive words or sentences to be highlighted, until you either hit RETURN, to complete the deletion, or control to abort.

Tabs can be set for text, as with numeric tabs and can be cancelled selectively or in a single operation. Strings can be found, and replaced by other strings. This facility is far more powerful than may at first be imagined. You may use abbreviations for frequently-used words or phrases,and then change them throughout the document in a single operation. However, and this is **enormously powerful,** the doucment may be linked to others on the disk, by a simple command, such that it becomes possible to do a GLOBAL SEARCH AND REPLACE. That is, the first document is loaded from the diskette, the occurrences of the search strings are found, and they are replaced with the replacement string. The revised document is then re-saved onto the diskette, the next file is loaded, and the whole process repeated, until all the linked files are dealt with. The

Continued on page 44

Continued from page 41

8050 disk unit holds over a million characters, and you can search right through to change references, fully automatically. Authors can update entire book manuscripts in this way, draft documents can be revised for incorrect references, and so on.

The GLOBAL command also allows you to print globally, so that the machine will load a file, print it, load next file, and so on, without operator intervention. Also linked files may be copied in a single GLOBAL operation.

A major innovation in this version is that simultaneous input and output are now available. This implies that the disk unit can control printing of a document, leaving you free to type more text into the computer. This is particularly advantageous if you have a slow printer, and would otherwise be held up for a considerable time. It is possible to use a sequential file to insert data to personalise standard texts, (as with mailing to customers individually), and you can even print a series of files, using the GLOBAL command. The precaution you must bear in mind is that once started, the process can only be interrupted by hitting CONTROL P, which aborts the entire operation, thus you cannot use single sheets or change daisy wheels part way through the operation. The disk unit is also tied up, so that you must remember to load any textfile which you wish to edit before commencing printing.

UNDERLINING blocks of text is easily accomplished, and if your printer allows it, you may EMBOLDEN also, failing which, you may achieve the same effect by switching off the line feed on your printer, if this is possible, and repeating the line.

Subscripting and superscripting are supported, provided you printer will accommodate them.

Many printers require special characters to enable them to carry out certain functions. These are supported, and defaults are provided for the "Spinwriter", if you do not customise the program to suit your own printer. The procedure for this is quite simple. This feature enables you to change to the red part of a ribbon, do line feeds upwards, access special characters on daisy wheels, etc

A SOFT HYPHEN is available so that any word which you choose will be hyphenated if otherwise it would leave space at the end of a line.

The latest enhancement to the numeric capability is a very important one for all users who have to produce figures in tables. By using COLUMN ADD /SUBSTRACT, the figures in a column may be added and subtracted at will, either completely or selectively, and brackets or "-" will be interpreted as deductions! This facility, combined with the NUMERIC TAB feature previously described, will be very useful in business applications.

Formatting commands: CENTERING permits any text to be centered between the left and right margins which have been specified. FORM ADVANCE will permit the printer to advance to the top of form position, if your particular printer has a form feed capability.

PAGING can be forced if you wish to start a new page, when the bottom of the page you have specified with the PG command has not yet been reached.

RIGHT JUSTIFICATION, eliminating the ragged right margin, as is done in newspaper and magazine columns, is supported. Proportional spacing, using even white space will occur if your printer can cope, otherwise spaces will be inserted. You may wish to vary the effect of this, by inserting soft hyphens, if the effect of extra spaces is bizarre!

The LN command, which does the specified number of Line Feeds, is used to economise on space. Since the way to start a new paragraph is to hit RETURN, it is wasteful to use up 80 characters of space for each line you wish to skip, the LN command thus saves a lot of potential waste of text space.

The MARGIN RELEASE will permit you to see text outside the left margin you have set.

The PAUSE command permits you to change daisy wheels, or realign paper before resuming printing.

LINKING TO OTHER PROGRAMS

A Major use of this type of program is to format and print data from other programs, and to output data for other programs to process. This is available easily in the latter case, by virtue of the DISK OUTPUT option, which permits the outputting of an ASCII file, for further processing. The manual gives helpful advice about this, and it is important to follow it, otherwise aggravating things are likely to happen! The package does not contain ways of using ordinary ASCII data files as input, although conversion programs have been published in the American magazine "Compute", and a direct-linking program for Compsoft's database "DMS" is available, so that one can use selected records from the database to provide information to be put into "form letters", to customise them. I read that the "Jinsam" Database, available in the U.S.A. also interfaces directly with Wordpro programs, and that there is a program available to check the spelling of all the words in your Wordpro file, but I have not had a chance to examine these.

Finally, I would like to say that for the person who wishes to do a lot of low-cost, high-quality word processing, this package is a worthy successor to former versions which can fairly claim to have made a very substantial contribution to the acceptance of the Commodore machines as serious business computers. With the majority of business applications for microcomputers said to be in the word processing field, it can be said that Wordpro, as much as any product, helped Commodore "come of age"!

Where is John Collins

??