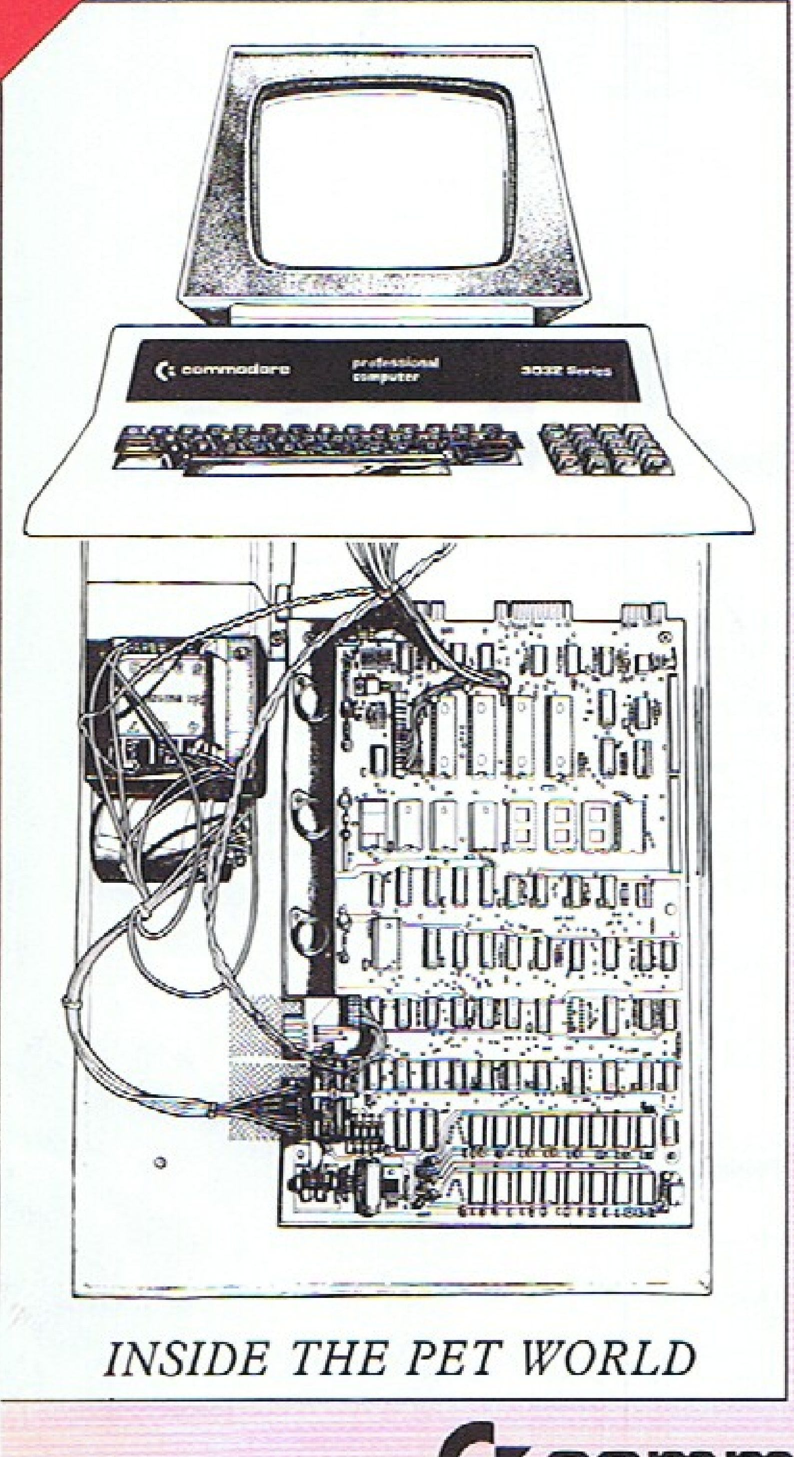
Commodore Offers Special Course Discounts to User Club Members

COMPUTER CAMP

COMMODORE CLUB NEWS

July 1981

Volume 3 Issue 6



INSIDE THE PET WORLD

APFEL BASIC

Add 30 commands to your Basic 2 or Basic 4 PET

COMAL

Implementation of a new language

8010 MODEM

An overview of the new Modem

SPOOLING

Spooling disk files to printers

EDUCATION

SECOND SPECIAL PULL—OUT SECTION

commodore COMPUTER

Contents

Educational Supplement

NEXT MONTHS ISSUE

Next months issue will be publishing the following articles:-

The start of a regular column from Jim Butterfield

The inside story of what happens to a PET when it comes into the country.

Reviews of VISICALC and WORDPRO 4 plus The start of a beginners course in machine code.

SUPERMON 4, the ultimate monitor for basic 4 PETs.

Plus all the usual programming hints and news of new COMMODORE products.

Finally the special central supplement will be concentrating on Communications.

See you next time!

Editorial

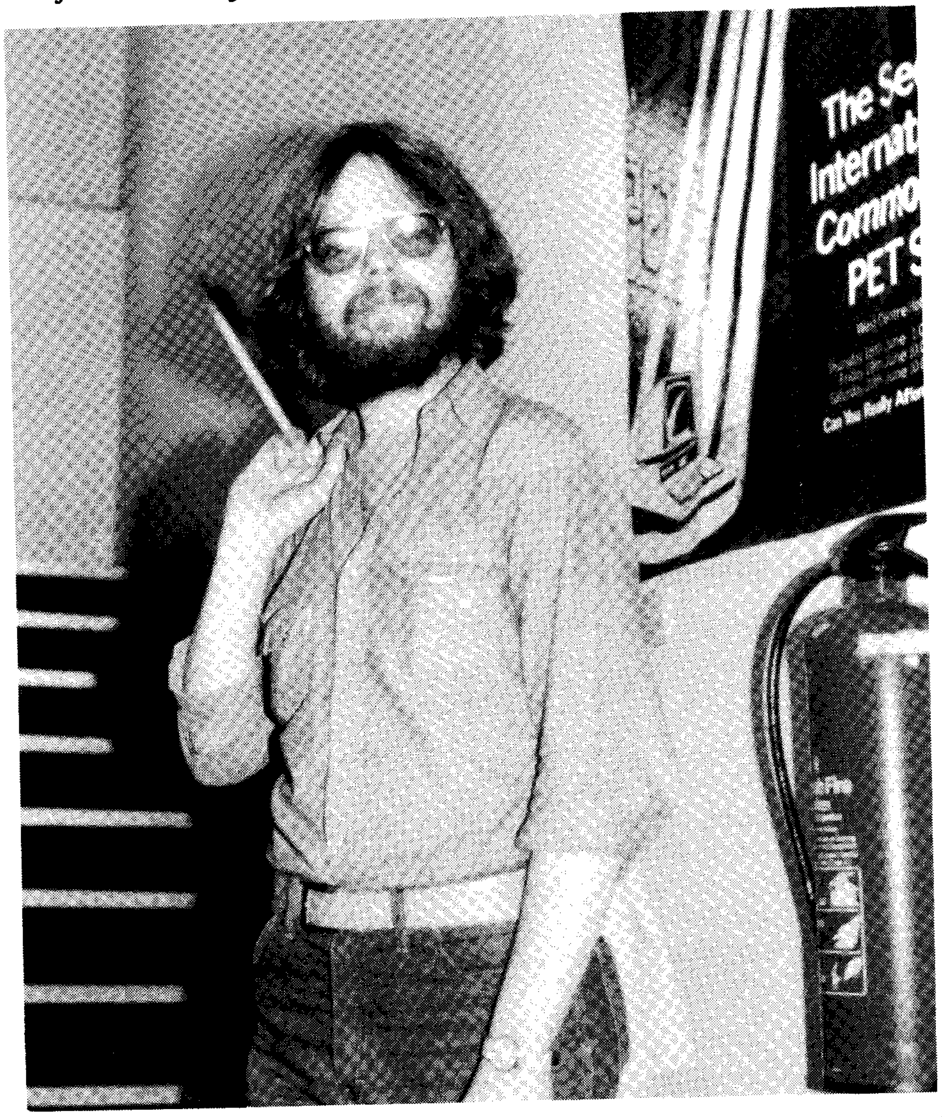
Introduction

Although you'll be reading this after the Second International Pet Show (which I hope you enjoyed if you went) it's being written before, so we'll have to save any comments on it until next time - the conclusions drawn from it and the hopes and aspirations for the future. I think we can safely assume that there's a 95 per cent chance that we'll be seeing the Third International Pet Show next year.

New Format

Thanks to all the people who've written with their comments about the new format of the magazine. So far it would appear that most of you are in favour, with a couple of exceptions who want to see more programming tips and software items. Well, that gives me two choices. Either to reduce the 'wordage' content and replace it with the requested software items, or keep all the 'wordage' and have a bigger newsletter. So, we have a bigger newsletter! I hope this appeases the people who complained last time, and also keeps happy the people who've written in with their praises. As usual, the address to reach me at is at the end of this editorial.

Editor Pete Gerrard leaving after a hard days' work



Criticisms

One of the other criticisms I've received from a couple of sources has been that we're neglecting the two ends of the scale i.e. the original 2001 series 8K so-called Old ROM Pets, and the 80 column machines. The comments are — there's nothing being done for old ROM machines any more, and there's an air of mystery surrounding 80 column Pets and their inner workings. As usual I'll try and respond to this - in this issue there is a review of an Assembler and a Disassembler for the 2001 series Pets, both only occupy a small amount of space, and are very reasonably priced, and for 80 column people we have previously published Basic 4.0 ROM entry points, and comparisons between Basic 2.0 and 4.0 ROM routines, but obviously you want more! Space precludes it this time, but the next issue will see a listing of Supermon for Basic 4 40 or 80 column machines. Supermon is an extremely powerful machine code utility that adds a number of commands to the existing PET monitor, and allows you to explore more intricately the inner workings of your PET.

Unbiased

Although this magazine is published by Commodore, we try and be as unbiased as possible in the articles that we print. There's not much point in simply publishing "How wonderful Commodore are" articles all the time - that would get us nowhere. Obviously I'm not going to put out something that slags us completely -more often than not that kind of thing is built more on fiction than fact, and the rare letter of that nature that I get is answered privately. On the other hand I am quite prepared to go to press with constructive criticism, as I'm sure that kind of thing does both sides good.

Comal

Comal has hit the world to great acclaim, and the disk going out with Comal contains 34 sample programs to get you on your way, and in this issue we have an article by one of the founding authors of Comal, Mr. Borge Christensen. It has appeared lately that every time you pick up a computer magazine there is something in there about Commodore 'vs' the B.B.C. over their choice of micro. Now, Commodore have absolutely nothing against the B.B.C. (except when I get reminders that my T.V. licence is about to run out!), so it must be stressed that the opinions contained in the Christensen article are purely those of the author AND NOT ANYONE AT COMMODORE.

Although there is the usual bimonthly educational section in this edition of the magazine, I've decided to put the Comal article in the main body of the magazine, as it is concerned mainly with progamming and the ideas behind Comal, rather than its use as an aid in the educational field.

Educational Section

The educational section seemed to go down very well last time. I've endeavoured this time around to get our educational workshops list as up to date as possible, but it's rather like sweeping leaves up in Autumn, a never ending task. So, as before, I apologise for any mistakes that I might have made, and if you can contact Jean Frost of the educational department at Slough, we'll try and get it right next time.

The release of Pascal, Assembler and Lisp on a single disk for educational establishments (detail of which were in volume 3 issue 4) has generated a large amount of interest. Articles which would be interesting in view of this would be comparison programs in the three different languages, rather along the lines of Mike Gross-Niklaus' article a couple of issues ago comparing a Basic and a machine code program that performed the same function. Timings between the three (and four, including Basic) would be fascinating to compare. So, programming experts, away you go!

Wordpro & Wordcraft

Talking of comparisons, people have in the past gone to great lengths to compare and contrast Wordpro and Wordcraft, the final assumption usually being that which ever one you use is better than the other one. I use Wordpro all the time, mainly because I've never bothered to learn how to operate Wordcraft. The lady on the next desk to mine uses Wordcraft all the time, so we can't be accused of bias! In this issue there's an article by Graham Sutherland, of Commodore's Software Products Department, on hints on using Wordpro. If you're an ardent Wordcraft fan and would care to do a similar thing so as to keep the record straight, I'd be delighted to hear from you.

On the right, Super Secretary Elaine Ryan

With the advent of Small Systems Engineering's CP/M Box, two other wordprocessing programs enter the arena - Wordstar and Magic Wand. I'd be interested to hear what your comments on those two are.

Word of Praise

Finally, a word of praise for the secretary for the Market Support department (me and Clive Booth, the Applications Manager) who goes unsung and who quietly and efficiently does a magnificent job - Elaine Ryan, thank you!

The address for any articles, snippets of information, new subscriptions, renewals etc. to be sent to is :-
The Editor
Commodore Club News
Commodore Business Machines
818 Leigh Road
Trading Estate
Slough
Berks.
If it is a subscription or a renewal, mark the letter for the attention of Margaret Gulliford, not me (unless you're using used fivers, in which case keep my name on it!).
Pete Gerrard

VIC Bits

More news on VICs. The first of the games packs have arrived, and they are superb - if you went to the PET Show you'll have seen what I mean. Any doubts I had about the 22 characters across limitaton to the screen have been dispelled completely. You just wouldn't believe that you were looking at such a screen - the advantages of having user-defined characters. On my admittedly brief look at the games they all look very addictive - a display where you can have graphics like this, and yet also produce something that can be read by many children in a classroom is a definite advantage. Detailed reviews will follow in future newsletters.

PETs on Parade

Commodore is taking part in a new computer exhibition: **'Computers for Your Business'** at the Kensington Exhibition Centre, London, 29th September to 1st October, 1981.

The exhibition is aimed at both the first-time business user and those wishing to upgrade their existing systems. A novel feature planned will be the 'Try before you Buy' room, where on display will be a representative selection of computers and word processors available to the business user - complete with working demonstrations.

Owners of PET systems will be especially welcome on the Commodore stand at the exhibition, where they will see the latest range of products and services available from the company.

The organisers, White Exhibitions of Luton, are mounting an intensive publicity programme to attract visitors, part of which will be a competition offering valuable computer equipment as first prize.

PET users will find 'Computers for YOUR Business' a worthwhile experience and time well spent. So make a date in your diary now!

Contact White Exhibitions - 11 Fairford Avenue, Luton, Beds -telephone Luton (0582) 23475/38226/32596 - for further information and tickets to the exhibition; or simply present your business card at the door when you visit 'Computers for YOUR Business'.

User Groups

A mention for Durham Pet Users Group, which has recently formed under the leadership of David Wardill, whose address is 7 Ashtree Close, Rowlands Gill, Tyne and Wear, NE39 1RA. If you're around that area and wish to climb aboad, get in touch with David straightaway. Incidentally David, if you've not yet received Comal and Adventure they're certainly in the post by now! Best of luck for the future, and if there's ever anything you want me to mention just write in and let me know, and I'll do my best to get it out in any future editions of the newsletter.

Needless to say that applies to all other User Groups as well.

Approved Products Feature

Two of the latest additons to the Approved Products range include Mills Associates and Wildes pca, who between them provide a very useful service for Pet Users everywhere in the U.K.

Mills Associates Ltd. is the newly appointed official maintanance organisation servicing Commodore products throughout the United Kingdom. The company has a nationwide network of computer centres and services a comprehensive range of peripherals - all PETs, disk drives and printers marketed by Commodore come under their wing. Their maintenance service is designed to meet the requirements of original equipment manufacturers and end-users.

Wilkes pca are distributing Safekit, a complete cleaning kit for your PET and peripherals. The Safekit contains: Safeclene tape drive cleaning fluid, Safebuds Cotton Bud sticks, Safewipes lint-free cotton squares, Foamclene anti-static foam cleanser, spun-bonded Safecloths and Safeclens anti-static VDU screen wipes, and the new Floppiclene flexible disc/diskette head cleaning system.

This means that, via the medium of the Approved Products scheme, you now have access to complete servicing of your PET computer and its Commodore originated peripherals. This is the way the scheme is designed to work - to help you, the user, find solutions to problems.

For full details of these, and other Commodore Approved Products, contact your local dealer.

Adding Sound to Your PET

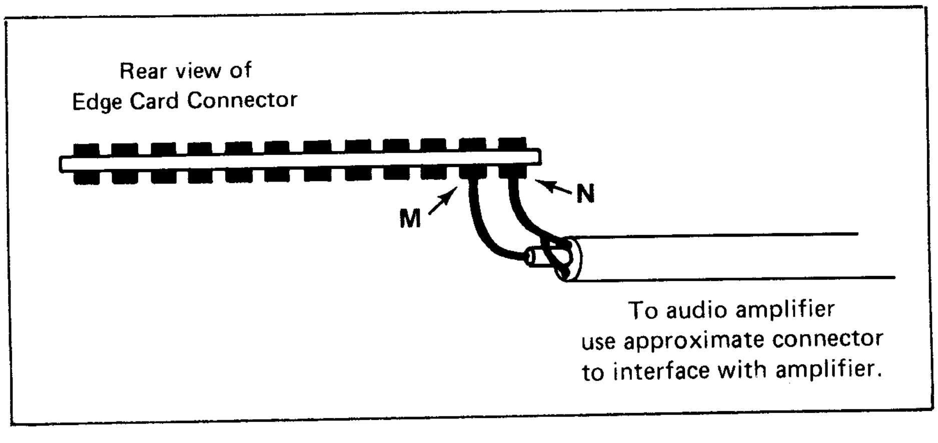
by Dave Moyssiadis

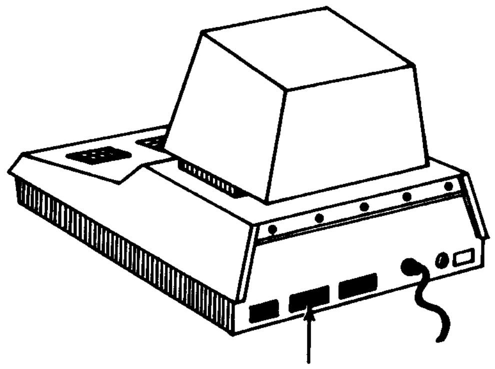
With a little hardware and a little software you can add sound to your PET and add a new dimension to such things as games. (Space Invaders is even more exciting with sound.) If you are not handy with a soldering iron you can get an edgecard connector from your dealer and hook the User Port up to your stereo system.

For the brave souls who do not cringe at the sight of a screwdriver, you will need a soldering iron, solder, and edge card connector available from your dealer and a suitable length of coaxial cable (the same stuff your stereo system is hooked up with) and the appropriate connector to input to the audio amplifier of any stereo system. Now take a look at the back of your PET and note the three edge connectors at the rear. (Fig. 1) The one on the left is the cassette interface. The one in the MIDDLE is the one we are interested in – not the one on the right which is the same size. There are a total of 24 connections on this edge, 12 on top and 12 on the bottom. On the bottom side there are pins 'M' and "N". (You may want to look at your manual under PET COMMUNICATES WITH THE OUTSIDE WORLD.) This will give you detailed diagrams of these connectors. The "M" and "N" pins are the two on the lower right side as you look at the rear of the PET right side up. "N" is the ground or shield side, and "M" is the signal connection or center conductor. Solder the edge card connector in this manner (Fig. 2) and be very careful about your work. You are fooling around with the 6522 chip which is the most expensive one on the mother board – and you can burn it up very easily. Before you plug anything in check your solder connections and cable to make sure there are no shorts either directly between pins M and N or in the patch cord you have just made. Then you can make your connections and start up the PET and your amplifier. Now comes the software part. Key in the following program:

```
10 POKE59467,16
20 POKE59466,15
30 FOR X=1TO255
40 POKE59464,X
50 NEXT
60 FOR X=255 TO 1 STEP-1
70 POKE59464,X
80 NEXT
90 GETA$
100 IFA$<>"S"GOTO30
110 POKE59464,0
120 POKE59466,0
130 POKE59467,0
140 END
```

After you key it in and run it, press "S" to stop the noise, What you should hear is a siren type sound. Next. change line 20 to read POKE 59466, 151 and run the program. Again press the "S" key to stop.





This is what is happening: POKE 59467, 16 turns on the sound, 59464 changes the pitch of the sound and 59466 changes the quality or timbre of the sound. Legal values are from 0 to 255 as with any POKE command. Experiment with different POKE values, to get different sounds. Note that POKE 59464, 1 is a high pitched sound and POKE 59464, 255 is low. Also POKE 59466, 1 is a sharp sound and POKE 59466, 255 is dull. One note of caution: You MUST turn off the sound completely by setting all three pokes to 0 before you can use the cassette LOAD or SAVE. Even if you do not hear sound you must still POKE everything off. Otherwise you will not be able to SAVE a program or to LOAD a program.

Of course, if you own an 8032, all you need is the software not the hardware! The 8032, as I'm sure you're aware, produces sound without the aid of any add-ons (i.e. the little bell that bleeps every time you reach the right hand margin). Consequently, running the sample program will work without any soldering and screwdriving. Experiment - sound with programs can be a very valuable addition e.g. it draws attention to when an imput is required. I'm sure you can think of many other uses.

What is a Compiler?

This article tries to outline the main differences between a compiler and an interpreter.

The first point to realise is that both a compiler and an interpreter are trying to achieve the same end, ie. they are both trying to provide a way of running a program. They both have to perform a similar set of tasks it is just that these tasks are performed at different times.

Consider what has to be done to 'run' a program. A program consists of a set of statements and intended by the programmer to define an algorithm, ie. it defines how a problem is to be solved or how a particular task is to be performed. The algorithm is defined in terms that are meaingful to the programmer but not very meaningful to the computer, ie. in terms of variables, operators, functions and line numbers etc.

The main tasks that have to be performed on each statement before the program can be run are:

1. the type of statement must be recognised;
2. the syntax of the statement must be checked;
3. for each variable name detected then the list of variables must be searched to see if the variable has been allocated an address, if not an address must be allocated;
4. for each reference to a line number (in a GOTO or a GOSUB) the address of that line must be determined;
5. for each expression the operator priority rules have to be applied and any brackets taken in to account in order to determine the order of evaluating the expression;
6. any non executable parts of the program such as spaces or comments (REM statements in Basic) must be skipped and ingnored;
7. finally the statement has to be obeyed.

Both compilers and interpreters have to perform all the above tasks (and others); the difference is when the tasks are performed. This is important because most statements in a program are executed more than once and often many times. An interpreter performs the above tasks every time that a statement is executed and this means that the same work can be repeated many times. Such repetition is obviously wasteful and can be very time consuming, eg. a large program can have several hundred variables so that each time a variable is referenced a long search will be required. A compiler avoids such wasteful repetition by processing a program and converting it to a different form. In this way each of tasks 1 to 6 above are performed once only for each statement and only task 7 must be performed many times. Tasks 1 to 6 are performed when the program is compiled and only task 7 need be performed every time the program is run. With an interpreter a program exists in only one form, ie. the text that the programmer has written. With a compiler the program has two forms:

-the text form;

-the converted form;

To distinguish between the two the text form is normally called the source code and converted form the object (or binary) code. The object code for a statement normally contains addresses where the source code has variable names and/or line numbers. Similarly expressions are normally re-ordered to cater for operator priority and brackets etc. Also all redundant information such as spaces, REMS and line numbers etc. is omitted and complex statements are normally broken down to a number of simple steps.

It should be clear from that by pre-processing (ie. compiling) a program a compiler can make the program run much faster but obviously the compilation process takes time. The advantage of an interpreter is that when a program is being frequently changed (eg. when it is being debugged or modified) the source can simply be edited and the program re-run. With a compiler the program must first be re-compiled before a change can be tested. The two techniques are thus complementary; interpreters are best during the program development phase but once a program is working a compiler is superior because it gives the best program performance.

Reviews

THE SMALL SYSTEMS CP/M BOX

The Commodore PET has for several years dominated the microcomputer market, but now, thanks to a spectacular breakthrough in add-on peripherals, the inbuilt limitations of the PET have been completely overcome by the SMALL SYSTEMS CP/M BOX. The PET user need no longer be confined by a limited BASIC in ROM, nor by the upper limit of 32K RAM.

Simply by plugging the SMALL SYSTEMS CP/M BOX into the PET IEEE port and loading the CP/M disk, the PET will run under the world's most popular disk operating system, CP/M (tm), allowing access to almost all microcomputer languages, including CIS COBOL, FORTRAN, CP/M PASCAL, APL, PL/1, CORAL 66, and Extended Microsoft BASIC (both interpreter and compiler), as well as the vast number of commercial accounting suites currently available under CP/M. SMALL SYSTEMS new range of cross assemblers may also be used, allowing the PET to be used as a development system for almost any microprocessor, as well as having access to the most up-to-date software. No internal connections or modifications to the PET are required, and on power-up, the PET functions as normal until the CP/M diskette is loaded.

The SMALL SYSTEMS CP/M BOX contains a Z80 microprocessor which, operating in conjunction with the PET's 6502 allows programs to be executed at 4Mhz with no wait states, resulting in greatly improved throughput.

Business applications and word processing packages designed to work with specific terminals (e.g. Lear Seigler ADM3A, Televideo 9122 or Hazeltine 1500) will need no modifications to work with the PET screen, as the SMALL SYSTEMS CP/M BOX allows the PET screen to emulate any of these devices. Further terminal emulations will shortly be available.

SPECIFICATIONS

- Full 60k byte RAM using 64k Ram chips.
- CP/M version 2.2.
- Z80 CPU running at 4Mhz with no wait states.
- Optional RS232 serial interface (with user definable baud rates) for use with a terminal or printer.
- Dimensions : 25cm x 9cm x 16cm
- Operates with any series 2000, 3000, 4000, or 8000 PET
- Supports up to 8 Commodore disk drives in any mix of 3040, 4040, or 8050 drive types.
- Diskette containing CP/M system with utilties, and full documentation included in price lists. Please specify 3040, 4040 or 8050 disk format when ordering.

PRICES

SMALL SYSTEMS CP/M BOX	£550.00 Ex.VAT.
CP/M BOX with RS232 interface	£595.00 Ex.VAT
IEEE/IEEE cable........................	£ 25.00 Ex.VAT

CP/M Is a trademark of Digital Research Inc.
Z80 is trademark of Zilog Inc.

A version of this Box for Hewlett Packard 9800 Series machines and Tektronix machines will be available shortly.

ROM X SUPERSOFT

This chip is different! You do not lose one of your precious spare sockets, because Supersoft have rewritten the routines on one of the chips which form the operating system of the machine. ROM-X goes in the UD8 socket, and by using the little bit of space which Microsoft left unused, the following extra features are supplied, and they are available immediately on switch-on. You do not have to remember, or type correctly, a SYS call:

Automatic repeat on all keys. This is the most valuable of add-on features, and once you have used it you will wonder how you ever managed without it.
Escape key. This enables you to exit from quote mode and reverse field easily.
User-definable key. This enables you to program a key to be any character you want. This sounds trivial, but some ingenuity will help speed keyboard entry. If your debugging technique requires the frequent use of a single keyword, then redefining a neighbouring key to the shifted equivalent of the second character will enable you to command the machine to LIST or RUN in a fraction of a second, particularly if you are the kind of typist who searches the keyboard valiantly for each letter!
Faster display. This speeds up the screen-printing to BASIC 4 standards, without endangering the chips in your machine in the way the "Fast Poke" did.
More accurate clock. The clock's speed is not tampered with as is the case in the BASIC 2 "New Rom machines at present, in the case of certain activities.
Lukewarm start. This gives the effect of a reset button from software, and gives a sporting chance that your crashed program will be intact, when you have followed the instructions.
Key-ahead protection This makes the keyboard buffer act in a more reasonable fashion, so that if you outpace the 10-character buffer, you do not lose everything you have typed in, as is the case with the normal operation. This is particularly important if you have keyed in a Carriage Return!
Optional Screen scrolling pause/. This means you can pause the scrolling of the screen if required.
Using only the remaining space in an existing Rom, this ingenious piece of programming, makes very full use of limited space. The limited space of course limits the number of facilities, but those provided are very useful, and the fact that they are always there is a major advantage. The use of disks and cassettes is unaffected, and major entry points are unaltered. No conflicts have been reported with any other chips, although the manual engaging expects that some will be revealed eventually! At £29.00, good value, and remember that some of the features benefit you every time you switch on the machine!

AUTORUN Supersoft

Provided that your program has been saved from address 1025, which is the case with all Basic programs, this program will give you the automatic load and run facility, as seen on the fancier commercial program.

You will get a measure of security if you kill off the run-stop key, since it will be difficult for the uninitiated to list your program.

If you, want the convenience of your programs running as soon as they are loaded then at £10.00 this is a fairly cheap way of doing it. Recommended.

MULTISORT Supersoft

Supersoft offer a variety of machine code sorts, but as Multisort provides all the facilities of their Tagsort and Speedsort, I am reviewing this one. This program will sort any single dimensional array into ASCII order, and at the same time, move the corresponding contents of one **or more** other arrays into the same sequence. These other arrays may be string, integer, or floating point. This represents off-the-peg multikey file access, and at £25.00 is very good value. It is fully relocatable, so you could put it on an EPROM and have it always available. I would not be surprised if Supersoft would put it on the EPROM for you! All these facilities operate at machine code speed, and 1000 strings in 5 seconds is pretty good going.

DTL BASIC COMPILER

This program is produced by Drive Technology, and distributed by Dataview, of Wordcraft fame. The ability to compile programs just once and to get there by a lot of time-consuming work carried out by the machine once only, is an attractive one. The approach to this problem by Drive Technology has been to say that whilst speed or operation of the final code is very important to the user, the prime consideration must be to put a Compiler onto the market, which is so compatible with the Pet Interpreter, that the only action needed is to type "Compile Fred", and the program called "Fred" on the disk will be compiled at once, and usable object code will be placed on the disk for future occasions.

I must say at the outset that this object has been achieved almost one hundred per cent. It is true that some time savings have not yet been as great as they might have been, but on the basis of providing the "Greatest good to the greatest number", the program is a success.

It functions on three logical combinations of equipment so far, there being versions for 3032, 8032 and a variety of disk drives. When ordering, you must specify which you want.

The first thing you notice is that the package has been made as user-friendly as possible. The menu which reacts to single key instructions has been carefully designed to require unshifted letters on the Business Keyboard.

Confident in the knowledge that if a program will run on the Interpreter, then it can be successfully compiled, you can set to compile your favourite program in BASIC. You need not worry if you have incorporated machine code routines in your program for speed; these will be preserved intact. Even more importantly, in these days of mutiple plug-in roms for the CBM machines, if the compiler comes accross an instruction which it doesn't recognise as a keyword, it puts that into the compiled program, absolutely unchanged. Thus, with some exceptions at present, you are able to compile programs which use commercial ROMS, without any worries. This is a considerable advantage.

Programs can be expected to run 10 to 20 times faster than on the interpreter, in that compiled code for long programs, (wherein such things matter), occupies considerably less space than does the original source program!

There are no artificial constraints, so programs require virtually no changes before compilation, and once compiled, can be run with the usual BASIC RUN command. The Basic part of the compiled program consists of a single line containing a SYS call.

True integer arithmetic is supplied, as well as real, and carefully thought out commands enable you to take advantage of the very material speed improvements available if integers are used whenever possible. You can specify that all variables are to by integer, that all are to be real, save for some specified variables. This approach will encourage even the lazy to use the most efficient program structure.

Compilation is fast:- one to two lines per second, and errors can be listed as compilation takes place if required. It is not possible to precdict acurately the likely length of the object code in terms of the reduction form the source code, simply because much will depend on your programming style. The highly informative manual makes some very important points about the bad code which results from attempts to make the interpreted programs run as fast as possible, and emphasised that if you are writing a program with the intention of its being compiled once you have it running successfully on the interpreter, then you need not declare variables at the beginning, put frequently used subroutines at the beginning, and eliminate spaces and Rems.

Thus your code should become clearer, both to yourself and to others, to the benefit of all. It should be easier to write properly-structured programs which are easier to debug and modify, and you will save a lot of time if you usually expend much effort in seeking speed imporvements.

The manual gives a tantalising glimpse of the future by listing a number of enhancements which are planned.

One enhancement which is not listed, is to speed up the disk handling. If your program is one which carries out a lot of reading and writing to the disk, you will find that, since the compiler used the rom routines, the speed improvements are limited. I expect this to be dealt with in the next few months. Also desirable is to make it possible for some sections, or all of the program to be compiled so as to run as fast as possible, allowing for the fact that generally speaking, faster code is lenghier code. Also on the wish list is the ability to locate the compiled code wherever you like in memory, and to protect it from BASIC so that even a NEW will not remove it. In this way the compiled program could coexist with a series of BASIC programs.

This is an extremely attractive package, and represents a considerable progress. Highly recommended.

Why do I need a PET anyway?

Let me start by defining the existing areas that most Pets get used in. Official research indicated that of the Pets in the U.K. (approximately 45,000 units) around 25% are in the educational field (schools, colleges etc.), 35% are used in industry, a similar amount in business, and a suprisingly (in view of the early days of Pet sales) small amount in the hobbyist market. In view of the impending arrival of the VIC the hobbyists will probably start gravitating towards that, and the original 8K Pets will re-assume their prime role in education. Since there have been a number of articles elsewhere about the VIC (and there's even one in the educational section of this magazine!), and in particular comparative reviews with other machines, we'll leave the VIC to speak for itself and concentrate on each of the other areas in turn.

In Common

What each of the three (education, industry and business) have in common, and this is due to the large number of sales in this country, is the tremendous backup available from over 200 dealers in the U.K., and from the Information Centre at Commodore itself (Slough 74111), who have a technical backup available for the public. There is also a 24 hour 'phone service for general information (High Wycombe 445211). So, wherever you are and whatever time it is, there is information to hand, which is a big plus for PET as opposed to any other micro.

Education

So, to education. When the original 8K PET with integral keyboard and cassette deck first appeared, it marked the start of a new era for schools. We'd all got used to hand held calculators and the wonders that they could perform, and then suddenly there was the PET. A unit that had a keyboard, a screen and a built in cassette deck for storage of programs. A unit where you didn't have to wait 20 minutes to load BASIC - it was there! All you had to do was switch on and go. People could see things on the screen, store programs for use by other people and pass them around different schools (an important point in an area where money is traditionally at a premium). The PET caught on - it was robust, easy to use, and ideal for teaching purposes. And it was cheap. Now of course it's cheaper still, and such a tremendous number of programs have been developed for it that it's difficult to see why a school should think of something else. People mention Apples, but have you thought of buying ROM cards, separate monitor and so on ?

Strathclyde Basic

Programs for the PET cover many areas - maths, physics, biology, english, foreign languages, and of course computer studies, to name but a few. A fine example of a computer studies program is Strathclyde Basic, developed at Strathclyde University under the tutorship of Professor Andrew Colins. This was initially an in-house project, and at the end of the first years course it was estimated that it cost just £18.00 to send each student through that course. That included equipment costs, staff salaries, the lot. Strathclyde Basic was one of the first educational programs released by Commodore.

Now there is a separate Education Department within Commodore, which has done a lot of work in co-ordinating the large number of schools who have invested in PET and are keen to share the information they have gleaned from this investment. Around 140 of these schools (designed as Educational Workshops -schools that will help others in the area keen on exploring the world of micros) were listed in volume 3 issue 4 of the Magazine, and there are more in this one. With that sort of support around, and with the vast amount of software available, it's difficult to see why education should go elsewhere.

Industry

In terms of industry, you've only got to look at the people who are already using PETs to see their potential value. People like Shell, Unilever, the United Kindom Atomic

Energy Authority, I.C.I. who have over 200 units installed - all companies who obviously know what they're doing. There are a number of Approved Products on the market now for use in industry - a prime example would be the Technical Software Centre, who were formed out of B.H.R.A., and who produce some 16 programs in this field. And of course there is as ever the benefit of the large amount of dealer support available.

Business

And so onto business, where it is perhaps easiest to immediately see the benefits of installing a PET in the office. The employer who previously spent a day working out his payroll, can do it in a fraction of the time using a PET. Stock Control can be done quickly and efficiently. How much do you spend on secretarial time doing standard letters that could be printed out using a wordprocessing program ? A filing system can easily be set up with one of the many packages available on the market. And we're still using the one system - the PET. Something that is gaining increasing importance these days is the so-called 'linking' between programs, the sort of thing that will pick out all the people who haven't paid their account for the last three months, say, and then via the word processor print them all out a letter that to them looks individual, but that you know has been the bare bones of a letter filled out by the word processor.

Programming Languages for Beginners and the Global Challenge

Borge Christensen

The first time I heard the word 'basic' was in 1946. I met a fisherman one day who told me that he had just returned from England. It appeared that he was one of the many Danish seamen who had escaped to England shortly after Denmark was invaded in 1940, and had come to serve in the English navy. At the time, when I met him, I had been learning English in school for about a year, and I was eager to know, how it was to come to England just like that. What about the language! Did he speak English when he came over? Or how did he learn it? He answered my many questions by telling me that they were all offered a course in something called "Basic English". As I asked him, what that was like, he gave me an answer that I did not fully understand then. "It was a whore of a language. It could be used to cover your immediate needs, but there was no real pleasure in it".

A New Miracle

Soon after that the word entered my English vocabulary in its general meaning, and it went on to be just another common, useful, decent English word until 1971. At that time our mathematics department got a mini-computer installed. A Data General Nova 1200. A new miracle We were all very proud and it was even mentioned in the local newspaper. Since I had taken an Algol course at the university, and was therefore considered to be the local expert - nobody knew that the course had only been a very short one, and that I remembered very little about Algol or computers in general -I was asked to take care of the "recent device" and maybe even teach some of the students to use it And then I met the word "BASIC" again in a more specific context. And spelt with capitals! After a year and several "miles" of BASIC programs, I realised that something was rotten in the state we were in. Very often I found it quite hard to find out what was going on in the students programs, and especially the faulty ones, of course.

Deficiencies

At first I blamed my own lack of profound knowledge for the whole misery, but gradually I dared to think that this marvel of a new language -that had come from THE STATES -might have some deficiencies. My suspicions were cofirmed as I started to talk about it to some of my colleagues at the department of informatics, university of Aarhus. They told me straightforward that BASIC was disaster, and that its success was totally undeserved and due to the fact that no other language was generally available on the small computers used in elemntary education. One of them, Benedict Lofstedt, took real interest in my problem - it is often very difficult for a teacher to get university researchers interested in educational problems - and he advised me to read a book that had just come out. Its title was "Systematic Programming", and it was written by one Niklaus Wirth.

Human Interface

I bought it, started to read it, and after a month I knew that the folks from Aarhus were right. BASIC **is** a disaster and to a very large extent to blame for the bad programming that was going on not only in our place, but in most other colleges and highschools. But what could be done? Pascal - the language submitted in"Systematic Programming - was only implemented on few large mainframes and used exclusively in academic circles. In those days very few people anticipated the status Pascal has today. And after all, the language is not all there is to it. The **environments** of the language are of crucial importance, too. What we today might call "human interface". The Basic on our computer - DG XBASIC - was fully interactive, supported by a good multi-user system, and included very useful file handling. We could not do without all that.

After having considered all this, I came back to Benedict Lofsted and he now suggested that we designed a few but powerful extensions to the language and system we had, and that we should use as much of the ideas from Pascal as possible. It was quite obvious, what was most urgent: **long variable names,** a **global** IF-THEN-ELSE structure (**not** the one line IF-THEN-ELSE that comes with any silly BASIC nowadays), REPEAT- and WHILE-**loops, a multi-branching** CASE-structure, and **named subroutines.** During the following six months the design was finished. I coined the work COMAL (COMmon Algorithmic Language) to name the extensions, we had planned, and in June 1974 we started to implement the facilities, mentioned above, on the Nova computer here in Tonder. The first versions of COMAL were launched in February 1975.

We were now able to write programs like this:

```
IF TRY  3 THEN
   PRINT "NO, TRY AGAIN"
ELSE
   PRINT "NO, THE COR-
RECT ANSWER IS ";RESULT
   PRINT "TYPE THAT."
ENDIF
```

I have chosen this example, because the displayed IF-THEN-ELSE structure was the one that made COMAL popular very fast. The early versions of COMAL allowed IF-THEN-ELSE branches to be nested to a depth of only four, but it is not very often you need more, and it is much easier to do it the COMAL-way than by applying GOTO statements.
You do not have to go much deeper than two levels to get BASIC programs that are at least hard to read, and most beginners will get lost in the labyrinth of BASIC statements that implement more than three nested branchings. I ought to mention that the line indentment shown in the example is done automatically by the COMAL-interpreter.

Impact of COMAL

The impact of COMAL has been stronger than we had guessed that it would be. The students write much better programs in COMAL than they used to do in BASIC, and - even more important - their programs have become **readable.** At the beginning our ideas were rejected by quite a lot of teachers. In general students were much faster to see that a good tool had come into their hands. Sometimes when I came out to talk about COMAL I got the impression that I had started some kind of a religious war and not just invented some useful improvements of a programming language that might be discussed in terms of expediency and effectiveness. Gradually the attitude changed in favor of the concept, however, and since 1977/78 it has become very difficult to sell a BASIC-computer to a school in Denmark. Most teachers will ask the salesman to come back again some other day with a computer that can run COMAL.

A new version

In 1979 I defined a new version of COMAL to be implemented on a microcomputer. This version - COMAL80 - was further improved by a working group with members from The Technical University of Copenhagen, the University of Roskilde and representatives of several Danish manufactures of microcomputers. COMAL80 includes such facilities as parameter passing (both value and reference), local variables, and recursiveness. It has been implemented on the Commodore CBM-8032 and 4032 microcomputers and two Danish made ones, the RC700 and the ICL-COMET (I wonder if ICL, England, knows that they have in fact a very good COMAL running in Denmark?).

Third and sad chapter

Happy ending, is it not? Well, not quite. I'm sorry that I have to write patient reader a third and sad chapter about "basic". A few days ago I met the word again in an important and fatal context. I got a paper from England entitled "The BBC Microcomputer. Outlined specifications of the BASIC language interpreter". The paper does not reveal its author(s), and so far that is the only trace of good taste I can find in it.

It is most laudable that this glorious institution, BBC, intends to teach the Englishmen about computers and programming. And of course those responsible for such a worthy enterprise know that it must be carefully prepared and that both hardware and software goes with it. But are they fully aware of the crucial importance of the programming language that may be used by hundreds of thousands of viewers. Not only for programming purposes, but to a still higher degree for **communication of ideas.**

Consequences

Programs for computers are of increasing importance and more are beeing written every day. These programs are meant to be used for something. They are going to **do** something which in most cases immediately influences peoples welfare in the broadest sense. Economy, health, culture. It has been said that the computer is a tool to extend the human brain in much the same way as the steam and combustion engine radically extended the potential of the human body. I think it is more precise to say that the computer extends our **linguistic potential.** Man has always been able to use the language to trigger off movements and changes in his environment. But it is new that we are now able to leave the traces of our language in tools which then, maybe long after the words have fallen and could be far away from the place where they were spoken, translate the message into action. The consequence may be new knowledge, new potentials, new wealth, but also disasters that may leave a long trace in our history. At the beginning the word was there...

Future Jobs

If one considers what language means to man in general, in our intercourse with each other, in our understanding of the world and ourselves, it is almost incredible to watch the improvidence with which we have designed and used programming languages. As the number of problems grow, it will be of growing importance that we can **read** each others programs. The message from one person to another will also in this field be of major importance over the communication with the computer.

In his book "The Global Challenge" the french author Servan-Schreiber says, in the chapter about the new information technology, "No industrial country will be able to survive the global revolution, if it does not create **future jobs** with this revolution as a starting point. The necessary readjustment consists in leaving the outdated pure commercial competition, the aim of which is the capture of markets, and which has lasted 30 years and is played out. Instead we must institute a new competition that is based upon the **education of man,** the training of brains, of the ability to create...".

Where BBC comes in

And this is where BBC and its computer course comes in. In England you have a great tradition for good programming. Names like C.A.R. Hoare and Elliot computers come to mind. And the rest of Europe. Where were languages like Algol and Pascal invented? This is where we are much better than the Americans and the Japanese.
Try to compare the miseries of Fortan and Cobol - not to speak of PL/1, the greatest scandal since the tower of Babel - with the elegance and power of Pascal, and you know what I mean.

This is where **we** can compete. And this is the field that matters in the future!

But instead of seeing this, BBC unconsciously wants you to ape after the Americans, in a field where they are definitely bad, and where you could be very good. But we have to use BASIC, the cowards yell. Anybody uses BASIC. A hell of a lot of programs have been written in BASIC. Now, take it easy. That a lot of programs have been written in BASIC, doesn't matter. They are very few compared with the immense amount that are going to be written in the future. And BASIC is not used by anybody. A lot of people are using BASIC, admittedly, but they are not the most important ones. And they are a minority compared to all those who will use computers in the future. In Denmark we have learned that people will turn away from BASIC if only they get something better.

For and Against

But BASIC is the only language that can be put into small and cheap microcomputers, is another argument.Not true. The early versions of COMAL only took up 12% more storage than the BASIC interpreter, and we had not thrown away the GOTO, GOSUB, RETURN, ON GOTO, which we could easily have done without, and which in fact have not been in use since 1976. On page four of the heroic paper about "BBC-BASIC" it says - about the extensions to a BASIC which is neither flesh nor good red herring - "These extensions should be avoided in simple programs that are intended to be used on a variety of machines.". Among the extensions is a lot of the same hopeless jumble that characterizes the rest of the design, but one or two important ideas are hid in between, namely those of **loops** and of **named subroutines.**

Meet the global challenge

And that kind of extensions should not be avoided in simple programs. On the contrary. Beginners should be urged to use such facilities, and at the same time be kept away from using the GOTO which is a very advanced tool only to be used by well trained and very good programmers. The whole concept has to be carefully redesigned, if BBC is not going to spread a mortal diesease among the next generation of British programmers and users of computers.

Meet the global challenge, where you are able. Where did British motorcycles go? And how are British steel, ships and cars these days? Is programming and control of computer systems to the the same way! Don't let it happen.

BBC, think again!

Microchess Conversion

As everyone knows, MICROCHESS 2.0 only runs on PET/CBMs that have BASIC 2.0 ROMs. For those that have upgraded to the new BASIC 4.0, the following sequence of commands will convert MICROCHESS for operation with the new ROMs.

Some copies of Microchess don't allowing re-SAVEing this easily! If yours is one of them, just issue the above POKE each time before playing. Don't re-SAVE over your 2.0 version. Use a blank!

```
     1.  LOAD "MICROCHESS"                 ; use ',8' for disk,
                                             do not use DLOAD

     2.  POKE 1055,0                       ; conversion done!

     3.  SYS 4                             ; break to monitor

     4.  .S "MICROCHESS 4",01,033A,2000    ;SAVE to Tape #1
                                            do not use Tape #2
OR
     4.  .S "0:MICROCHESS 4",08,033A,2000 ;SAVE to disk drive
                                            drive #0
```

Infinite Numbers on the PET

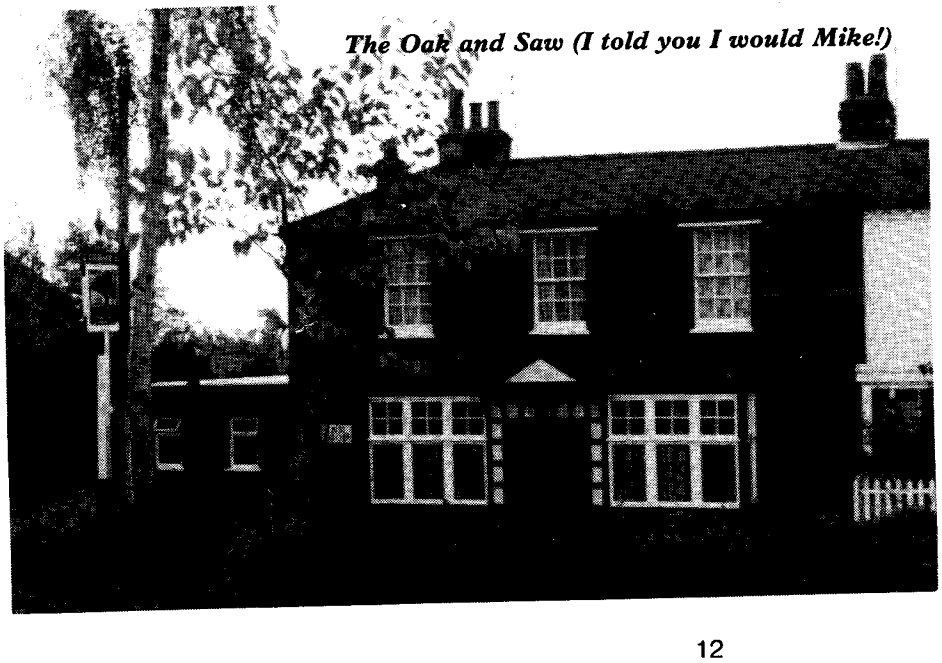
```
100INPUT"TOP NUMBER";T:INPUT"BOTTOM NUMBER";B
110A = T/B:N = INT(A):D = A-N:R = INT(D*B + .5)
120GOSUB200:PRINTNS;:T = R*10
130IF X=0 THEN .PRINT".";:X=1
140GOTO110
200N$ = MID$(STRS(N),2,LEN(STRS(n))-1)
210 RETURN
NB. Subroutine returns a character string NS for the number N.
The PET prints numbers as space,number,cursor right which causes
problems with printing numbers exactly next to each other.
```

The PET only works to 9 decimal places at the most. If you want to work out 1/23 for some obscure reason it doesn't produce enough decimal places to see if it recurrs or not. This program works out a fraction to as many places as you want. Subroutine returns a character string NS for the number N. The PET prints numbers as space, number, cursor right which causes problems with printing numbers exactly next to each other.

Assembler & Disassembler for Old ROM PETs

The following is a run down of what is available on a new Assembler and Disassembler that have been developed for the original 2001 PETs with "old" Roms. It is hoped to produce programs suitable for other PET models at a future date. Copies of the programs can be obtained from the address at the end of the article, and the cost is £12.00 for the Assembler, and £8.00 for the Disassembler. Both products come highly recommended.

```
6502 ASSEMBLER FOR PET 2001 SERIES

Before  use, the assembler must be initialised  using  the  command,
  SYS 6000. Following  initialisation, the program  occupies  memory
  locations  5832 - 8191. Two  new  commands  are added  to BASIC as
  follows:-

TO ASSEMBLE, &A
This  command must be followed by the address from which assembly is
  to commence. e.g. &A 826

TO SAVE AN ASSEMBLED PROGRAM, &S
This  command  must be followed by the start and finish addresses of
  the area of memory to be saved. e.g. &S 1000-2000 "PROGRAM NAME"

The  speed  of  assembly  will  vary  depending on the nature of the
  source code,but is approximately 25 bytes / sec.

TEXT INPUT
The  assembler  should  be  loaded and initialised before any source
  code  is  entered.Text input is to the BASIC text area,thus taking
  advantage of the PET screen editor facilities.
The minimum  line number which may be used is 1000.
Addressing  modes are specified using 3 character abbreviations,plus
  of course,  index letters X or Y, where appropriate. Abbreviations
  used are as follows:-

  IMP  implied    ACC  accumulator  ABS  absolute   ZPG  zero page
  IND  indirect   IMM  immediate    REL  relative

  e.g.  1000 LDA ZPG X 201

All  instructions  must be followed by their addressing modes,except
  in the  case  of implied or relative instructions where mode entry
  is optional.

LABELS
The first character in any label is '#'.The second character denotes
  the label type,of which there are three.

 1. #I2000
  This  label  refers to an instruction number used elsewhere in the
  text  and  may  be  used  following  JMP  or  JSR and must be used
  following all relative branch instructions.
  e.g. BNE REL #I2500,   JMP ABS #I3000
  In the  case of  JMP  or  JSR  instructions, the  actual  location
  where known,may be entered.For example,when using ROM subroutines.
  e.g. JSR ABS 62500,    JMP ABS 59000

 2. #V****
  The last four characters of this label may be any word or mnemonic
  desired  by  the programmer.This label defines a two byte variable
  for  data  storage. The  variable  is defined simply by naming the
  label in a program statement:-

    1000 #VAAAA #VBBBB #VCCCC
    1010 LDA IMM 255   #VDDDD
```


The Oak and Saw (I told you I would Mike!)

The editor's local, the Oak and Saw (only pub in the country with that name, apparently - if you know of any other, can you write and let me know ? Both I, and the landlord, would be keen to find out), in Rectory Road, Taplow, a small village near Maidenhead in Berkshire. The place where many ideas are hatched out, over a nice pint of John Courage, and where equally as many never get off the ground. Well worth a visit if you're anywhere near the area.

Each variable defined is allocated two bytes in the area immediately following the assembled program.The Lo/Hi bytes of these variables may be referenced by program instructions as follows:-

```
     1000 #VDATA          ;  Defines variable
     1010 LDA ABS #LDATA  ;  Lo byte
     1020 LDY ABS #HDATA  ;  Hi byte
```

3. #=****
   This label is similar to the #V type,except that it is used to refer to the 2 address bytes following an instruction using the absolute mode.

   e.g. 1000 LDA ABS #=ADDR

   in this way,the address used at 1000 may be altered:-

```
        2000 INC ABS #LADDR
        2010 BNE #I2030
        2020 INC ABS #HADDR
        2030 .........
```

   The address may be initialised to any required value by entering the required value before the label

```
        1000 LDA ABS 32768 #=SCRN
```

   this allocates the label to the 2 address bytes,which now acquire an initial value of 32768. The uses of these labels can best be understood by studying the following example program:-

```
        1000 LDA IMM 0
        1010 STA ABS #LSCRN
        1020 LDA IMM 128
        1030 STA ABS #HSCRN
        2000 LDX IMM 0
        2010 TXA
        2020 STA ABS X #=SCRN
        2030 INX
        2040 BNE REL #I2010
        2050 INC ABS #HSCRN
        2060 LDA ABS #HSCRN
        2070 CMP IMM 132
        2080 BNE #I2010
        2090 LDA IMM 128
        2100 STA ABS #HSCRN
        2110 RTS
```

The effect of this program is to print,on screen,the full PET character set,4 times.
Lines 1000-1030 set the start address used at 2020 to the first screen location, 32768. These lines may be omitted,and the same effect achieved,if line 2020 is rewritten as follows

```
        2020 STA ABS X 32768 #=SCRN
```

In the course of assembly,the line numbers and labels in the source text are changed to their assembled values,so that a listing of the assembled text is available.

To assemble the example program, &A826

LIST now gives

```
        826 LDX IMM 0
        828 TXA
        829 STA ABS X 32768 #=SCRN
        832 INX
        833 BNE REL 828
        835 INC ABS 831
        838 LDA ABS 831
        841 CMP IMM 132
        843 BNE 828
        845 LDA IMM 128
        847 STA ABS 831
        850 RTS
```

The assembled program may now be saved using, &S826-850

This demonstration program is included on the tape supplied, immediately following the Assembler program, so a trial run may be performed as follows:-

```
    LOAD the Assembler
    SYS 6000 to initialise
    LOAD the Demonstration program
    LIST
    &A 826 to assemble
    LIST
    SYS 826 to run the assembled program
```

Programs may be assembled to any area of usable RAM, i.e. from 826 to the top of RAM. As this implies, the source code or the assembler program itself may be overwritten. However,in these circumstances,the following points should be observed:

1. When any part of the source code is overwritten,the BASIC command NEW is automatically performed,so that the assembled text is not available for listing.

2. When the Assembler is overwritten,the command &A is automatically disabled,so that further assembly is not possible.
   The &S command remains intact,so that the assembled program may be saved. However, &S may only be used once after an overwrite, then it too is disabled. In fact,in these circumstances,the use of any tape command,such as LOAD,SAVE or VERIFY,will cause &S to be disabled. For this reason, &S should be used before any other tape operation is attempted.

Remarks may be included in the text,at line starts or following instructions,provided they are preceded by a semicolon (;).

ERROR MESSAGES

Some existing BASIC error messages are used

```
SYNTAX ERROR
UNDEF'D STATEMENT ERROR
ILLEGAL QUANTITY ERROR
```

Four new messages are added

```
ILLEGAL MODE ERROR
UNDEF'D LABEL ERROR
REDEF'D LABEL ERROR
BRANCH TOO LONG ERROR
```

These messages are self explanatory, but in relation to the BRANCH TOO LONG ERROR, the following point should be noted. This error, which occurs when the permitted relative branch displacement, (+129 to -126), is exceeded, is only detected at an advanced stage of assembly, i. e. after labels in the text have been converted to instruction numbers. Following this error therefore, it is necessary to re-enter the labels in the source code before assembling the corrected program.

DISASSEMBLER FOR PET 2001 SERIES

Start or finish addresses of the area to be disassembled may be specified in Hex. or Decimal. Where Hex. is used, addresses should be preceded by the symbol '$'.

Output may be directed to the PET screen or CBM printer. Where output to the screen is selected, the program outputs one instruction to the screen, then pauses to avoid premature scrolling. Successive instructions may be disassembled by pressing the space bar.

The program occupies RAM locations 1024 to 5400.

Patrick Walshe

Loughteague
Stradbally
Laois
Ireland

Apologies Section

Sincere apologies to everyone following David J. Pocock's series on disc drives, and to David himself, who appears to suffer more than most at the hands of the printing gremlins. I take the blame, as it is myself (the editor) who proofreads everything, and obviously I just misssed out on this one. My only excuse is that having typed it in on wordpro, read through it, corrected it, printed it and corrected it again, by the time it got to checking the typeset version I'd read through it so many times that I was seeing what I was expecting to see, not what was actually there. Please accept my apologies - it won't happen again (if it did David would probably lynch me). Anyway, the mistakes. Basically, the two times that the word PRINT appeared outside of a program, it should have read PRINT#, and on line 30 of the sample program it should have read INPUT# 15, EN,EM$,ET,ES. Finally on line 20 we should have had (CRL) rather than (CLR), (CRL) standing for cursor left. I suppose you could say we made a hash of it.

Apologies also to Dr. Porter, of the Glasgow Institute of Radiotherapeutics and Oncology, who submitted the note on the Shell sort in Volume 3 issue 4 but without getting his name mentioned. Sorry. You were missed by name this time around, when producing a speeded up version of counting on the screen.

Your turn now - the lack of an article by Clive Booth is to do with a little something called the PET Show. Lack of time is the basic problem, but Clive tells me he'll be doing something soon.

Educational Supplement

Introduction

Welcome my friends to the show that never ends. The second in our series of special educational pull-outs, a series that will be continued every other month as the newsletters go by. By now I feel we've firmly established the fact that we are appearing monthly, and the dark days of the first few months of this year, when only one newsletter appeared in over four months, are far behind us now. Volume 3 of the newsletter will run through until December, giving us 11 issues in all, and another 4 educational supplements including this one.

From then on, the newsletter volumes will be running from the beginning of the year through to the end i.e. Volume 4 issue 1 will appear in January of 1982, and issue 12 in December of 1982, continuing of course with the educational features. A further bonus will be the yearly appearance of the "Best of..." series - a compilation of articles that have appeared over the year in the newsletters, along with many other previously never before published articles. Needless to say there will be many educational features in this. It's a bit early yet to start talking about pricing, but we'll let you know nearer the date of publication. It is hoped to have this ready by the beginning of December of each year - an ideal Christmas present!

We've updated the list of Educational Workshops with this issue, and at the time of going to press we know of no mistakes in the list published. More than likely though the odd 'bug' has crept in, so if you would write to Jean Frost or Nick Green at the usual address, we'll correct it next time around. The workshops are there for your help - use them!

Pets at School

This issue contains an interesting article describing the "first terms work" at Peckham Rye Primary School, and describes how they went about using and getting acquainted with their new PETs. Also included, for comparison, is an account of a similar activity in the States, at Robbinsdale School in Minnesota. Other articles include a review of MUPET, something about the very successful regional conferences that Nick Green has been setting up, plus much more.

One of the more interesting features is one by Jan Owen, one of the cleverer men in the PET world, and who has long been a teacher with an interest in microcomputers, and has done excellent work over the years at the school he teaches at, in Crawley, West Sussex. Not aimed at the PET in particular, but at micros in general, he puts (and answers) the question -"Do all secondary schools need a microcomputer ?".

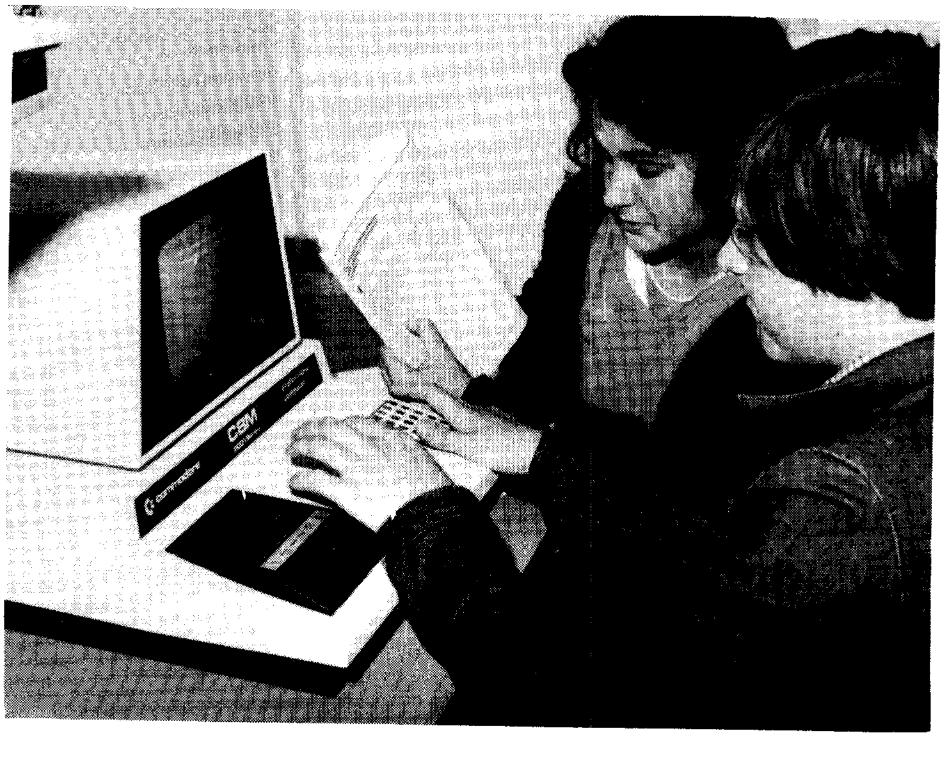
The article about the VIC has been brought about mainly by a request for more information on it, and as it seems likely that many of these will be going into schools and colleges, that's why the articles here. The VIC will revolutionize the microcomputer in education - at the low cost of £200.00 there can't be many schools, even in these financially troublesome days, that won't be able to afford one.

Elsewhere in the newsletter there is an article describing the implementation of Comal, the language which has been successfully launched into the public domain by Commodore, via the medium of the educational workshop. This article, by the founding father of Comal, Borge Christensen, has NOT been placed in this section for the reasons outlined in the newsletter editorial, but should be of great interest to all in the educational world.

Workshop Scheme

Commodore's other main release in the educational world, the Pascal, Assembler and Lisp release, has again met with great approval - I look forward to contributions from people who've used this, and Comal come to that. I'd be interested to see how you're getting along. Another benefit that the educational workshops receive is the public domain cassette releases under the watchful eyes of Nick Green, the special projects manager. There have been three of these so far, and each tape contains a large number of educational programs. If you're interested in joining the workshop scheme, get in touch with Jean Frost on Slough (STD 0753) 74111 for all the details.

With over 10,000 Pets in education, this is obviously an important section in the newsletter. If there's anything you'd like to see in it, or anything you'd like to contribute yourself, get in touch with the editor at the address at the end of the newsletter editorial.



The World of VICs

The VIC is a very important arrival in the educational arena - at only £175.00 it's possibly the MOST important arrival, certainly since the early days of the original 8k PET with small keyboard and integral casstte deck. A lot of the excitement currently being felt about VIC is indeed very similar to those early days of the PET, and the promises it holds for education, and computing generally, are even greater than when that first Pet appeared.

So why was the PET so successful, and what can we learn from that that can be applied to the VIC ? Well, when it was first launched the PET cost £695.00 (4 times the price of the current VIC) and was instantly a great success - a single unit that could put up with lots of bashing from kids, and being a single unit it was rather hard to go around pinching bits of it. So sales went from strength to strength, and as they did so grew the level of software that was available for it.

Software Compatability

One of the things about the VIC is that it is compatible with existing software (taking into account screen layout, as the VIC has a 22 column by 23 row screen, as opposed to 40 x 25 or 80 x 25 for the PETs), so that an awlful lot of existing educational software can be used or easily adapted immediately. That gives you a very firm base to start building up a library of useful software, before the machine is even commercially available!

Commodore themselves are producing and commissioning a large range of software, both on cassette and cartridge, that will cover many fields, including of course education. Thus we have an amazing range of good educational software, a mixture of programs that already exist, and ones that are under development. The future in this area looks good.

They say a computer is only as good as its software, so VIC is off to a good start then.

Complimentary Review

VIC has many other features that make it an attractive proposition to the educationalist, apart from it's price and software. These include such features as full colour, high resloution graphics, sound, expandable memory, and will include the ability to interface to joysticks, paddles and light pens, a range or perpherals including modems, disk drives and printers, plus an awlful lot more. Commodore News at the tail end of last year contained a lot of VIC information, and Printout, not a magazine that can often be accused of showing any great love of Commodore, had a very complimentary review of the VIC in its latest issue, so I don't think we need to stay too long in this area.

One little known fact, whilst we're here, is that a special IEEE interface card will be available as an accessory for the VIC, allowing it to use existing PET peripherals. How long will it be before PETs and VICs are linked up in a chain with, say, a master disk unit being accessed by them all ?

Another VIC asset is it's extreme portability. The PET was hailed as a desk-top computer; the VIC is just about a finger top computer! Very light, very mobile, making transportation from classroom to classroom very easy. And when somebody produces, as they surely will, a battery backup, the prospects for fields trips etc., recording and processing information actually on site, look very good.

But does this portability imply a disadvantage, since I stated earlier that one good point about the PET was that it was all one unit, and thus coudn't be easily pinched. I still think it's going to look a little bit obvious if someone starts walking off with a VIC, and when you consider that calculators have been going for many years without people complaining about the fact that they're portable, I don't think we've got too much to worry about. Again, going back to earlier statements about the PET the durability of the keyboard is very good on the VIC, and should pose no problems for schools pondering that point.

The VIC was launched in Japan, traditional home of electronic wizardry, to test the market - the theory being that if it sold in Japan it would sell everywhere. On its first day orders were taken for over 1,000 units, and Commodore knew they were onto a winner. The VIC is here to stay.

Microprocessors in Primary Schools- a Pilot Project in Division 8

Report on the First Term's Work

We received our two 8K "Pets" rather later than hoped - just before Easter 1980. They came into use at the beginning of the Summer Term. One or two programs were available from the Principal Adviser for Learning Resourses, who is shepherding the project; I bought a few e.g. Wordpack, English Choice, Ello, and "saved" a few from listings in Computing journals. In addition, with the limited knowledge I had gained from a "Basic" course, I began to write some short programs and also to learn how to use the "Pet".

After a week or two when I felt a little more confident I tried using the M/P with children and John West also visited to help us with this. I also talked to my staff about it and demonstrated one or two programs. Several showed interest and one, Phil Redman, enthusiasm. He quickly learnt from me and forged ahead, trying the M/P out in his own class of third and fourth year jouniors. He soon showed considerable aptitude for program writing and progress was rapid. I impressed on him the purpose of the project - to test the viability of using M/P - and my view that it should be intergrated into normal class routine and not concentrated on to the detriment of other activites. Another teacher of 3rd year juniors also used the machine from time to time.

Meanwhile I had assumed responsibility for a group of about eight very retarded readers we were worried about (aged 8-9 yrs.) with poor motivation, whom I met for half an hour each day. I found the "Pet" useful as one activity in a remedial programme, relying on "Hangman" and "Anagrams". This group continued for the rest of the term, and handled the M/P with increasing confidence.

Additional Opportunity

As an additional opportunity for the children to use the Microprocessor, a "Computer Club" was formed in the second half of the term, meeting after school once a week for 1½ hours. Its popularity necessitated strict rationing of time but several children became very keen and one or two children were helped (11 yr. olds) to write little programs, which pleased them. (It is interesting to note that they and others now at secondary school have come back to ask to be allowed to rejoin the club).

As the term went on (and the Summer Term is not the best for starting projects) more programs were acquired from various sources and Phil began to organise his work with the "Pet" effectively, writing and amending programs as he went along. In addition to basic skills programs -computation and spelling - we found an excellent simulation program in "Dairy Farm" which several children became expert at.

We have now 10 programs in regular use and several others in less frequent use. We have had help from Nick Green of "Commodore" in letting us try out new programs. We are also building a collection of subroutines on cassette which can be used for compiling programs.
Perhaps one or two observations may be permitted:

1. There are a number of good published programs but not enough. The 'simulation' section is particularly dificient.
2. Games programs are not to be despised. We have used some to help with co-ordinates, for instance.
3. There is a lot to be said for the flexibility of the Microprocessor as a teaching/learning aid.
4. It would appear that teachers without advanced training in programming can learn to write "Basic" sufficiently well to produce short programs applicable to primary schools.
5. Nevetheless it has to be recognised that it is a time consuming business (programming) especially the debugging.

L.R.TATE.
HEADTEACHER
PECKHAM RYE
PRIMARY SCHOOL

29th September 1980.

Another view

I was first properly introduced to our school PET computer just a few weeks before the end of the summer term. It had been here a little before then but I had not really had the time to investigate it properly.

However, once I had played a few games etc., I got quite interested in creating my own programs.

The first program I wrote, with the help of my headteacher, was one which would test a child's knowledge of tables. Mental arithmetic in a mixed age and ability class like mine is something of a problem. The PET seemed to offer a solution. I remember both my headteacher and I spending a whole weekend trying to work out how to put a counter in the program to work out how many the child got right or wrong. I'm glad to say we've both progressed somewhat since then.

Needless to say, the program was a great success with the children. the program is still in use today, although the version now in use has been somewhat updates as I have aquired new programming techniques.

As soon as the tables program was up and running, I started work on a spelling program. Again this is another problem area given the circumstances mentioned earlier. I then went onto write programs to help with division and addtion. My latest venture has been a program based on the Logiblock game of passing blocks through a gate. For people not familiar with the blocks, they each have four attributes being shape, colour, size and thickness the computer thinks of two of these attributes and the child attempts to discover these two by a sequence of logical steps.

Continued on page 5

Microcomputers in Secondary Schools

The first question to be asked and answered is - Do all secondary schools need a microcomputer system of some sort? The short answer to this is no! Secondary schools probably could manage to continue to teach their present curricula reasonably satisfactorily without the aid of a microcomputer, as they have had to do up to now. In a similar sense, schools probably could quite happily exist without video-cassette recorders, photocopying machines, overhead projectors and many other items of modern technology that they currently possess and use. The real reason for obtaining and using some of the afore mentioned pieces of equipment is they primarily improve the quality of teaching and secondarily they make better use of existing teaching resources. To an extent these reasons could apply in the case of a microcomputer - used in the right way it could improve the quality of teaching and make better use of teaching resources.

Most popular use

In reality schools use, or attempt to use, microcomputers in a variety of ways. By far the most popular use seems to be for teaching computer studies as an exam course from C.S.E. upwards. Many schools have struggled for years trying to teach these courses with little or no computing facilities. The most fortunate have had provided an on-line terminal to a mainframe, but the 'norm' is usually a batch processing system with a turn around time anywhere from 48 hours to 5 weeks.

Imagine the frustrations of a 14 year old who has spent several weeks writing a program and submits it only to find 5 weeks later that it failed to run because the card-punch girl at the computer installation made an error when punching out his program. In this situation it is not suprising that these schools have leapt at the opportunity of having their own equipment; but they should beware, they might be replacing one set of problems with another.

Overcoming snags

Somehow they will have to overcome the snag that one microcomputer can only service the needs of one pupil at a time, and pupils are notoriusly slow at typing and debugging. Solutions to this difficulty range from running the microcomputer in a batch-processing mode, complete with typist seconded from the office area -this seems to negate all the advantages of pupils learning interactively -to purchasing sufficient machines for one per pupil -this is expensive, and probably not a good use of scarce resources, nevertheless there are some schools doing this.

What is probably required is several machines, a lesson structure sufficiently flexible to allow some pupils to use the equipment whilst others are being taught, developing programs etc., and provision for pupils to use the equipment in private study periods, breaks etc...

Apart from those schools who have obtained equipment to support existing computer studies courses, there are many others who have purchased a microcomputer because they were relatively cheap, because the school down the road has bought one, or because they felt that they ought to do something about the microprocessor revolution that the press keeps talking about!

Justify expense

Having invested in some equipment, the school then feels that the expense involved, particularly in the current climate of educational cutbacks, needs some justification and so it decides to put on some exam course called computer studies. This is unfortunate, since rarely does the school possess anyone with experience of teaching the subject and moreover the teachers who have had any experience of computer programming are usually in the Maths and Science departments - the very two areas where there are acute teacher shortages in many schools.

What usually seems to happen is that the school orders it's machine in March or April -using up the remnants of previous year's capitation or taking advantage of the the new financial year. The machine arrives in May of June (if the school is lucky), whereupon the unfortunate teacher nominated, sometimes self-nominated, to run the course disappears into the depths of his department and spends the rest of the term trying to understand how to use the machine and its manuals.

Running Problems

It is not until well into August that he has a chance to think about the course, its contents, structure and organisation, by which time the new terms is upon him and he has to work hard to keep at least a week ahead of his pupils. In the mean time his pupils have had to opt for a course, about which they have had no prior knowledge in terms of the skills required or the qualifications to be gained.

A minority of schools have purchased, or seek to purchase a microcomputer because the headmaster or his deputies have been convinced that such a machine could take the tedium out of school administration. There are people who believe that present day microcomputers, bought off the shelf so to speak, can construct school timetables!

Jan Owen — Head of Maths Department at Holy Trinity School, Crawley.

Many thousands of pounds have been spent on research into this problem using very large mainframe computers and the best results that have been produced still fall a long way short of perfection -figures of a 95% to 98% fit have been quoted -and the programs require almost as much effort to prepare and run as it would to do the job by hand.

Schools records

Then there are schools records, to computerize these could be a very useful facility, but it would require the purchase of a fairly sophisticated system costing in the region of two to three thousand pounds, and the administration area would need to have access to it at all times, thus effectively precluding its use by the rest of the school. At the present time it would be a bold school who could afford to specalize in this way.

Does all this mean that schools should not purchase and use microcomputers unless they have the expertise and wish to run courses in computer studies? No. At the outset it was said that microcomputers used in the right way could improve the quality of teaching and make better use of teaching resources.

This does not mean getting the 'star pupil programmer' to write a tables testing program and then thinking that all lesson problems with the non-exam fourth year maths group will now be over. It means that it is possible to write or buy programs that enable the microcomputer to perform or simulate tasks that would be difficult, dangerous, unrealistic or too time consuming for pupils or teachers to do themselves.

Typical Programs

Typical examples could include the simulation of throwing several coins several thousand times to show the nature of the ensuing distribution, the plotting of the path of a satellite about a body in space, the modelling of a country's economy, the simulation of urban growth etc.. Even in this quite exciting area of use there are snags, and the biggest is getting hold of suitable software. It could be produced within the school, but it is likely to be used by a non-computer specialist, so it must be user proof and work no matter what the input.

This is quite difficult to achieve in practice and a single short program could take from ten man hours upwards to write - this is a significant amount of time compared with the time spent on normal lesson preparation! It could be 'farmed out' to a specialist software writing team, but this could be very expensive and quite useless unless the team has 'teaching' teachers on it.

Commercial Programs

Alternatively, software could be bought from publishers and software houses - unfortunately a lot of programs currently on the market are either trivial or only suitable at sixth form level and higher. The problem is then compunded by the fact that most commercial programs have been written for a particular microcomputer and cannot easily be used on others, and where programs have been designed to run on several different machines, they tend not to use the sophisticated graphics of each individual machine and the nett effect is rather drab and remeniscent of teaching programs written for output on a teletypewriter.

The comments in the previous paragraph are not meant to depress, but merely underline the problems of a technology still in its infancy -after all most of the microcomputers currently being installed in schools have only been on the market for just over two years. Certainly schools should not be disuaded from buying equipment, but they should think carefully about what they are going to use microcomputers for. When it comes to software, they should be very critical and thoroughly test programs before use. If the software is commercially produced, it must be accompanied by instructions for use and relevant examples; if it is produced internally or by another school, then it should not be passed around to other colleagues without a set of notes on its use and preferably a demonstration on how to use it in the classroom. It ruins a lesson, and puts microcomputers in a poor light if in the middle of a teaching program the computer 'crashes', drops back into its operating system or produces a ridiculous result for no apparent reason; however, if the program has been thoroughly tried and tested and comes with a detailed set of notes on its operation and use, this sort or thing can be avoided.

It has been stated earlier that good educational software takes many hours to produce -individual schools have not the resources to 'go it alone'. Schools in a given area should look at the advantages of buying similar equipment - it will allow them to learn from each others experiences, and more importantly, it will enable them to share in existing software and share in the work of producing new programs. Then as the quality and quantity of software increases, so will the quality of computer enhanced lessons.

Continued from page 3

The computer has been in my classroom every day since I first started to show an interest.

The children have all become quite competent users of the machine, with some of them able to set it up from cold. Mostly, I use my own programs working round the class in a rotational basis. First thing in the morning a group of children play 'Logiblock' together. From after then until playtime one of the math's programs will be running. After play it is usually spellings or other language work. The children keep bar charts to show their progress with these programs. They each have different levels to take account of the mixed ability.

T.V. Interface

We are also currently experimenting with the use of a T.V interface thus enabling the computer to be used for a whole class lesson.

As part of an in-service training scheme, I have been given one half day a week to develop programs and to work with other teachers and classes. I find I am using a lot of my free time with the PET in order to try and do everything I want with it, and to explore all the possibilities. It is hopeful that a lot of the work done by myself and my headteacher had been in the nature of pioneering work, particularly with the creation of software.

However, I can't deny that I have not enjoyed every moment of this period with the PET, although some of them have been frustrating. One of the most satisfying features from my point of view has been the element of discovery. Every day we learn something new. As for the children, they love their PET. They treat it with the care, affection and reverence usually reserved for their most favorite things. What's more, they are contiually learning from it in a pleasurable and enjoyable way.

Phil Redman -Peckham Rye School

Robbinsdale Schools Adopt PETs

Reproduced from "Interface" (U.S. version of "Club News")

by Bill Heck

Commodore PET microcomputers, 8K and 16K, are being used extensively in the Robbinsdale, Minnesota school district. Here is the Robbinsdale story:

Microcomputers in Education is a three-year development project funded by a Title IV-C federal grant. The main focus of the project is to develop and classroom-test microcomputer programs for the elementary school curriculum. The second and third years of the project will also be used to research the effectiveness of the programs.

PROJECT GOALS

Development: To develop and classroom-test computer programs for the elementary school curriculum.

In-service: To develop and provide in-service training for the staffs at 15 public and 3 non-public schools in the district.

Implementation: To develop a system for providing and maintaining computers, programs, and program documentation.

PROJECT HARDWARE

60 –	8K Commodore PETs
10 –	16K Commodore PETs
2 –	2040 Disk Drives
3 –	Printers
1 –	32K Commodore PET
1 –	8050 Disk Drive
1 –	Apple II with Disk
1 –	TRS 80 Level II 16K

PROJECT SOFTWARE

Software programs have been completed for use in elementary classrooms in the following areas: mathematics, science, language arts, foreign language, reading, career education, music, spelling, and special education.

Software Development

A key feature of the project is developing and testing computer programs for use in elementary schools. These programs are designed to make the computer an instructional device for use in the classroom by the teacher. Ideas for programs generally originate with the classroom teacher. The project director, Bill Heck, and facilitator, Dick Maus, with possible assistance from the relevent subject coordinator, expand and further define the programs in sufficient detail so the programmers can complete the programs.

When a program is completed, it is classroom-tested by key teachers in a minimum of three of the 18 schools. Revised programs become a part of the media center of each school.

In-Service

Each elementary staff person at the 18 sites attended two half-hour in-service sessions. This in-service included topics on computer literacy, an explanation of the project and its goals, and demonstration of a sample of some completed programs.

One key teacher from each site participated in two half-day workshops. During these half-days they were shown new programs and met with the project director and the facilitator. Eight 16-hour in-service classes were held for approximately 60 staff people. Of the 16 hours, approximately 8 hours were used to introduce the teachers to programming in BASIC. The other 8 hours were spent in lab activities related to loading, saving, using, and modifying existing software.

Implementation

A fundamental concept of the project is that computers should serve as classroom instructional aids for the teachers and students. To accomplish this, the programs must be relevant, interesting, and easy to use. One nice feature of the Commodore PET is its ability to prevent the student from "breaking out" of the program. The PET tape operating system has proved very dependable and the one- to two-minute load time becomes insignificant when a program is loaded and used in turn by 10 to 30 students for an hour or two, or possibly all day.

The media center is generally considered the "home base" for each computer. The teachers check out and move the machines to their classrooms as they would a tape recorder or any other piece of equipment.

HARDWARE

In the two years prior to the start of the project, the school district purchased one APPLE, two Commodore PETs, two OSI, and one Radio Shack TRS-80 Level II 16K Computers. The computers were evaluated relative to each other and compared with the time sharing computer used via the telephone. Since these computers are purchased by the district with no maintenance contract, cost and dependability were important considerations. Future expandability, program capabilities, and general ease of use also had to be considered. At this time, the PET is the favored computer. The district now owns 60 PETs with 40 8K PETs used within this project.

PROGRAM CLASSIFICATION

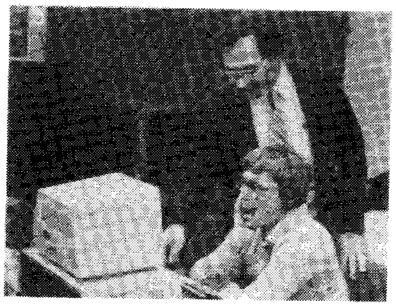
A basic assumption within the project is that the computer can support and supplement instruction at all grade levels and in any subject area. Programs usually fall into the classifications of drill and practice, simulations, or tutorial.

Drill and Practice

The computer can provide endless practice in any chosen area. It can randomly select items from a list such as a spelling list or it can randomly generate addition, subtraction, multiplication, and division problems at many levels. Given exercises can be timed as well as a summary given. Most programs are in this category.

Simulations

This type of program attempts to duplicate some real situations. One very popular program is called "Oregon", which simulates a trip to Oregon in the year 1847 with the computer randomly generating various situations that could occur: bandits, hunting for food, etc. Simulations have great value in that problem-solving and decision-making can be required of the user. One can also bring to the classroom situations which otherwise would not be possible.



Tutorials

Drill and practice, simulations, and tutorials may overlap in some programs. A drill and practice program that allows hints when incorrect answers are given has some tutorial aspects. Another type of tutorial is one that provides instruction with student interaction coming throughout the tutorial or at the end of the tutorial program.

INSTRUCTIONAL CONCEPTS

Along with the exploration of computer uses in all elementary school subjects, various kinds of instructional concepts have been examined. A description of these concepts follows:

Kid Proofing Programs

A basic philosophy of the project is that the computer should be used in the classroom by teachers and kids. Most of the programs developed within the project contain features which make it impossible to get a computer system error message. Through some subroutines within the programs, the stop key on the PET computer is disabled and the input characters controlled. This insures the teacher that students cannot accidentally exit the program or disturb the screen display.

Personalization

Initially, giving the computer human characteristics seemed desirable. After several months of usage it now seems more desirable to view it as a machine. Statements such as "I like you" and "thanks for working with me" have all been deleted from the programs. This type of personalization takes up space and offers nothing instructionally. At first it also seemed desirable to have the computer ask 'What is your name?" and to use the name throughout the program. However, after using several programs with this feature the student tended not to respond with their name but with undesirable words. Our present programs make reference to students by "Player 1" or some other technique controlled within the program.

Positive Reinforcement

Positive reinforcement can be done in many ways, from using simple words to showing something spectacular on the screen. If possible, some reward should be given. Too frequently the reinforcement for a correct answer seems to be the next problem. It is also necessary to minimize what may occur in response to an incorrect answer. If something quite spectacular happens when an incorrect answer is given, students will purposely get wrong answers. The programs do not reinforce incorrect answers in any way.

Program Duration

Several concepts were examined and used. Some programs never end. The computer randomly selects items for use and continues to do so until the computer is shut off. Teachers like this concept particularly at the kindergarten through third grade levels. Students work at the computer for a specified time limit. Some of the programs are set up to give a specified number of items, usually ten. Other programs allow the teacher or student to select the number of

items. A summary is usually given at the completion of these programs.

Whole Class Involvement

Although most computers and computer programs are meant to be used by one or two individuals, there are times when working with an entire class is desirable. A computer at each school site has been equipped with an output jack so that one or more large video monitors can be used. A program can then be introduced to a whole class or used with the whole class as an instructional activity.

Programs Requiring One, Two, Three and/or Four Students

Most programs seem to be written for a single student. Two or more students may use such a program by taking turns. An additional concept is to write programs requiring two, three, or four students. Positive feedback has been received from teachers regarding programs requiring more than one student.

Game Aspects

Students seem to prefer some kind of a game context. Two types have been used within the project. One allows students to participate in a game after giving the correct answer. The second is to put the instruction or practice within a game format. The second type of game context is preferred.

Competition Between Students

A game situation involving competition between students is quite popular. Techniques have been explored which would allow two students with different abilities to use the same program. Programs involving competition can be used with small groups or the whole class for discussions or decision-making.

Cooperation Between Students

The concept of cooperation was built into a program in the following way. Two students are given a time limit to alternately give correct answers to math facts. Each time a correct answer is given, a part of a rocket ship appears. If they can complete the rocket ship within the alloted time, it will fly. If not, the ship disappears and they must start over.

SOFTWARE

The project had as one of its major goals the development and classroom testing of 80 computer programs for the elementary school curriculum. This has been accomplished. Programs developed by category are as follows:

PET

Math – 45
Social Studies – 7
Science – 2
Directory Program – 5
Reading, Language Arts – 17
German – 2
Library – 1
Music – 1
Spelling – 9
Careers – 2
Special Education (Individualized Educational Prescription) – 6

APPLE

Math – 5
Reading, Language Arts – 3

The PET programs run on any 8K PET while the APPLE programs have been used with a 32K machine. Documentation sheets have been developed for each program with the following information:

1. Program Data
2. Description
3. Any directions appearing in the program
4. Sample run

These programs are now available for distribution.

For more information, please feel free to contact:

Bill Heck
Project Director
Robbinsdale Area Schools
Independent School District 281
4148 Winnetka Ave. No.
Minneapolis, MN 55427
Phone: (612) 533-2781 Ext. 291

Computer Camp

Two men are providing the driving force behind what could possibly become the most exciting innovation to the world of microcomputers in education ever. Computer Camp (U.K.) Limited is the brainchild of Denison Bollay, known universally as Denny, of Computer Camp Inc., and he is being helped in the U.K. by Stewart Wyley, who is a leading figure behind the publication of Summer Schools and Camps, under the auspices of The International Association of Summer Schools and Camps.

The basic idea is a series of inexpensive training courses in BASIC computing, using only Commodore PETs and disk drives, for children between the ages of 10 and 18, but before going more fully into that let's take a look at the background to our story.

The Beginnings

We must start in California, when Denny was just 13, and the world of computing entered his life for the first time. Then he was at school in Santa Barbara, where the school had one teletype terminal doing time-sharing work with a large computer bureau. He got so engrossed in this that he decided to form a computer club at the school, and had the precosiousness to ask the computer bureau if his club could have free time on the terminal, and in the grand manner in which these stories happen, the bureau agreed. So although the school itself had a budget for this of $800 per month, his club had literally thousands of dollars per month of time, for free! They had complete acess to massive computing complex, all for nothing, but with one disadvantage - it was from 4.00 p.m. to 12.00 p.m. - you can imagine the battles that went on with the parents?

This early programming was all in BASIC, and they developed a large string handling package which saw, for the first time in BASIC, something which is now so familar to PET users - the $ notation for strings. String handling in BASIC! Unheard of.

One of the many projects the group undertook was in 1968. In that year a major oil spill occured in Santa Barbara, a disaster that really saw the start of the whole ecology movement in the States. Denny and his club offered their services for a whole 10 bucks a day to the oil company concerned, who, perhaps thinking they'd got nothing to lose, took them on. The project was a great success; by consulting weather men and many other people in the field, they were able to predict where the oil slicks would spread to next, and thus the company could get there first and do something about it. Incidentally, that research was carried on and was very useful when the infamous Torry Canyon disaster hit Britain not too long after.

Eventually Denny went into the role of consultant, and had great success building and designing hardware and software. One of his more profitable jobs concerned a real-time interaction with the stock market - the computer predicted from information gleaned from the market which shares to go with, and was so succesful in doing this that the system he'd implemented paid for itself within three weeks - and it cost $100,000!

The Next Step

His consultancy, and his love of, by now, computers in general and micros in particular, lead him into the belief that more people ought to become aware of the power of these machines. Especially youngsters, who after all are the people who are going to use micros in the future. It was over lunch one day that the idea came - summer camps have been going in the States for many years - why not do one on micros ? And, after some thought, why not ?

And so, Computer Camp Incorporated was born, and the first course, for children between the age of 10 and 15 years, took place in March of 1980 at Santa Barbara, California. About 100 children went on the two week course, including a number from abroad - London, Tokyo, Paris. From all over the world, people came to use this newcomer in the world of computing. In over 300 acres of outstanding natural beauty, with all sorts of sporting and recreational facilities, with many hired staff to help out, the children were introduced to the world of microcomputing. A typical day had something like three hours on the computers, and then free time. Half the kids elected to stay on the computers, and nothing could drag them away from their beloved computer.

The course was a great success, and immediately plans went under way for a similar course in 1981. This has been booked up for some time now, and has been extended to cover the 10 to 18 year old group. Around 15% of the people are coming from outside the States to this course, including a family of 6 from Venezuela!

The U.K. Connection

And where does the U.K. come into all this ? For that we need to turn our attention to Stewart Wyley, who in the year or so before the first Computer Camp, was the managing director of a holiday company, and who set up the INFOX system for storing information on microfiche, a system used by people such as Thompson's, Thomas Cooke's and so on. This was then taken over by B.A.S. Publications, and Stewart moved to Paris to start putting in motion a similar system over there. He subsequently became involved in producing the Summer Schools and Camps brochure mentionied earlier, (which went to schools, Libraries, tourist boards etc.) Then came the 'simple twist of fate' that is necessary for events like this to happen.

Stewart was on holiday on Nantucket Island (having a sleighride ?) and saw Denny talking on T.V. about the computer camp idea. He was so impressed by this that he went to the length of tracking Denny down via the T.V. station showing the program, and the two of them met up in January of 1981.

They then went through the process of getting Computer Camp (U.K.) Ltd. set up, and started organising the first ever U.K. computer course for kids (they cover the 10 to 18 years old age group), using only Commodore PETs and disk drives. These commence at the end of July 1981, and details of the courses can be found from ringing 01-402-0050. The courses are being held at Camp Beaumont, which revolves around the St. John's Beaumont School, a fine building boasting the same architect as Westminster Cathedral. This sits on high ground above the River Thames at Runnymede, having a view down to Windsor Castle and the Chiltern Hills beyond. Around about the Camp is Windsor Park itself, with the Safari Park and the Thorpe Theme Park also nearby.

The Robot!

Computer Camp also has a robot, and in keeping with the tradition of having the same architect as Westminster Cathedral, the robot is called Charlie. One day perhaps he'll have a lady companion called Diane! Charlie will be walking around the camp, and has the ability to carry a PET with him: the PET would operate off battery, and so could be used to pass messages around, for instance, on the PET's screen. Charlie was developed at Newport Beach, California.

The Cost

The cost of these courses is £98.00 per week, and they are initially intended for children in and around the Windsor area (London etc.). It will operate as follows - children will be picked up in the morning, spend the day in and around Camp Beaumont itself, and then be taken home during the evening. The price includes everything - transport, food etc.

Although these courses are not residential, if that is the kind of course you want to send your kids on ring the number given earlier, and if sufficient interest is expressed Denny has said that they will look very seriously into the possibility of doing such courses, as indeed he has already done in the States.

What They Teach

There are five courses in total, and they start at a beginners level, giving an introduction to BASIC programming on the PET. As the weeks pass by the courses develop more and more understanding of the PET, so it is possible, by going on more than one course, to gain a very good knowledge of programming.

The equipment available at the camp consists of 20 2001 series PETS, 5 8032's complete with disk drives, and a variety of printers. There will be 50 children going to each camp, so we have a 2 to 1 situation. Ideally the use of the computers will be staggered, so that whilst one group of children are using the computers, the others will be out exploring the delights of Windsor Park and the surrounding areas, under careful control by the camp monitors of course. This then means that we have the ideal position of a 1 to 1 relationship between child and computer.

The Advantages

And what are the advantages of sending your child on such a course ? For the summer holidays there's the obvious point of getting the kids out from under your feet! But there's a lot more to it than that.

Microcomputers are undoubtedly becoming more and more important in the world of today - business, education, industry, home consumer -every aspect of the 80's and beyond. The youth of today are the people who are going to inherit this micro-revolution; it's time they learnt about it. NOW! This is exactly what Computer Camp aim to do. Not only will your kids have fun, but they'll also be equipping themselves a secure job for the future. An important point in these times or rising unemployment. By the time that a child comes out of this course he'll have gained a thorough grounding of what "The Mighty Micro" can do. When he goes into a job, he'll know what problems can be solved and, more importantly, how they can be solved - with a micro. By proving himself early on, there's a safe future ahead of him.

Who Attends?

What sort of people will your child meet ? According to Denny, the average attendant at these courses is between 12 and 13 years of age, doesn't necessarily have a knowledge of microcomputers to begin with (but a lot at the end!), male (approximately 80% of attendants are male, but all female attendants are at least the equal of their male counterparts), and all have a very good time. The world of micros, combined with the relaxed atmosphere of these courses, seems to bring out the best in everyone who attends.

The Future

This is the first course of its kind in the U.K. It is expected to expand these courses as time goes by, and to start doing similar courses in Europe and beyond. Details will appear as and when they are available.

By backing these courses, Commodore believes it is helping to ensure jobs for a large number of young people in the future. By giving away 5 free courses at the PET Show, we're helping youngsters today to get an opportunity that otherwise might pass them by. And finally, as a special offer to User Club members, we're offering a 10% discount to any member who would like to send their child (or children) to one of these courses. Simply ring the 'phone number mentioned earlier, and quote your membership number.

As stated earlier, if you are keen on residential courses, ring up anyway and see what the level of interest is. It's estimated that being in residence would add approximately £75.00 pounds to the cost of the course.

By sending your kids along, you'll be setting them up with a job for life.

A 12 year old receives instruction in the elementaries of "BASIC" at Computercamp.

Developments in Accounting Teaching

Commodore's cassette library program, "Books 2.0" is now in its third year of use in the Department of Accounting and Finance at the University of Lancaster, where the program's author, J.R. Mace is a Senior Lecturer.

The program is specially written for students of accounting, so that they may familiarise themselves with the processes of double-entry bookkeeping by seeing the impact of transactions upon displayed accounting statements. For teachers of accounting, the program allows the class teacher to illustrate the dynamic effects of bookkeeping entries, without resorting to blackboard and chalk. It also provides a framework, within which students can work out answers to bookkeeping problems without being constrained to do their own arithmetic!

Students in the past have often compounded their errors in understanding the conceptual issues in bookkeeping by making calculation errors. These calculation errors make it difficult for the student to learn the subject and the program "Book 2.0" has the distinct advantage of concentrating the student's mind on the aspect of learning bookkeeping that matter most.

The design of the program is such that the user cannot avoid learning the use of the concepts of debit and credit. This is because each transaction is entered into the system by means of the input of three numbers, being first, the account number of the account to be debited, second the account number of the account to be credited, and thirdly the amount of the transaction.

Improved Effectiveness

The program has recently been amended so that it runs on the CBM 8032 computer (as well as on the 2000 and 3000 series PETS of 8K upwards, for which it was originally written). The amended versions of the program represents an improvement in the effectiveness of the original program from the viewpoint of both teachers and students. It makes use of the "screen window" facility of the CBM 8032 to enable the display of the accounting statement framework at all times alongside the instructions for entering transactions or choosing other options within the program, whereas the original program was constrained by screen limitations to showing these items consecutively instead of contemporaneously.

This revised version (for the CBM 8032), known as "Books 8032" is available, at present from Megapalm Ltd., Halton Road, Nether Kellet, Carnforth, Lancashire, LA6 1EU. (Price £12.00, cheque with order please).

Schoolsoft

Schoolsoft is dedicated to providing low cost but high quality educational software. The programs are mainly written by teachers and are the result of many hundreds of hours of development. All the programs are in use in the classroom and each program is rewritten several times as a result of feedback from other teachers and the pupils themselves. Pupils have written some of the early games programs, assist with rewriting of programs and have contributed six of the serious programs. We intend to continue paying a rate of commission of 40% so that the long hours of really hard work can be rewarded to some extent. Schoolsoft is the CBM PET division of the MUSE software library, and will begin to supply programs translated from other machines with D.E.S. help.

Detailed description of some of our best programs:-

PEG1B Marine erosion

This program gives a continuous animated display of the gradual formation of a stack from a headland cliff, with the intermediate stages of cave and natural arch formation. A second animation demonstrates the formation of cliffs from a newly exposed land surface. Tutorial notes are displayed at the same time. This unique program has been widely admired at various conferences and shows the full capability of the Pet graphics.

PEDP2J Mark book

This program has been in continuous use for over two years in several local schools for class records and C.S.E. assessment. It enables the rapid creation of a pupil file to which any number of marks may be added. An ordered list of pupils may be rapidly run off at any time, along with the corresponding total or average mark. This program has been found particularly useful in setting the new intake on the basis of several tests.

PEEN1U Spelling tutor

This has proved to be one of our most popular programs and has been well tried in Primary schools, remedial departments and with a range of secondary pupils. Pupils are presented with a complete sentence in which one word disappears. Only correctly typed letters are accepted, until the missing word is complete. An attached printer will produce achievement reports for each pupil. A 'happy' face appears to reward correct spellings and a pleasing jingle may be played on an attached soundbox

PEM20S Sum Master

This produces a very wide range of arithmetic problems (from the simple 2 + 2 level up to any required level of difficulty e.g. 3 x 17 + 112/8). Special features include progression to harder example for high scores, an optional time limit, class list and question recording on cassette for later use and printout of the results. The program also produces work sheets and answers.

PEM3B Divisor

This program sets long division questions in five levels of difficulty for pupil or computer solution and can produce a very wide range of work cards with answers. Guidance is given in setting out the working during the calculation which is along traditional lines. Three attempts are given to obtain the correct answer and if the pupil is still wrong, the correct working out is shown. This is another highly motivating program which enlivens the teaching of a normally dull and repetitive skill.

Mupet Review

Modern Tutorial Colleges are using a MU-PET multiple-PET system to allow five 32K PETs to share floppy disk and printer facilities. The system is used for O and A level teaching, and for Computer Aided Instruction exercises, using their own machine code PILOT interpreter, MTC PILOT. In this article, David Parkinson reviews the MU-PET system.

Introduction

For some time now, floppy disks and printers have been a severe financial burden on school and college computer science departments. As they are mechanical devices, these units have remained comparatively expensive, while otherwise the cost of microcomputers has dropped consistently. Given that in normal circircumstances these units can only be used by one computer, and that most of the time they are therefore inactive, many schools have baulked at investing in them, despite the obvious advantages they offer. The MU-PET system, which allows up to eight PETs to share disk and printer facilities, promises to change this.

First Impressions

The MUPET is an extremely neat little system, obviously carefully designed and well constructed. It consists of a master control box which lives with the floppy disk and printer, a small unit to plug into the IEEE port of each PET involved, plus ribbon cable to go between the units. For the three PET system which is supplied as standard, a connection diagram is given below.

It can be seen that connecting the MU-PET is extremely easy. As soon as the system is powered up, any PET can communicate with the disks and printer, just as if it was on its own. If the bus is already in use when a PET tries to use it, the system simply waits until the bus is free, then executes the command given.

How it works

The MU-PET functions entirely by looking at what is happening on the IEEE bus, and will therefore work equally well with any combination of PET ROMs, and indeed with any system which uses the IEEE bus. The vital parts of the system are the units which plug into the PET IEEE ports, which are quite intelligent, and may well contain some form of processor. The units function as follows.

1). If a PET wants to do anything involving the disk drives or printer, a message is sent out on the IEEE port, and intercepted by the MUPET unit. The MUPET immediately completes the "handshake", which is expected by the PET to tell it that there is a device present. It is therefore impossible to get a "device not present" error unless the MUPET itself is absent or powered off, so that it is impossible for example to crash a program by atttempting to output result on a non-existent printer. The condition can however be detected under program control, as the status word ST becomes set to -128 in the event of an absent device.

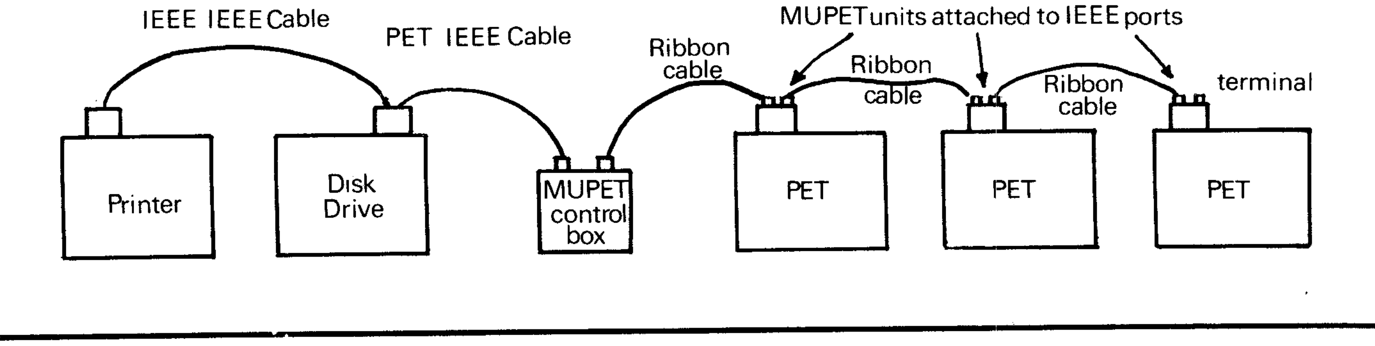
2). Having completed the initial handshake, the MUPET unit next looks at the next unit along towards the disk drives to see if it is in use. If it is , the PET is kept waiting, so that if it is attempting to load a program the PET will display the message "searching".

3). As soon as it is possible, the PETs command is passed along to the next unit, and hence eventually to the master control box and the disk unit. The command is then executed, with the revelent information being passed along the bus between MU-PET units until the transaction is completed - say listing a program or results on the printer, or saving or loading programs or data from disk drive.

4). The system is then clear until another command is issued.

Use with program files

In most school and college departments, the main use of MUPET will almost certainly be in rapidly saving and loading program files. This is particularly useful in CAI exercies, and we have found it possible to get a revision class of five students all working at instructional programs in MTC PILOT within a few minutes of the start of a lesson. It is also useful in computer science teaching, for example to let each student make a listing of his program at the end of a lesson for further study at home. We have found the MU-PET quite admirable for these purposes. The only very minor problem we have encountered is that with the present ROMs, if a save or load operation is not successful then the bus remains active until the error channel on the disk drive is cleared. It is therefore not possible for any other PET to access the bus until this is done. Mildly irritating, but it is a problem which is solved with the advent of the new DOS and BASIC ROMs.

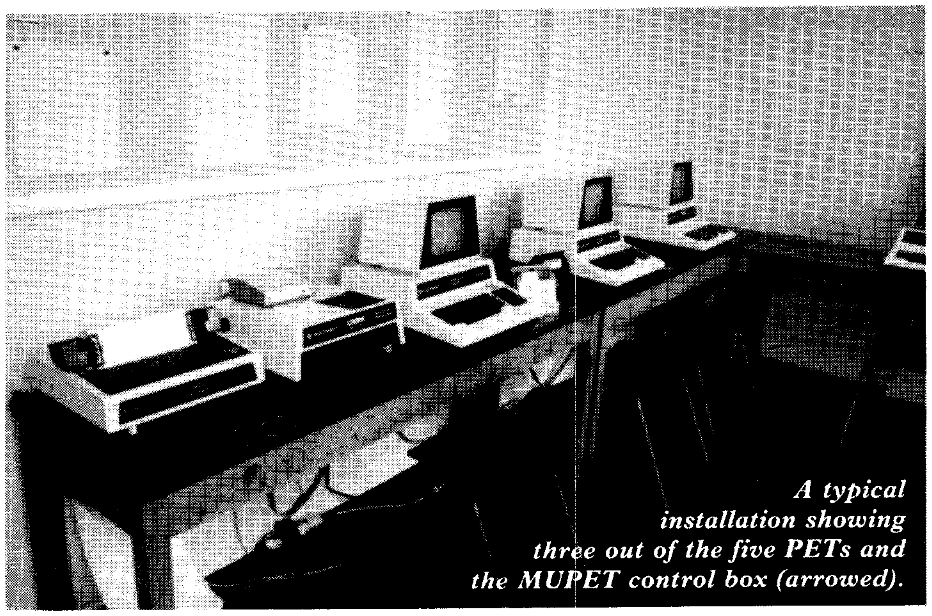


Use with data files

A second application of MU-PET is in allowing any PET to read or write disk data files, or to display results on the printer. The difficulty encountered here is that in between reading or writing data to a sequential file the bus is clear, and therefore the MU-PET has no way knowing if one particular PET has a sequential file open to the disk drives, and has not yet finished reading data. In theory, this could cause trouble, with two or more PETs muddling information from the same file. In pratice however the solution is quite straightforward.

1). The designers of MU-PET have built in a delay of about .8 seconds between the last transaction, and the bus going clear. This means that provided sequential files are dealt with by opening, reading or writing rapidly in a loop then closing, no problems will occur. In nearly all circumstances, this is the best proceedure to apply anyway.

2). In a few special cases such as Adventure, it is necessary to have continuous access to a data file on the disk drive. It is possible to do this by taking advantage of special features of the Commodore disk drives. When a sequential file is opened, a region of RAM in the disk drive is allocated, depending on the secondary address sent down the bus. This is then used as a working area for the file. Thus by making sure each PET transmits a different secondary address, it is possible to have up to five PETs each maintaining their own place in the same sequential file. Thus it would be possible for example to have five simultanious games of Adventures using the MU-PET.


A typical installation showing three out of the five PETs and the MUPET control box (arrowed).

Conclusions

The MU-PET is a very useful system for computer science teaching and for Computer Aided Learning. At about 800 pounds for a five PET system it is not cheap. However, this price is not excessive given what has to be done by the units, and certainly compares very favourably with the cost of five disk drives and five printers. We are very pleased with it, and can certainly recommend it to other schools and colleges.

Strathclyde Pascal

Strathclyde University, already well known throughout the United Kingdom as the fountain of inspiration for Scottish Engineers, and of course Strathclyde BASIC, have just bought 25 Commodore TCL PASCAL systems for the Computing Science Department headed up by Professor Andrew Colin.

Their decision to go to micro based PASCAL was essentially one of cost and reliability. Doctor Summerville relates:

"The cost of providing our service for a number of students using our main frame computers would have been greater than the cost of providing micro's, also the local University's machine could not support the required number of terminals. There was also the important aspect of service - should the main frame go down there is no service, but with 25 micro's if one goes down you still have 24, and this was at least as important as the initial cost. The cost aspect was very good experience using micro's in teaching BASIC and we were happy with this method of teaching".

Doctor Summervilles's investingation into the market for a PASCAL system he sees as a pioneering role for Strathclyde University as PASCAL is not widely used in the United Kingdom. Systems such as the CREMENCO and VECTOR were quite adequate but expensive, and it was finally clear that only 3 systems could cater for the number of keyboards within the cash limits, those were Apple II, Healthkit and the Commodore Pet. Healthkit failed to impress with their level of service, Pascal as implemented on the Apple II is not the standard ISO version, and while some of the features are worthy, their implementation was less than satisfactory.

The design of the Apple which consists of a number of plug connected separate modules caused security problems.

The final decision to buy Commodore's PETS, says Doctor Summerville:

"Was based on experience of already having some full two years in a student environment and considering them excellent value for money. Also the fact that Commodore could mass produce a product almost all of which worked properly, which is not necessarily true of other manufacturers".

Robox Ltd, Commodore dealers in Glasgow supplied the PASCAL systems and also serviced the other Pets as Strathclyde. John Thomson, Director at Robox said:

"The educational market is very important to us as it represents substantially greater than 50% of our hardware market and the PET is particulary suitable because of its intergrated keyboard, screen and processor".

When will we see **Strathclyde Pascal?** Final word from Doctor Summerville:

"There is a real need for a good teaching manual, but it will not happen before the end of 1981".

1000 Educationalists Attend Micro Conferences

Up to a thousand teachers, advisers, inspectors and others interested in the educational possibilites of microcomputers must have participated in the highly successful conferences organised by Commodore that were held from September to October, in Glasgow, London, Manchester, Bristol and Birmingham. Attendance was limited only by the number of micros that could conveniently be brought to the conference venues, but most 'first timers' were able to try out the machines.

One of the principal speakers at the conference was Dr David Burgess of Cranfield College of Technology. Dr Burgess has considerable experiences in teaching teachers to use microcomputers.

He urged teachers to get "hands on" experience at all costs and whilst not convinced that all teachers should learn to programme, he felt that man/machine interaction was the most reassuring experience that prospective users could have.

This became a matter that was discussed at all the conferences and opinion seemed broadly to suggest that no programming skills were necessary to use micros in classrooms, providing the appropriate application software existed. In many cases, of course, this software does not yet exist.

"Computer in the Curriculum"

Teachers were given copies of "Computer in the Curriculum" a leaflet produced by Edward Arnold describing software for topics, e.g. Physics, Chemistry, Biology, Geography and Economics with an additional unit on Home Heating. Science Simulation material has been available for slightly longer and covers topics such as Population Dynamics, Plant Competition, Interferance and Diffraction, Enzyme Kinetics, The Synthesis of Ammonia, Chemical Reaction Kinetics, Scattering, Orbits and Satellites, Evolution and Genetic Mapping, both student and teacher notes. Interest has been such that Chelsea College, who distribute the software, have decided to get into the copying business themselves.

Five or six hundred pieces of software

Other curriculum areas are covered less systematically (and the majority of software falls into this category of "Teachers' Aids") e.g. remedial work in spelling, arithmetic, algebra, graph plotting in mathematics and simulation of physics experiments. In total there must be some five or six hundred pieces of software that fall into this category and the recently formed teachers' software exchanges in Shropshire and Bradford account for two to three hundred of these titles. Commodore are seeking ways of making this software more widely available through their public domain scheme, which was launched at the Conference.

Workshop Software scheme

The Workshop Software scheme is a means of obtaining software free for your school or college. If you are involved in developing software, teaching teachers or familiarising teachers with the use of micros in schools (as are many schools who have pioneered the introduction of microcomputers), then you should write to Jean Frost at 818 Leigh Rd, and she will send you a workshop application form. On completion and return of this you will receive a large number of programs with subject matters varying from arithmetic to political science and for age ability ranges from 3 years to 1st year University. The software has come mainly from Canada and the United States but illustrates the use of many techniques which teachers will find useful. Some of the programs are written to a very high standard indeed. Commodore hopes that teachers will themselves contribute and that they will extend and improve programs already in the public domain. Already, over 200 workshop centres have registered and it is hoped that this will encourage the exchange and development of software.

Without any extentions to the scheme a considerable supply of "Teachers' Aid" material could become available in this way. However, at the London Conference, teachers were adamant that they require more systematically planned software, covering larger areas of the curriculum.

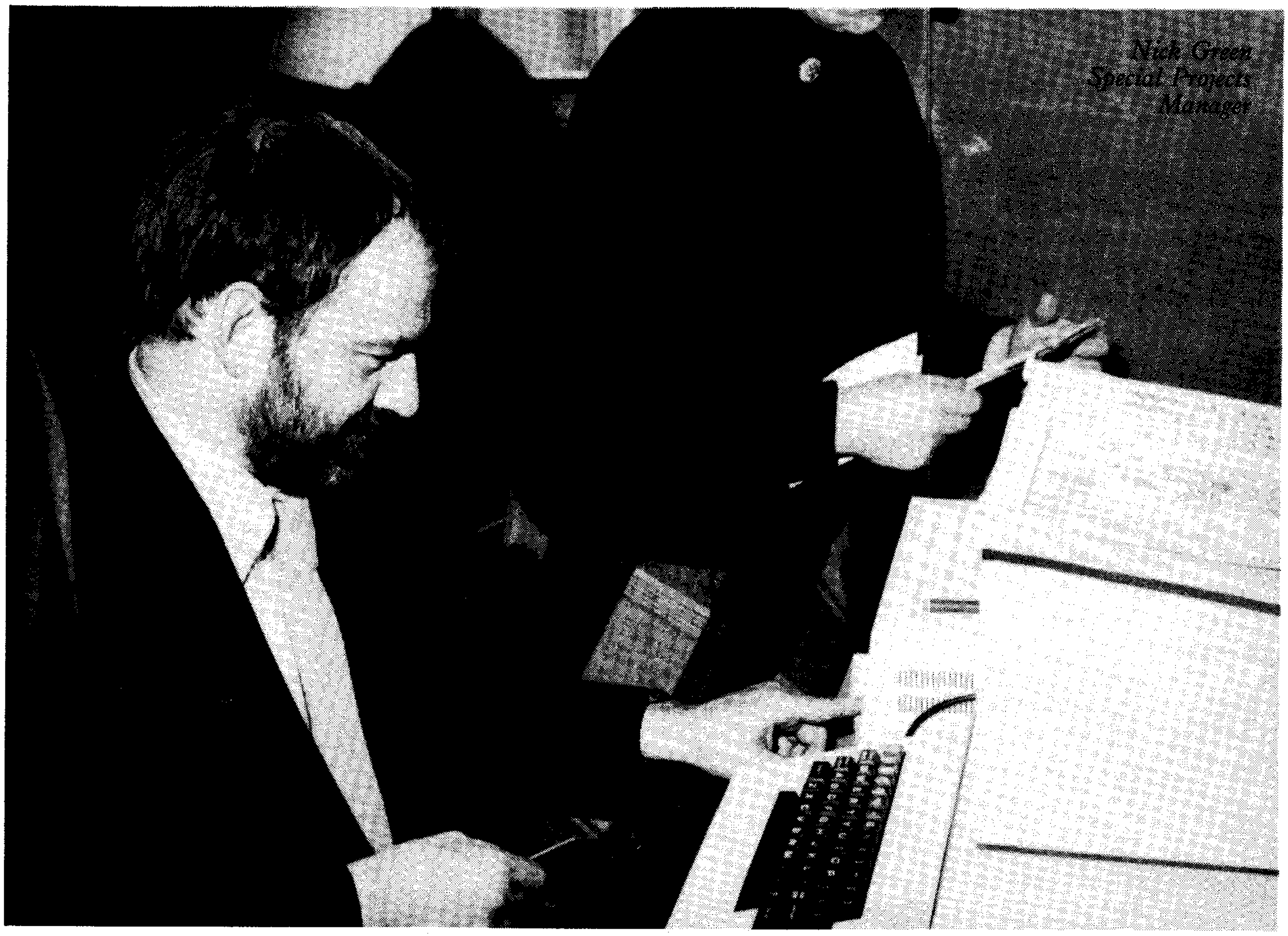
This became a major theme at the Glasgow and Bristol conferences and the proposal was made that Subject Matter Experts should be identified. These might be teachers with experience who know their particular subject area well or teachers who have written text books or even revision notes. (The revision notes that many publishers are producing could be most helpful).

Topic Lists

Having identified a teacher or group of teachers who can function as subject matter experts for a particular part of the curriculum, a list of topics should be drawn up which reflects the syllabus for the particular subject to a particular standard. Commodore announced that they would be happy to publish these lists (as indeed I am sure would Educational Computing), so that teachers working with micros might be able to programme one or two topics each. The Commodore Education Group is actively discussing standards at the moment. This is an ad hoc group comprising David Burghes, Chris Smith and Nick Greeen and we are on the look-out for people who would like to get more involved. Clearly the recently proposed MUSE standards are a useful departure point. It is the intention to coordinate this work through the software workshops. But first we must have the topic lists.

Clearly, the first versions of this software are not going to satisfy everyone and it will only be by the time we get to two or three versions that we will begin to feel that something substantial is being produced. Hopefully the introduction soon of telesoftware techiniques will facilitate this development considerably.

For the future, some work will soon be available on frame or topic management programs and this will


Nick Green
Special Projects
Manager

enable much more freedom for students who want to tackle a particular area of the curriculum. The topic lists can be converted into topic maps in which prerequisite topics are defined and a topic management system is envisaged with several hundred topics on a batch of floppy disks with which students may interact as they wish. This will have the effect of beginning to realise that dream of CAL enthusiasts - the "pupil centred" approach.

Schools administrative applications received attention at every conference, particularly in Manchester where some considerable work has been done. Keith Johnson has published a book with Hutchinsons called "Timetabling", which contains lists of programs for the Pet. Many teachers might find this useful but beware of undertaking major administrative projects on behalf of the Headmaster! Time is very easily eaten up. The most useful compromise that was suggested was to give pupils project work in the area of schools' administration.

The Manchester and Glasgow conferences were distinguished by having two teachers who have, in partnership with other teachers in their area, formed their own software companies : Schoolsoft which was represented by Paul C. Thompson of 67 Anderby Drive, The Willows, Grimsby, South Humberside and Qwertysoft, represented by its founder Jim Cocallies of 20 Worcester Road, Newton Hall, Co.Durham.

Some of the initial difficulties experienced by both companies seem to have been overcome now and many people are enthusiastic about the quality of their programs. They have the great advantage of having been developed by teachers for teachers in classroom.

Success Assured

Educationalists have also made their contribution to the series of conferences - notably David Walker at Glasgow, David Tinsley at Birmingham and Kathleen Hennessey at Manchester. Typically they argued for more resources to teach teachers and in many cases expressed doubts of the capabilities of many schools to teach examinable computer science. However all were enthusiastic about the use of micros and whilst having reservations would, in general, support the enthusiasm of local Pet gurus, e.g. Alan Goodall in Manchester, Les Davison in Bristol, Mike Bawtree in London and Peter Massey and Trevor Lusty in Birmingham, for getting a micro (particularly a Pet) at any cost into the hands of teachers and pupils and for them to have a go. Again, everyone had reservations about this approach but all agreed that if the micro was introduced into a school and most importantly, if free access to all was granted by the headmaster, then success was usually assured. The machine should not be locked away in a cupboard - it should be available for taking home by staff and senior pupils and it was said that Norwich Union ran an appropriate insurance scheme.

These teachers generally expressed the view that senior pupils could be very useful in teaching others to use the machine and in particular helping the teacher himself with some of the more arcane aspects of use. Here machines were broadly being used in maths and science.

There was a loud and clear call for more software in the non-numerical fields of language teaching, history etc. Languages software in beginning to appear now with titles in both the Commodore and Petsoft lists. For English Literature, History etc less is known but the program "Policy" in Workshop Software 1, which can be obtained from your local software workshop, demonstrates a model of American Presidential policy which could be an inspiration to those traditionally non-numerical fields.

"Add ons"

In general equipment was supplied by dealers for the conferences but occasionally other colleges in the area chipped in with a loan of machines and this was greatly appreciated by everyone. Commodore's dealers demonstrated high resolution graphics, TV Monitor interfaces and Multipet systems. Recently low cost interfacing peripherals have become available and now for £50 an A-D or D-A converter can be obtained. Also a set of switches and LEDs together with instruction manual which fits on to the user port and is known as the User Port Laboratory is available through local dealers at the same price. These peripherals should prove most helpful in that difficult area to get started in of process and laboratory instrument control and monitoring. The existence of this equipment now makes it possible, say in the context of a practical laboratory period, to tackle an interfacing problem.

The highly successful poster, "Anatomy of a Pet", recently produced by Commodore, was given away to conference participants and whilst no first time users will understand all the entries on it, this at least gives a record of what is to be learned in order to understand the microcomputer fully.

Requests to put on conferences at other population centres around the country have been comming in and we look forward to events in Southampton, Liverpool, Newcastle and Aberdeen soon. If you wish to find out more we suggest you get in contact with your Local Authortiy Adviser or local Commodore dealer.

Lastly, the idea was floated that a full-blown software conference be mounted some time at the end of the next summer term. Again the Commodore Education Group should be consulted : Chris Smith, David Burghes or Nick Green. Teachers who would like to contribute should send a brief abstract of their intended paper to one of the above-mentioned; if possible before February 1st 1981. Needless to say, we will be interested in papers addressing problems of software standards and we will be interested in papers by subject matter experts describing topic lists. Indeed, we can hope that by then a major new suite of curriculum software will have been written by software workshops. Papers discussing such projects would be highly prized.

Footnote

One observation that was repeatedly made at the Conferences, both from the floor and by leading speakers, was that pupils who are unruly in class became much more co-operative when in front of a microl In fact, one speaker recommended that diffficult children be given the task of explaining the micro to the rest. It is interesting to think that computers, far from dehumanising people might be an aid to socialisation.

Chris Smith of the Pet in Education Users' Group can be contacted at:
Queen Elizabeth College
University of London
Campden Hill Road,
London W8

Educational Workshops

The following is a list of amendments to the list of workshops published in volume 3 issue 4 of the newsletter. Copies of that earlier list are available to new subscribers of the newsletter, who ask for their subscription to be backdated to January of 1981.

Only one correction to the existing list, in the Yorkshire section - D.W. Ellingham, Marcliffe Junior and infant School, Marcliffe Road, Sheffield S6 4AJ, who have 1 system and who provide courses for teachers.

Additions to list :-

College Name & Address	No of PETs	Courses for Teachers
LONDON		
T. Lyth, South Thames College, Wandsworth high Street, London SW18 2PP	6	Many
Zelda Issaacson, North London Polytechnic, Prince of Wales Rd., London NW5	2	Yes
D. St. J. Gray, Sidney Chaplin High School, Folly Lane, Walthamstow, London E7	3	H & A
P. Gibson, rowan High School, Rowan Rd., London SW16	4	Internal
C.R. Mitchell, Walmsley House, The City University, St. John St., London EC1	2	No
Essex		
F. Bonner, Kingsdown School, Snakes Lane, Southend-on-Sea, Essex	1	Yes
V. Buhagier, Bishop Ward Comp. School, Beacontree Heath, Dagenham, Essex	2	Yes

Wordpro 4 Tips

Graham Sutherland

How many of you have sat down at your PET and thought "I'll have a go with this Word Pro package I've just bought, it can't be that difficult!" and within fifteen minutes you have given up in despair? If that situation has ever happened to you this article may be of some use, and hopefully if you are a dedicated user of the Word Processing package you may find some useful time-saving hints in here as well.

Formatting your page

The first and normally the hardest thing to do is to format your page, as it will be printed, in the format that you require. This will depend on the size of stationery that you use, the type of printer you use, and the style of presentation of the document that you prefer. Due to the complexity of the page formatting, for the first time user, it is a good idea to create a format that you are happy with, and re-use it again and again.

This is done by using the lm (left margin), and rm (right margin), the pp (the number of lines on the page), and the pg (the number of lines you wish to use for printing) commands. This will give you the three defined borders to your page, as you define the top of the form by the positioning of the page in the printer. All that you now have to decide is whether to use the right margin justification, normally used on reports but not on letters; this command like the text centering is 'switched on', with a 1 following the command, and 'switched off' with a 0 following the command. A typical document will use the following commands to set the format of the page -lm10:rm70:pp66:pg62:jul - this is for A4 stationery, using the right margin justification. Note that the command string starts in column 2, and is immediately preceeded by the tick mark (which is produced by typing CONTROL and the QUESTION MARK, and will be referred to from now on as CNTRL), and must always be superceeded by the return key (or right arrrow).

You will notice that the individual commands, when on the same line are separated by a colon; you will see that this is always the case. IF the last command in a string is followed by the text of the document it must be followed by a semi-colon, and not a return.

Comment Lines

In previous section I mentioned that once happy with a page format you should re-use it again and again. I shall now show how to use this to your best advantage. There is a command (cm) which will create a comment line, this means that anything on a line that starts with 'CNTRL'cm: will not be printed as output, this then lends itself very nicely to being used for notes that you can insert into the text, and that will not be printed.

You may ask how this helps you to re-use the same format. It is very simple, you use the comment lines to describe which format you are using, so you can immediatley tell whether this format will fit your requirements. A sample is shown below:-

```
'CNTRL'cm:**************************************************'RET'
'CNTRL'cm:** This document is set up as plain A4, using  **'RET'
'CNTRL'cm:** 7Ø cols and 62 lines per page.              **'RET'
'CNTRL'cm:** OK with ASCII or CBM printers,              **'RET'
'CNTRL'cm:** remember to check the printer type before   **'RET'
'CNTRL'cm:** starting to print.                          **'RET'
'CNTRL'cm:**                SAVE FILE FIRST !            **'RET'

'CNTRL'cm:**************************************************'RET'
'CNTRL'lm1Ø:rm7Ø:pp66:pg62:jul'RET'
```

None of this will be printed, but it is displayed on the screen, at the front of the file, for your guidance. The format commands are on the last line.

There are two other useful tips about the comment lines, the first being something that I will advocate more than any other of these tips, and that is to use the first line in EVERY file to show the file name as a comment line, as below:-

```
'CNTRL'cm:Blank File'RET'
'CNTRL'cm:**************************************************'RET'
'CNTRL'cm:** This document is set up as plain A4, using  **'RET'
'CNTRL'cm:** 7Ø cols and 62 lines per page.              **'RET'
'CNTRL'cm:** OK with ASCII or CBM printers,              **'RET'
'CNTRL'cm:** remember to check the printer type before   **'RET'
'CNTRL'cm:** starting to print.                          **'RET'
'CNTRL'cm:**                SAVE FILE FIRST !            **'RET'
'CNTRL'cm:**************************************************'RET'
'CNTRL'lm1Ø:rm7Ø:pp66:pg62:jul'RET'
```

This allows you to always know which file you are in, and can save you a lot of time and effort. One of the most common mistakes that I and many others make is the incorrect spelling of the filename, when we wish to recall or memorize a file. If you always have the filename as the first line in the file, you can get to that line by pressing CLR/HOME once to get you to the top of the screen, and again to get you to the beginning of the file. If you have access to a machine try and repeat the following commands. Having loaded the program, load a disk into drive 0, then position the cursor over the first character of the filename, now press CONTROL and CLR/HOME together, press M to memorize, press 0 for the drive number with the disk in and now press the oblique key (to the left of the cursor up/down). The filename as you typed, now appears on the control line as the filename you wish to store this file under. If you always do this you save keystrokes by not having to retype filename again and again, and you will not have the original and revised files stored under mis-spelt versions of the file.

The other use for the coment line is when checking the spelling of grammar in a large or complicated document. I frequently use a comment line inserted on a blank line to show how far through the file I have reached before one distraction or another. To do this all that is required is :-
'CNTRL'cm: ** CHECKED TO HERE **'RET'

This legend is enough to get me straight back to where I left off before I was interrupted.

As you can probably see using the filename as the first line of the file makes the storing of the file far easier, and more accurate, and using the comment line to show how far you have got, you will be able to see the ease with which you can achieve another goal, for which you will be thankful, and that is ALWAYS store the file when you leave the machine, even if only to turn round and answer the telephone.

The Use of a File Index

I have cause to send quite a number of letters and memos, that do not need to be typed by a tyist. I have a disk for memos and another for letters, and they contain the program, a bland format of the document, and an index file. This allows me the luxury of using more than the 16 characters to describe what the file contains. Allow me to explain further. All my letters and memos have a filemane that is in the format of 'Letter XXX' or 'Memo XXX' where XXX is a number allocated sequentially. Every time I am about to write a letter or a memo, I call up the file 'Letter Index' or the file 'Memo Index', make the appropriate entry, re-store the index file. Then call up the blank format, write the filename into the comment line as the first line of the file, and then type the letter or memo. Having completed the letter or memo I then store the file onto the disk before printing. An example of an Index file is shown below:-

```
'CNTRL'cm:Memo Index'RET'
File    To Whom   Date      Subject'RET'
====    =======   ====      ======='RET'
Memo 1  J.Smith  16/4/81    Bonus scheme re-payments'RET'
Memo 2  J.Bloggs 17/4/81    Prod'n of sales statistics'RET'
```

Output to Video

The more ambitious you get, the more complicated and intricate the documents you produce become. This obviously means that the text has to be laid out in a format that will look right when you come to print it. This is when you will be very glad of the facility to display the document as it will be produced by the printer.

To produce the document on the screen you must be in 'control mode', in other words the 'C' on the control line must be lit up by the cursor, then press 'O' for output, and follow this by 'V' for video. If the format of the document does not conflict with any of the commands given the screen will be filled with the document as in printed format. The document will be displayed and all you do is press any key, (except RUN/STOP) and the text will scroll vertically. If you want to break out of this , press RUN/STOP and CONTROL. If you transcend page boundaries you will see a continous line across the screen, and you may wonder why you have lost the first line on the new page, this is because you cannot give line feed commands from the head of the page, with the lf command. If you want a blank line you have to press 'RET' in the first column of the line you wish to be blank. At the end of the document you will see a continuous line across the screen with three 'up arrows' in the left of the screen, to regain the text editing mode you now have to press 'CONTROL'.

If at any point the screen reverts back to the text edit mode this is because you have a FORMAT ERROR, and while you were pressing a key to scroll, the error message appeared, and then disappeared in response to your finger gamely struggling on trying to scroll the text in 'output to video'. A format error will normally occur, for example, when you specify the left margin as 10 for the body of the text, re-specify the left margin as 15, for indenting a section, and trying to give the left margin release command a value of more than 5, in other words further to the left than the first left margin.

```
The indent and left margin release are used to produce a section that
looks like the following : -

        a) The left margin has now been set at 15, it was 1Ø.
        b) The left margin release is set at 3, this allows the first
           line to start in what is position 12.
```

As you can see the lines have been reduced in length, the margin release allows the 'numbering' system to be put outside the normal constraint of the left margin.

Reprinting

When printing large documents you will find, to your anger, the spelling mistake that slipped through, or the printer ribbon runs out half way down the page. There is a simple way to get round this particular hurdle, if you remember what I said about the comment lines being used to describe the format of the document. Apply the same thinking to the problem of the printout that only got part of the way through, re-use the same format, and use the comment lines as previously described.

You will I hope remember the bulk of the above, the additions to the previous versions are to show that this is to reprint a file. The note about the page numbering is to remind you that you are not starting the pagination at page 1, you could be anywhere through the document, in the example above the page numbering will start again at 16. This assumes that you have printed pages 0-15 successfully. All you now have to do is 'RECALL' the file you were printing with the cursor positioned below the bottom line of the example above. You now remove the sections that have been printed, making sure that the complete document is safely stored, and commence the printing from page 16.

```
'CNTRL'cm:Reprint'RET'
'CNTRL'cm:***************************************************'RET'
'CNTRl'cm:** Use this to reprint all reports, but not      **'RET'
'CNTRL'cm:** letters or memos.                             **'RET'
'CNTRL'cm:**              PAGE NUMBERING  !!               **'RET'
'CNTRL'cm:***************************************************'RET'
'CNTRL'cm:** This document is set up as plain A4, using    **'RET'
'CNTRL'cm:** 70 cols and 62 lines per page.                **'RET'
'CNTRL'cm:** OK with ASCII or CBM printers,                **'RET'
'CNTRL'cm:** remember to check printer type before         **'RET'
'CNTRL'cm:** starting to print                             **'RET'
'CNTRL'cm:**             SAVE FILE FIRST !                 **'RET'
'CNTRL'cm:***************************************************'RET'
'CNTRL'lm10:rm70:pp66:pg62:jul:p#16'RET'
```

Use of the Extra Text as a Notepad

For the more advanced user, a tip I discovered which is very useful when composing the document on the screen. I am moderately proud of the fact that I can type equally as fast as I can write, and why write anyway when it has only to be transcibed later ? If you have the luxury of being able to compose the document as you type, you will find the lack of a notepad somewhat of a disadvantage. Do not despair, why not use the Extra Text ? If you are not using constants with the append function or variable data blocks, there is at the very least one screen of spare memory not being used. You can jot down your notes as and when you think of them, and gain access to them with 'CNTRL'X. You can then look up your notes as and when you like

As will be aware the first question you are asked, after loading the system is 'Lines Available : 151 How many for main text ?' this is because there is a trade-off depending on the value you input, of the amount of memory available to the main text and the amount of memory available to the extra text. If you select the maximum of 151, you will only have 23 lines available in extra text.

Use of Command Strings

As mentioned in the previous section, there is a trade-off of space in the main text against space in the extra text. There is one way to minimise the waste of space in the text, and that is to put as few commands as possible into the text. As you will have seen from the earlier examples you can put many commands onto one line, thus saving space, but perhaps the most commonly mis-used of the functions is the line feed. How many people do you know who resort to a 'RET' in column one of the line to produce a blank line, and how many times do they have to do it to format a document.

You can achieve the same as 10 lines with a 'RET' in column one as you can with 'CNTRL'ln10; all you loose here is six characters, as you can continue typing text after the semi-colon, as opposed to ten wasted lines. Ten lines is nearly 7 per cent of available space wasted.

The Frills

At last we come to the frills, the bits and pieces that make the effort worthwhile. The HEADER and FOOTER commands are the frist two we will look at.

The header and footer commands allow you to reserve an area at the top and bottom of the pages, that you cannot otherwise use. The format of the commands are as follows:-

'CNTRL'hd2:Test Document,1/5/81'RET'

'CNTRL'ft4:,-   -'RET'

In both of these commands you specify a number, in the header it releates to how many blank lines you wish to be printed between the header line and the start of the text, and in the footer command it relates to how many lines from the bottom of the page the foot will be printed. You will cause a FORMAT ERROR if the difference between the pp and pg values is, for example 6, and the footer specifies 6 blank lines.

In both cases the comma is to indicate that the first of the three preset fields is not to be used. The output from the hd command is as follows :-

Test Document   1/5/81

The 'less than' and 'greater than' symbols placed in this position in the ft command will cause automatic page numbering. Always leave at least one space either side of the 'less than' and 'greater than' symbols, as when you reach double figures the result looks a little crampled. The output from this example of the ft command is as follows:-

- 1 -

You can arrange the legend you wish to appear in any combination, as long as they do not over-run the format commands for the page width, and page length.

The NEXT FILE command is used when you fill up the 151 available lines for one file. The command must be the last entry in the file and is in the following format 'CNTRL'nx: Next filename'RET' You can now link more than one file together, and when trying to output them, to either the screen or the printer, you must specify 'G' for global output, this will link together all the files in a specific sequence.

Conclusion

In conclusion I will say that the tips that I have explained, are the ones

Continued on Page 18

The Commodore 8010 Modem

Reproduced from Transactor - Gord Campbell, Toronto

Hurray! It works! In fact, the main benefit of the Commodore Modem (CBM 8010) is that it is so easy to use. You just take it out of the box, plug it in, attach an IEEE 488 cable, and you have completed the installation.

But what does it do? It allows the PET or CBM to communicate with other 'similar' devices. Thus you can communicate with not only another PET with a modem, but also with mainframe computers or a wide variety of teminals. The only restriction is that the other end must also run at 300 baud (30 characters per second). Most terminals these days support 300 baud: the major exception is the good OLD Teletype.

To establish communications, one end phones the other, agreement is made on who will run in answer mode, and both ends place their telephone handset in the cups in the top of the modem. The modems sing away at each other, sending and receiving characters.

Like any device on the PET, it doesn't do anything without a program, but that isn't a big deal. In the 11 page manual are listings of programs to do PET-to-PET or PET-to-mainframe communications, as well as a program which helps to pin down any problems which might occur. Unfortunately, at least one of these has a bug (take line 240 out of the terminal program). Since many of the people who might like to use the PET as a terminal have no interest in programming, it would have been nice to have a tape right in the box. Presumably any dealer who is selling modems has working versions of the programs.

The most difficult concept to grasp in programming for the modem is that the program must handle events which are coming from an external source, possibly quite quickly. For example, a mainframe computer will send characters to you at the rate of 30 per second. If you fall behind in processing them, you miss a few. A BASIC program to accept characters from the modem and write them to disk must be structured for speed or it will lose some. This will be less of a problem if the other end is producing characters at someone's typing speed. Any extra overhead which slows down programs (such as DOS support) will make the problem worse of course.

The commands used in programming the modem are the same as for other devices (OPEN, CLOSE, PRINT#, GET#, and INPUT#). Thus it should be simple to set up two-player games by phone. One of my desires is a program to play bridge by phone: me and the PET versus a friend and his computer. The computers would get to do all the dull stuff like deal, keep score, and be dummy. The only clever things they would have to do is bid and play defense. The easy part of such a program would be the part handling the modem.

The price of the modems does seem a little steep but this is a professional, well-intergrated peripheral. The owners of the other two major personal computer systems have to pay about the same amount to obtain the officially-approved equivalent capability. The only reason that it seems so much is that there are other ways to get communications without paying as much. Unfortunately, the other ways are not nearly so easy to use.

Are there any complaints? A couple of minor ones, but no big deal. The Commodore modem does not allow sending a true 'break'. Many large computers accept a break (which is not a character as such) to stop them from doing what they are doing. For example, if you are editing a large file using IBM's TSO (time sharing option), and you accidentally say LIST, you will have to wait a long time before you can enter another command. (This can be overcome by telling TSO, 1. don't send more than say 20 lines without giving me a shot, and 2. if I send you a certain character string, take that to be a 'break').

Finally, the main thing which makes the modem so easy to use (the IEEE bus), can be a drawback at times. For example, it is very difficult to use the modem and the printer at the same time. This is because the printer will not accept the first character of a line until it has finished printing the previous line. By the time your program continues, you have lost characters. This could have been avoided by giving the modem a fair sized buffer, but that would have made the price even higher.

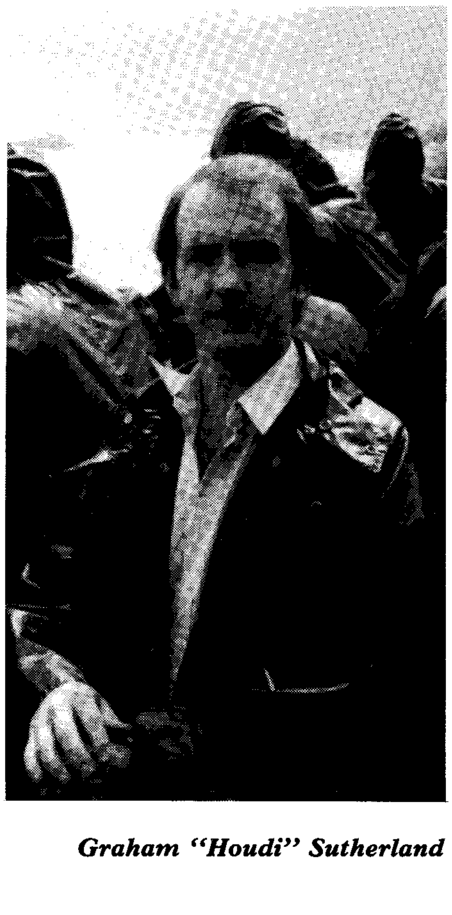
In summary, if you want to add data communications to the PET, the Commodore modem deserves consideration. For hassle-free installation and operation, it is probably your best choice!

Continued from page 17

Word Pro 4 tips - Graham Sutherland

that have been of most help to those people who I have shown how to use some of the functions of Wordpro, and are the type of things that can often only be hinted at in a manual. I hope that by reading this article you will not go through some of the trauma, that I as a self-taught user, have been through learning some of these lessons.

This article was induced on Wordpro 4, and any spelling mistakes are purley occidental.



Graham "Houdi" Sutherland

Basic Programming

It's not feasible to use disks with any damaged blocks with the CBM system. Give them to APPLE users.

PETs **internal firmware** can talk with 12 distinct (4-15) external IEEE addresses. This does not necessarily mean 12 items can be connected to the IEEE bus simultaneously, that depends on the driving ability of the TALK circuits. More than 1 PET can be connected to the same IEEE bus, with no special precautions, provided only 1 PET uses the bus at one time, & they're BASIC4 PETs (Not checked, I have a modified screen scroll routine in my $E000 ROM instead).

OVERWRITE in WordPro II & REPLACE in WordPro III actually do SAVE with REPLACE. You have to SCRATCH the old file before you record the new version if you need to OVERWRITE - hazardous.

Disk Drive Tips

If the floppy mis-behaves on cold days, the mains voltage is probably low. Fit an autotransfer, or, much better, a Constant Voltage Transfomer.

Providing you Initialize every time you insert a disk, especially on a 4040, a duplicate ID will never bother the system, it will merely confuse you.

The drives used by 2040/3040/4040 do not centre a disk in it's packet before clamping the hub. On a small percentage of occasions, it will be eccentric. Either centre the disk by eye, pushing the centre of the disk with your fingers, Or, don't close the drive door on a stationary disk. In the writer's experience, this is 100% effective.

Positioning for DATA READs

Everyone knows how RESTORE, READ and DATA statements operate. The first READ gets the first DATA element, and so on. RESTORE sets the READ pointer back to the beginning of text. But there is no command that allows positioning to a particular DATA line.

This could be useful if, for example, a DATA line were part of a subroutine. The only way to accomplish this in strict BASIC is to RESTORE and then issue enough READ commands to position to desired data. This can be a pain!

RUN, CLR or RESTORE sets the DATA Read Pointer (address 62 and 63 decimal) back to $0400; the start of BASIC text. When a READ command is given, this pointer starts advancing through text looking for a 'DATA' line. If the pointer reaches the end of text before finding data, an ?OUT OF DATA ERROR occurs.

PET maintains another pointer that climbs up and down through text. This pointer is part of the CHRGET routine and essentially points at the code currently being executed. If this pointer (addresses 119 & 120) is transferred into the DATA Read Pointer, the next READ command would force a search to the DATA line.

```
    10 DATA FIRST, SECOND,
THRID
    20 DATA FOURTH
    30 READ A$, B$
    40 POKE 62, PEEK(119) :
POKE 63, PEEK(120)
    50 READ A,B
    60 PRINT A$, B$, A, B
    70DATA 1,2,3,4
    80 END
```

The READ command in line 30 gets "FIRST" into A$ and "SECOND" into B$, leaving the pointer at "THIRD". Line 40 moves the pointer past line 10 and line 20 leaving it at some point in line 40. Since this is obviously not a DATA statement, the next READ causes an advance to line 70.

In summary, the POKEs of line 40 position to the **next** DATA statement in text.

Counting to the Screen

In volume three, issue four, there was an article by Mike Gross-Niklaus comparing a BASIC program to a machine code one. The programs were concerned with counting up to a million, and displaying the result on the screen. Well, it had to happen and it has. Mike's BASIC program, being directly comparable with the machine code program, was not terribly efficiently written, as Mike quite happily agreed. Consequently his statement that the time taken to count up to one million was over seven hours was for that program, and he cheerfully conceded that it would be possible to produce a program that did the count far more quickly. Dr. Porter, of the Glasgow Institute of Radiotherapeutics and Onclogy (ed. what IS oncology ?), has done just that, and the program is listed below.

This takes 1 hour 15 minutes and 13 seconds - the fight is on! I can't help feeling Dr. Porter is "cheating" somewhat, as I'm sure someone would argue that this isn't in fact counting (a bit esoteric I know), so I'll leave the debate to the next mailbag!

```
50  S5=32773:J5=J4:S4=S5-1:S3=S5-2:J3=J2:S2=S5-3:S1=S5-4:J1=J0:S0=S5-5
100  T=TI
110  FORJ0=48TO57:POKES0,J0:FORJ1=48TO57:POKES1,J1:FORJ2=48TO57:POKES2,J2
120  FORJ3=48TO57:POKES3,J3:FORJ4=48TO57:POKES4,J4:FORJ5=48TO57:POKES5,J5:NEXT
140  NEXT:NEXT:NEXT:NEXT:NEXT:T=TI-T
180  T=INT(T/60+0.5):J0=INT(T/3600):T=T-3600*J0:J1=INT(T/60):T=T-60*J1
190  PRINT J0;"HR";J1;"MIN";T;"SEC"
```

POKE OPEN files

Reproduced from Transactor

If a file is OPENed from a program and a ?SYNTAX ERROR occurs before the program gets a chance to CLOSE it, you can still CLOSE it from the keyboard using a direct command (i.e. CLOSE lf). But if you edit the program before CLOSing, you'll find this is no longer possible.

Editing essentially does a CLR which also aborts all file activity. However, nothing is sent to the disk to close it's open files. If these were 'read' files, send an Initialize or Catalog command and the disk is back to normal.

OPEN 'write' files are a different story. If they aren't CLOSEd properly, a number of nasty conditions can occur that are no fault of the DOS.

All disk files (PRG, SEQ, REL or USR) are created by recording data in sectors around the disk and then linking them together using two bytes to store the track and sector co-ordinates of the **next** sector in the chain. The first link is stored in the directory as soon as the file is OPENed for write. As successive sectors are written, successive links are also recorded (first two bytes of sector). Thus the directory points at the first sector, the first sector points at the second, and so on.

When the DOS closes a write file, and 'end marker' is recorded in the last sector. This is characterized by setting the track byte to zero (since there is no track 0), and pointing the sector byte at the last character recorded. If the file is not closed, this end marker never gets stored. An "open chain" condition results.

Blocks that haven't been used contain all zeros. The previous link would point at two zeros which is ok. However, blocks that have been used and then freed due to some earlier scratch operation, still contain old links. If an open chain points at one of these, a "phantom chain" is created. This can be deadly!

Consider a Scratch operation on an improperly closed file. The DOS would follow the chain, releasing sectors to the BAM (Block Availability) right up to the point where the write was aborted. But if a phantom chain exists, the DOS continues along the chain releasing more sectors which might lead right into a good file! In the next write operation, the DOS might select one of these sectors and clobber live data (BLECCH!).

Fortunately we have the Validate or Collect command. A Collect will discard improperly closed files, freeing all sectors that **do not** belong to a chain and allocating all sectors that do. Collect also frees any allocated direct access blocks, which may not be too desirable. Besides that, an 8050 Collect operation can take awfully long if your directory contains many entries. Wouldn't it be nice if we could just CLOSE that pesky open file, scratch it, and start all over?!

Well,... you can! As long as the disk is not disturbed by a Catalog, reset, etc. When the PET aborts external file activity due to editing, a CLR, etc., it merely sets the number of OPEN files to zero (address 174). All file parameter tables are left in tact (addresses 593 through 622). These files can be ressurected as long as no new files have been OPENed.

POKE 174, X

... where 'X' is the number of OPEN files. If you don't know how many files were OPEN, simply POKE 174,10. This sets it to maximum at which point even some properly CLOSEd files would now be OPEN again! Now you can issue any necessary CLOSE commands directly from the keyboard and exterminate those disk gremlins that show up as an asterisk beside the file type.

Although this will get you out of a jam, it is not suggested practice! CLOSing disk files properly will reduce knashing-of-teeth and pulling-of-hair considerably! (right Jim?)

APFEL BASIC

By David Simons

Although the PET and the Apple have many features in common, there are some features available on only one of the two machines. Hi and Low res plotting is one example but the Apple does have such things as POP which clears the last GOSUB off the stack. In this enhancement, which adds over 30 new operations, I have used the plot routine and jnoin routine from CPUCN Vol 3 issue 2. The utility is for BASIC 2 (Basic 4 on request from the address at the end of the article) and has been written using the Commodore assembler, it takes about 2K. The version below is located so that it will work in either 32k or 16k machines though you may change the start address so that it is in the top of the memory.

As well as adding commands to the PET it also adds new key meanings, which are described below. To use the command keys press RUN STOP twice and then the desired key.

Q - Quits Quote Mode.
W - Delete line to cursor
E - Delete line from cursor.
R - Delete line
T - Go to next set tab.
Y - Set current cursor column as a tab stop.
U - Clears all set tabs.

The following operations are BASIC commands and can be used quite normally, though if they are used after a 'THEN' you should place a colon before the command, e.g.
10 IFA=0THEN:%A 10,10TO20,20
The commands added are as follows; if there is a similar Apple command it is shown in brackets.

%A - Sets line in Hi res.(HPLOT)
%B - Clears line in Hi res.
%C - Sets a square in Hi res.(HPLOT)
%D - Clears a swuare in Hi res.
%E - Sets vertical line.(VLIN)
%F - Clears vertical line
%G - Sets horizontal line.(HLIN)
%H - Clears horizontal line.
%I - Sets inverter.
%J - Horizontal tab.(HTAB)
%K - Vertical tab.(VTAB)
%L - Enables RUN STOP.
%M - Disables RUN STOP
%N - Plots character in low res.(PLOT)
%O - Kills last gosub.(POP)
%P - Sets a shape table.
%R - Clears a drawn shape in Hi res.(DRAW)
%Q - Draws a shape in Hi res.(DRAW)
%S - Sets the rotation value for shapes.(ROT)
%T - Sets limit of basic memory.(HIMEM)
%U - Enables key commands after LOAD.
%V - Prints characters and makes them flash.(FLASH)

%W - Sets speed of flashing characters.

%X - Stops any flashing characters.(NORMAL)

%Y - Kills Apfel basic.

%Z - Fine Hi res plot.

Syntax Of The Commands.

Those that are not listed below you can use without parameters.

%A and %B

Should be followed with the X coordinate, a comma, and the Y. Then the word 'TO' should be placed and the finishing coordinates. E.g. :-
%A 15,10TO40,40 ;will draw a line from 15,10 to 40,40.

%C and %D

Should be followed with the x coordinate, a comma, and the Y. E,g, :-
%C 10,15 ; will draw a dot at 10,15

%E and %F

Should be followed with the column,. a comma, the start of the line, the word 'TO' and the finish of the line. E.g. :-
%E 10,1TO20 ; will draw a line in the column 10 from 1 to 20

%G and %H

Should be followed with the row, a comma, the start of the line, the word 'TO' and the finish of the line. E.g. :-
%G 15, 1TO 50 ; will draw at row 15 a line from 1 to 50

%I (WITH CARE !!)

Should be followed with the start of the screen area to be inversed, a comma, the finish, a comma, and the speed of inversing (from 1-130). This command will not work at the same time as the %V command. Your finish address must be greater than your start and both addresses must be on the screen.

%K

Followed by the distance down from home that you require.

%J

Followed by the distance across. This may seem like the normal 'TAB('command but %K is not used in 'PRINT' statements.

%N

Followed, by the Y coordinate,a comma, the X coordinate, a comma, and the character.

%O

No parameter but if there is no 'GOSUB' it will return with an error.

%P

With this command you may find that it is easier using graph paper for planning shapes. The 'P' is to be followed by a number and then a string up to 254 characters long, this string must only contains numbers from 0-9, which mean :-

0 - Move right by 1.
1 - Move up by 1.
2 - Move down by 1.
3 - Move left by 1.
4 - Do no use !!
5 - Move right by 1 and plot.
6 - Move up by 1 and plot
7 - Move down by 1 and plot.
8 - Move left by 1 and plot
9 - Stop reading numbers in.

The number before the string is the shape table number, start at one and go up in steps of 1. The shape tables are not protected from BASIC but can be by following method.

1. Take 255 multiplied by the number of shape tables you require protected from the decimal of the Hi byte of the start of Apple basic mulitiplied by 255.
2. Place the number obtained after the %T command.

For example, with the start location as used in the assembled listing presented, assuming there are 4 shapes :-
A=(255*29)-(255*4):%T A
You can save the tables by using the monitor.

%R and %Q

These must be followed by the shape required, a comma, the start coordinate (x), a comma, and the y coordinate. By using the command in the correct way good animation can be created.

%S

This allows you to decide on the rotation value for the %R and %Q commands. The value of the rotation goes up by 45. I.e. a value of 8 gives 45 deg and a value of 16 gives 90 deg.

%T

Must be followed by a number, which is greater than 1030. After using this command should you get an '?OUT OF MEMORY' type error it means you have not given yourself enough memory and should use a larger number.

%V

Great care must be exercised when using this command. It must be followed by a string, do not use cursor up,home,clear movements if you want to be safe. You may only have 1 item flashing at a time.

%W

Just follow with a number 0 and 255

%Z

The first number must be the Y coordinate, a comma, the x coordinate, a 'TO', the finish y coordinate, another comma, and the finish x coordinate. The grid it uses is 255*25, but there can only be 1 mark in a square, so in some cases the results appear to be strange.

Typing in the program

You can use the Commodore assembler, but if you do not own this you may find yourself in trouble, so the following items are available from me, at the address below, on cassette. 16K/32K (40 column BASIC 2 or BASIC 4.0 machines - state rom & memory) MC loader programs at £8.50. This includes demo programs on the other side; these are demo, tutorial, and a plotter program (draws different types of charts). A version with DOS SUPPORT for BASIC 2 is available at £8.50 (includes demos), or for £13.00 you can have a version with SUPER BASIC (BASIC 2 only) and demos. Enable the version given with SYS 8349, after a LOAD use the %U command.
David Simons,
19 Reddings,
Welwyn Garden City.
AL8 7LA
Alternatively, if you send a large S.A.E. to C.B.M. we'll send you a copy of the listing (all 21 pages of it!)

Printer Hints

Thanks to the IPUG South-East regional group, and Mark Wardley in particular for this useful discovery. It concerns Basic 4 and outputting to the printer. Whereas on Basic 2 machines onc could quite happily type OPEN4,4CMD4:SYS4, on Basic 4 it doesn't work! You have to SYS54935 instead, the reason being that as well as moving the TIM monitor to a different location, there was some extra code added at the beginning that turned off all IEEE devices, so that a call to the beginning of the monitor (which is what SYS4 effectively does) now does not work as before.

As a footnote, I've tried this on an 8032, where it works in the following way :- on typing in, say, m 033a 0400 and hitting RETURN (after the OPEN... of course), the printer acknowledges my presence, and the PET screen displays M 033A 0400 in the top right hand corner of the screen. If I then cursor up and hit RETURN over the original m 033a 0400 the printer then happily prints the required section of memory.

Some Commodore Disk Utilities

Reproduced from Transactor

The following routines are for use with 4040 and 8050 disk units.

The first two are for reading the disk ID from the specified drive DR, device DV. Line 150 sets the drive. Line 160 sets the track. Line 170 tells the disk to read any header on track 18. Initializing not necessary! 180, 190 and 200 wait for the DOS to finish the read. Line 210 does any error processing (i.e. read error or no disk in drive). The first character of the ID is read from DOS memory (line 220) and the DOS puts it in the command channel. Line 250 reads the second character and both characters are put in 'IDS'.

This routine is particularly useful (and fast) to see if the user has placed a disk in the drive. It can also be used to detect insertion of incorrect diskettes. The software would have to anticipate ID numbers, perhaps ID's that were selected by the program in an earlier formatting operation.

The second two will return the BLOCKS FREE count BF, from the specified drive DR, device DV. BF is reset before entering. Initialize is necessary here! The block free count is not stored on the disk but rather calculated from the Block Availibility Map. Line 170 sends the DOS off to that routine in disk ROM! The result is placed in disk RAM where it is read, once again, into the command channel by lines 180 and 210. A few calculations and presto! Block Free!

Knowing blocks-free from within a program can be especially useful for anticipating the nasty DISK FULL error.

Versions for both 4040 (DOS 2.0) and 8050 (DOS 2.5) have been provided.

```
100 REM ID READER FOR 4040
110 ID$="" : REM RESET ID$
120 DR=0   : REM DRIVE #
130 DV=8   : REM DEVICE#
140 OPEN15,DV,15 : REM UNLESS ALREADY OPEN
150 PRINT#15,"M-W"CHR$(18) CHR$(0) CHR$(1)CHR$(DR)
160 PRINT#15,"M-W"CHR$(43) CHR$(16)CHR$(1)CHR$(18)
170 PRINT#15,"M-W"CHR$(4)  CHR$(16)CHR$(1)CHR$(176+DR)
180 PRINT#15,"M-R"CHR$(4)  CHR$(16)
190 GET#15,X$
200 IF ASC(X$)>127 THEN 180
210 IF ASC(X$)<>1 THEN PRINT#15,"M-E"CHR$(37)CHR$(217):PRINTDS,DS$:END
220 PRINT#15,"M-R"CHR$(41)CHR$(16)
230 GET#15,A$
240 ID$=A$
250 PRINT#15,"M-R"CHR$(42)CHR$(16)
260 GET#15,A$
270 ID$=ID$+A$
280 PRINTID$
```

```
100 REM ID READER FOR 8050
110 ID$=""          : REM RESET ID$
120 DR=0            : REM DRIVE #
130 DV=8            : REM DEVICE#
140 OPEN15,DV,15 : REM UNLESS ALREADY OPEN
150 PRINT#15,"M-W"CHR$(18) CHR$(0) CHR$(1)CHR$(DR)
160 PRINT#15,"M-W"CHR$(43) CHR$(16)CHR$(1)CHR$(18)
170 PRINT#15,"M-W"CHR$(4)  CHR$(16)CHR$(1)CHR$(176+DR)
180 PRINT#15,"M-R"CHR$(4)  CHR$(16)
190 GET#15,X$
200 IF ASC(X$)>127 THEN 180
210 IF ASC(X$)<>1 THEN PRINT#15,"M-E"CHR$(179)CHR$(238):PRINTDS,DS$:END
220 PRINT#15,"M-R"CHR$(41)CHR$(16)
230 GET#15,A$
240 ID$=A$
250 PRINT#15,"M-R"CHR$(42)CHR$(16)
260 GET#15,A$
270 ID$=ID$+A$
280 PRINTID$
```

```
100 REM SUBROUT. RETURNS BLOCKS-FREE FOR DOS 2.0
110 DR=0 : REM DR = DRIVE #
120 DV=8 : REM DV = DEVICE#
130 BF=0 : REM RESET BLOCK FREE COUNT
140 OPEN15,DV,15          : REM UNLESS ALREADY OPEN
150 PRINT#15,"I"+STR$(DR): REM UNLESS ALREADY INIT'D
160 PRINT#15,"M-W"CHR$(18)CHR$(0)CHR$(1)CHR$(DR)
170 PRINT#15,"M-E"CHR$(52)CHR$(219)
180 PRINT#15,"M-R"CHR$(119)CHR$(67)
190 GET#15,A$
200 BF=ASC(A$+CHR$(0)) : REM IN CASE A$=""
210 PRINT#15,"M-R"CHR$(120)CHR$(67)
220 GET#15,A$
230 BF=BF+ASC(A$+CHR$(0))*256
240 PRINT BF
```

```
100 REM SUBROUT. RETURNS BLOCKS-FREE FOR DOS 2.5 (8050)
110 DR=0 : REM DR = DRIVE #
120 DV=9 : REM DV = DEVICE#
130 BF=0 : REM RESET BLOCK FREE COUNT
140 OPEN15,DV,15          : REM UNLESS ALREADY OPEN
150 REMPRINT#15,"IO"
160 PRINT#15,"M-W"CHR$(18)CHR$(0)CHR$(1)CHR$(DR)
170 PRINT#15,"M-E"CHR$(231)CHR$(211)
180 PRINT#15,"M-R"CHR$(158)CHR$(67)
190 GET#15,A$
200 BF=ASC(A$+CHR$(0)) : REM IN CASE A$=""
210 PRINT#15,"M-R"CHR$(160)CHR$(67)
220 GET#15,A$
230 BF=BF+ASC(A$+CHR$(0))*256
240 PRINT BF
```

Spooling Disk Files to Printers

Reproduced from Transactor

In Compute #8 , T.M. Peterson published a neat trick for getting the 2040 disk to talk to a printer without PET/CBM supervision of the IEEE bus. I imagine this would work for

4040s, 8050s and any make printer interfaced via the IEEE bus, but naturally I can't be sure for all cases. However, the idea was so incredible that I felt it definitely worth repeating.

Everyone knows how to LIST a program to the printer. But long listings can wear patience thin, especially on a slow printer! Not only that, but while your printer is chugging along, the PET just sits there with everything disabled except RUN/STOP. Wouldn't it be nice if the disk fed the printer while you continue editing or play a quick round of space invaders or Microchess, that is if you can bear some of those arrogant printers. By the way, those wing nuts on the bottom of Commodore 202X printers... take them out. They're only shipping screws that hold the mechanism tight. Once removed, the noise level is reduced considerably.

First a file must be created on disk. This could be any SEQ file with any contents that are printer recognizable, but for now we'll create one of a program LISTing.

```
1.  Enter some small program
2.  OPEN 8,8,8,"0:TEST SPOOL,S,W" : CMD8 : LIST
3.  PRINT#8,""; : CLOSE8
4.  NEW
5.  OPEN 8,8,8,"0:TEST SPOOL"       ;defaults to ',s,r'
6.  POKE 165,72  : SYS 61695        ;use SYS 61668
7.  POKE 165,104 : SYS 61695          for BASIC 2
8.  OPEN 4,4 : CMD4 : POKE 176,3 : POKE 174,0
```

On hitting return the printer should fire up and continue at full speed to the end of the file. At this point your cursor might be acting funny (try hitting return on a blank line). To stop this, POKE 14,0 for BASIC 2.0 or POKE 16,0 for BASIC 4.0. If you're lazy like me, invoking a couple of ?SYNTAX ERRORS (eg. '=' and Return) will do the same thing. However, to restore normal cursor operation under program control (yes program control!), you would have to use the POKE. More on this in upcoming paragraphs.

Once the printer starts, don't try using the IEEE bus or the spool will abort. When its all finished you can CLOSE the open disk file by sending an Initialize command or with:

OPEN 1,8,8 : CLOSE 1

A filename isn't necessary, but use the same secondary address as in step 5.

The NEW command at step 4 is only for clarity. Instead you might load another program or just leave the current one in for further editing. You can RUN the program in memory and even use the cassettes, but they're still as slow as before.

The example here uses all direct commands but they could just as easily be put in a program. Think of the applications! In a user oriented system, a report could be output to the disk and immediately spooled to the printer while the operator continues working on the next task. Of course the user might inadvertently try a bus operation which would kill everything. Fortunately this busy state can be detected using the following "trap":

```
100 IF (NOT(PEEK(59456))) AND 64 THEN 100
110 OPEN 1,8,SA : CLOSE 1
130 ...and continue
```

If a spool is in progress, line 100 will loop back to itself until the bus is free. Line 110 is for closing the disk files and also turns off the active LED. SA is a variable containing the secondary address which might be used in coding the OPEN command that starts the operation. Also not that line 110 causes no disk activity so there's no need to go around it to save time.

Theory and Variations

This example used device number 8 right through. However, you might have more than one disk on line which would mean a different device number. In step 6, address 165 (the IEEE output buffer) is POKEd with 72. This number (72) is derived from 64 + 8, where 8 is the device number. For versatility, this '8' might be replaced by a variable like DV.

The following SYS activates the ATN line on the bus telling all devices to 'pay attention'. The contents of 165 are then sent to the bus but only the device that has a matching 'talk' address responds, in this case device 8; the disk.

The disk is ready to start sending but from where? All it needs now is the secondary address of the OPENed file. Step 7 sets up the output buffer with the secondary address plus 96. Since 8 was chosen, the result is 104. This step might also be modified to read POKE 165,96+SA.

Nothing happens yet because the ATN line isn't released by the PET. When CMD4 is executed (step 8), the printer becomes the output command device. PET releases ATN, the disk starts talking and the printer listens.

POKE 176,3 tricks the PET into thinking the output command device is the screen and POKE 174,0 simulates no files OPEN. In a program you would have to re-open files (eg. command channel, modem, etc.) at spool completion. By the same token, you might want to CLOSE any open write files before starting.

Should anyone discover any useful variations to this technique, let us know. We'll be glad to here about it!

Disk Use for Beginners

David J. Pocock

Once more unto the breach dear friends!! (What is this man talking about ?)

This month's article will be shorter than normal due to organisation for the PET Show, a little something we put together last month. (You were there, weren't you ?). Nevertheless, as promised last time I will cover Scratch, Verify, Rename and abbreviations for the commands.

Scratch

This means delete a file from the disk. To remove an old version of a program that you are 100% sure you are finsihed with (there's no way to get it back), the command is :-
"SX:program"
where X is the drive number. This is to be typed in after the usual OPEN1, 8,15 and needs to be preceded by the PRINT#1, statement. What this does is to remove the program or data file from the directory, and free up the space on the disk for other use. A program (or file) written to the disk now will appear at the same point in the directory as the old file. More about why this happens when I go into the directory in more detail at a later date. If you read the error message (the sample program in last issue, or if you're one of the BASIC 4.0 users just PRINTDS$ will produce the goods) just after a scratch you should get a message of :-
01,FILES SCRATCHED, XX,00
where XX is the number of files scratched. As you will see later that it is possible to scratch more than one file at a time.

Verify

Also called Validate. The command is :-
"VX"
where X is the drive number, and again this is to be used after the usual OPEN etc. command. What this command does is reconstruct the BAM on the disk. The BAM, or Block Availability Map, tells the disk unit how much of a diskette has been used and where the empty space is (if any). The command assumes that the disk is empty, and it then goes through the directory reading through every file to find out which sections of the disk have been used. The error message at the end should be :-
00,OK,00,00

If it is not, this indicates that one section of the disk has become corrupt and cannot be read. If this happens the BAM is not re-written and the original BAM is left on the disk. The best thing you can do at this stage is to format a new disk and copy each file across seperately. There is a program for this on the Utility Disk, and it is called COPY DISK FILES.

Rename

This changes the name of a file in the directory, and the command is :-
"RX:new name=old name"
where X is the drive number, and new name and old name are pretty self explanatory. (There's not really a lot to say about rename is there ?).

Short forms

For the moment I will only deal with the short forms of file names. Those of you familiar with cassettes will know that if you say LOAD"PAY" the PET will load PAY, PAYROLL, PAYOUT or whatever it finds first. In other words, anything that begins with PAY. With disk you have two options for abbreviations which are :-
* replaces any number of characters at the end of a file name.
? replaces any single character anywhere in the file name.
e.g. LOAD"PAY*",8 has the same effect as LOAD"PAY" for cassette. or LOAD"TEST ? PROG",8 would load TEST 1 PROG, TEST A PROG etc., whichever appears first in the directory.

WARNING

When used in conjunction with Scratch abbreviations can be disastrous, as it would mean scratching everything that matched the pattern. e.g. "S0:PAY*" would scratch ALL files on drive 0 which begin with letters PAY. Be careful when using short forms with scratch.

See you next time!

Utilities & Programming Examples

A collection of utilities, mainly designed for Basic 2 machines only at the moment, that you can use and perhaps add to your own programs for the increased efficiency that machine code brings. These programs work - as you see them is as they were generated from the internal monitor, and not typed in listings that had to be checked, double checked, typeset and checked again, with the inevitable errors creeping in.

The first is called "unnew/sys826", which as you might infer from the name is a program to

Example 1

```
?

.
.:  Ø33A A9 Ø4 85 2E 85 2F AØ ØØ
.:  Ø342 2Ø 6F Ø3 DØ FB 2Ø 6F Ø3
.:  Ø34A A5 2E 8D Ø1 Ø4 A5 2F 8D
.:  Ø352 Ø2 Ø4 A9 Ø1 85 2A A2 Ø3
.:  Ø35A 2Ø 6F Ø3 DØ F9 CA DØ F8
.:  Ø362 2Ø 6F Ø3 A2 Ø3 B5 2C 95
.:  Ø36A 2A CA 1Ø F9 6Ø E6 2E DØ
.:  Ø372 Ø2 E6 2F B1 2E 6Ø ØØ ØØ

.
```

help you recover from the horrors of typing in NEW, and then immediately wishing you hadn't. With this program resident in the machine (and no other that occupies the same area of memory), you simply type SYS826 and hey presto! Back comes the program again.

The second is "low case list", and again it's function is fairly obvious from the name. Again, type SYS826 to set it up.

The third is called "keyprint/826", and is used by loading it into the machine, typing SYS826, and then letting it sit their until required. Then, at any time whilst another program is running (only one that doesn't reside in the same area of memory, or locks up the keyboard), if you hit the '/' key, the program will stop it's operation and the contents of the screen will be printed out on your friendly Commodore printer. When finished, the program starts up again from where it left off.

The fourth one, a Basic program this time, but one that has many uses, as it is a form of 'Print Using', a function often desired in a program, but one which most people give up with after playing around with MID$, LEFT$ and so on for hours on end.

Example 2

```
?
.
.:   033A A9 00 85 11 85 12 20 2C
.:   0342 C5 68 68 A0 01 84 09 B1
.:   034A 5C F0 46 20 E1 FF 20 E2
.:   0352 C9 C8 B1 5C AA C8 B1 5C
.:   035A C9 FF D0 04 E0 FF F0 31
.:   0362 84 46 20 D9 DC A9 11 20
.:   036A 45 CA A9 20 A4 46 29 FF
.:   0372 20 C2 03 C9 22 D0 06 A5
.:   037A 09 49 FF 85 09 C8 F0 11
.:   0382 B1 5C D0 10 A8 B1 5C AA
.:   038A C8 B1 5C 86 5C 85 5D D0
.:   0392 B2 4C 89 C3 10 DA C9 FF
.:   039A F0 D6 24 09 30 D2 38 E9
.:   03A2 7F AA 84 46 A0 FF CA F0
.:   03AA 08 C8 B9 92 C0 10 FA 30
.:   03B2 F5 C8 B9 92 C0 30 05 20
.:   03BA 45 CA D0 F5 49 80 D0 AC
.:   03C2 48 09 C0 C9 DB 90 08 C9
.:   03CA E0 10 04 68 49 80 48 68
.:   03D2 4C 45 CA EA 53 44 00 00
.
.
```

Example 3

```
?
.
.:   033A 78 A9 03 85 91 A9 45 85
.:   0342 90 58 60 A5 97 C9 45 D0
.:   034A 03 20 51 03 4C 2E E6 A9
.:   0352 80 85 20 A9 00 85 1F A9
.:   035A 04 85 B0 85 D4 20 BA F0
.:   0362 20 2D F1 A9 19 85 22 A9
.:   036A 0D 85 21 20 D2 FF A0 11
.:   0372 AE 4C E8 E0 0C D0 02 A9
.:   037A 91 20 D2 FF A0 00 B1 1F
.:   0382 29 7F AA B1 1F 45 21 10
.:   038A 0B B1 1F 85 21 29 80 49
.:   0392 92 20 D2 FF 8A C9 20 B0
.:   039A 04 09 40 D0 0E C9 40 90
.:   03A2 0A C9 60 B0 04 09 80 D0
.:   03AA 02 49 C0 20 D2 FF C8 C0
.:   03B2 28 90 CB A5 1F 69 27 85
.:   03BA 1F 90 02 E6 20 C6 22 D0
.:   03C2 A6 A9 0D 20 D2 FF 4C CC
.:   03CA FF 00 00 00 00 00 00 00

.
.
READY.
```

Example 4

```
 100 REM DEMO PROGRAM FOR SUBROUTINE
 110 FORJ=1TO20
 115 REM V IS VALUE TO BE FORMATTED
 120 V=EXP(RND(1)*14-6)*SGN(RND(1)-.2)
 125 REM V1 IS # OF DIGITS LEFT OF .
 126 REM V2 IS # OF DECIMAL PLACES (RIGHT OF .)
 130 V1=4:V2=0:GOSUB50000:PRINTV$;" ";
 140 V1=3:V2=1:GOSUB50000:PRINTV$;" ";
 150 V1=3:V2=4:GOSUB50000:PRINTV$
 160 NEXTJ
 170 END
 50000 REM 'USING' ARRANGE IN COLUMNS
 50010 REM V IS VALUE; V1.V2 PRINTS
 50020 V4=INT(V*10^V2+.5)
 50030 V$=RIGHT$("          "+STR$(V4),V1+V2+1):Q$=V$
 50040 IF V2<1 GOTO50080
 50050 FORV5=V1+2TOV1+V2+1:IF ASC(MID$(V$,V5))<48THENNEXTV5
 50060 V6=V5-V1-1
 50070 V$=MID$(V$,V6,V1+1)+LEFT$(".00000",V6)+MID$(V$,V5)
 50080 IF ASC(V$)>47 THEN V$=LEFT$("**********",V1+V2+2+(V2=0))
 50090 RETURN
READY.
```

Control Statements in PET BASIC

1. BASIC is an unstructured language, which means that all statements and variables are accessible from parts of the program. PET BASIC, like most micro-based versions, is interpreted, which implies that when things happen is largely immaterial - it is the order in which they happen that is important. This article describes the control statements in BASIC, which enable the order of statement execution to be varied, possibly depending on values calculated during the course of running the program. The control statements in BASIC are IF-THEN, GOTO, GOSUB, ON-GOTO, ON-GOSUB, RETURN, FOR and NEXT.
2. The following conventions are adopted for providing a precise definition of the syntax, or form, of BASIC statements:-

(a) Anything in capital letters stands for itself.

(b) Anything in lower-case is the name of a class of objects, any valid example of which may replace the class-name.

(c) Anything enclosed in square brackets   optional, and mauy be omitted. If the thing in the brackets is followed by three dots, it may be repeated any number of times.

(d) If two or more things are bracketed together using curly brackets ( ), they are alternatives -one thing must appear.
For example, a train consisting of an engine, followed by at least one truck which may be a van or an open-truck, followed by a guard's van would be described as

$$\text{engine}\left\{\begin{array}{l}\text{van}\\ \text{open-truck}\end{array}\right\}\left[\left\{\begin{array}{l}\text{van}\\ \text{open-truck}\end{array}\right\}\right]\quad\text{guard's-van}$$

3. Before describing the IF-THEN statement, it is necessary to look at the relational expression. BASIC has only two types of variable - numeric and string. A relation, e.g. A > B, C = 2 is either TRUE or FALSE, depending on the values either side of the relational operator. In BASIC, TRUE is represented by the number -1, FALSE by the number 0: thus if A=B=C=2, A>B is an expression with a value 0, C = 2 is an expression with the value -1. The relations in BASIC are any combination of 1,2, or 3 of < = and > in any order (A< = >B means "A is less than, equal to or greater than B", and will always give the value -1). Numeric values are compared algebraically: string are compared character by character from left-hand end until a difference is found, and compared on the numeric (ASC) values of the differing characters. If the end of either string is reached, the shorter is treated as being less than the larger. Equality occurs only if both strings are the same length and identical character for character.
The numerical value of a relation may be used in normal arithmetic e.g.

```
100 REM TO DERIVE THE CHARACTER
  STRING CORRESPONDING
110 REM TO THE HEXADECIMAL
  (BASE 16) VALUE OF X
120 A$=" "
130 Y = INT(X/16):Z = X-16* Y: X=Y
140 A$ = CHR$(48+Z-7*(Z>9))+A$
150 IF X<> 0 THEN 130
```

The statements on line 130 place the value (0-15) of the next hexadecimal digit of X into Z.
If Z   = 9, then (Z   9) has the value 0, and the character 48 + X i.e. "0" -"9" will be added to A$. If Z   9, (Z   9) has the value -1, and the character 55+Z, i.e. "A" - "F", is added to A$. This process may be used generally for adjusting carries after non-decimal arithmetic e.g. adjusting an integer so that whenever the last two digits become 60, an extra 40 is added in to give the effect of "clock arithmetic".

4. IF-THEN statement. The syntax of this is

$$\text{IF expression}\left\{\begin{array}{l}\text{GOTO}\ \ \text{line-number}\\ \text{THEN}\left\{\begin{array}{l}\text{line-number}\\ \text{statement}\end{array}\right\}\end{array}\right\}$$

Note that the form GO   TO is not accepted, although the THEN statement form can contain GO   TO. Expression should give a numeric value, although it is not checked, and a string expression will not be signalled as an error - the results will be unpredictable! If expression is non-zero, the remainder of the statement is obeyed: if zero, BASIC skips to the end of the current **line,** ignoring the rest of the statement and any other statement on the same line. A line number is an interger in the range 1 - 63999.
The effect of the IF-THEN form is to obey, or ignore, all the statements on the remainder of the line containing the IF-THEN statement, depending on the condition, e.g. to arrange two consecutive elements of an array in ascending order, the following forms are equivalent:-

```
100 IF A(I)<= A(I+1) THEN 140 : REM
  ALREADY IN ORDER
110 X = A(I) : REM OUT OF ORDER
  -INTERCHANGE.
120 A(I) = A(I+1)
130 A(I+1) = X
140
```

OR

```
100 IF A(I)>A(I+1) THEN X=A(I): A(I) =
  A(I+1):A(I+1)=X
```

The only limit on the number of statements following THEN is the limitation imposed by BASIC Screen Editor that a line, incliuding its line number, may not contain more than 80 characters.
In the THEN and GOTO forms, BASIC searches the stored program for a line with the specified number, as for the unconditional GOTO statement. The statement form after THEN may be any executable BASIC statement, including control statements.

5. The GOTO statement. The syntax is:

$$\left\{\begin{array}{l}\text{GOTO}\\ \text{GO}\ \ \text{TO}\end{array}\right\}\ \text{line-number}$$

The BASIC interpreter skips to the end of the line containing the GOTO and compares the number of the next line with the line number which is the destination of the GOTO. If (GOTO line-number/256) is greater than (next line-number/256), BASIC searches forwards from the current position until it finds the GOTO line number, or that it is missing. If the comparison is not satisfied, BASIC searches from the beginning of the program e.g.

```
240 GOTO 260
250 REM (line number/256) =0
260 REM (target line-number/256) =
  1-forward search whearas
240 REM (line-number/256)=0. (target line-
number/256)=0,
   search from beginning of program.
```

6. The ON-GOTO statement. the syntax is

```
     On numeric-expression
GOTO line-number   , line-number...
```

This statement gives the facility to GOTO one of a number of line-numbers, depending on the value of a numeric expression. Note that the G TO form is not permitted. The numeric expression is converted to an integer, any decimal fraction part being ignored. The integer must be in the range 0-255 inclusive, otherwise an error "ILLEGAL VALUE" is displayed. The list of line-numbers is searched until either the corresponding line number is found or a line-number terminated by end-of-line or colon has been passed, indicating end of the list of line-numbers. If the corresponding line-number is found, BASIC performs a GOTO to that line-number. If not BASIC ignores the rest of the current statement and continues with the next statement.
Note that the syntax of the statement is checked only as far as the required line number, e.g.

```
100 ON I GOTO 200,300.400, 500
```

will work correctly for I = 1 or 2, and only give a SYNTAX ERROR if I is 3 or more.
The limit on the number of line-numbers which can follow an ON-GOTO is the maximum number of characters which can be inserted in one line, although not more than 255 such line-numbers can be made use of.

7. The FOR and NEXT statements. These are mainly intended for the construction of sections of program to be repeated a number of times. They are, however, separate statements, and do not act like the statements of similar appearance in other languages. FOR and NEXT statements communicate by means of a **stack,** which is a variable-length area of a computer store with the property that information can be stored and retrieved only from the free end ("last in, first out"). If the maximum space reserved for this stack is exceeded, an error "FORMULA TOO COMPLEX" is displayed.
Syntax of the FOR statment is FOR indentifier = numeric expression-1T0 numeric-numeric-expression-2(STEP numeric-expressior-3) identifier must be a simple variable (i.e. not an array element) of type numeric. BASIC calculates the value of the three expressions - if the STEP option is not used, the value of the third expression is taken as 1.0 - and assigns the value of the first to identifier. Identifier can appear in the expressions: its old value will be used. BASIC constructs a packet of informaiton about the FOR statement and saves it on the stack i.e.

Number of line containing the FOR
Position in the line of the end of the FOR
 statement (: or End of line)
Value of the TO expression
SIGN (-1/0/+1) of the STEP expression
Value fo the STEP expression
Identifier
'FOR' marker

BASIC then continues to execute the current line of program, i.e. the statements following the FOR statement will be executed at least once regardless of the initial, TO and STEP values. If a FOR packet referring to the same **identifier** is already on the stack since the last GOSUB, everything on the stack back to and includinng the pre-existing FOR packet is deleted before the new FOR packet is added, i.e. a FOR statement will automatically close any "open" FOR statement using the same identifier, together with any later FOR statements still open e.g.

```
100 FOR I = 1 TO 10
110 FOR ~~J = 1~~ TO 10
120 IF A(I,J)= 0 THEN 160
130 NEXT J
140 NEXT I
150 STOP
160 FOR I = 1 TO 10
170 REM THE I & J LOOPS IN LINES
  100-140 will be closed.
```

Note that until its identifier is reused in another FOR statement, or it is closed by a NEXT statement, the FOR packet will remain "open " on the stack.
Syntax of the NEXT statement is

    NEXT identifier ,identifier...

e.g.

```
NEXT
NEXT J,
NEXT J,I
```

When a NEXT statement is encountered, BASIC searches back along the stack from the free end looking for a FOR packet which refers to the identifier (if no identifier is specified, it will accept the first packet regardless of identifier). Any FOR packets for other identifiers found on the way will be deleted, e.g.

```
100 FOR I = 1 TO 10
110 FOR J = 1 TO 10
120 IF A(I,J) = 0 THEN PRINT I,J:GOTO
  140
130 NEXT J : REM J PACKET REMOVED
  WHEN J  10
140 NEXT I : REM WILL REMOVE J
  PACKET IF STILL OPEN.
```

BASIC then finds the variable referred to in the FOR packet, and adds the value of STEP expression to it. It calculates SIGN (new value of variable - TO value), and compares this with SIGN (STEP value). If they are the same, the FOR packet is deleted form the stack, and BASIC continues with the remainder of the NEXT statement if any, or the next statement. Otherwise BASIC recovers the location of the statement following the FOR statement, and continues with that. Thus:

(a) If STEP is non-zero, NEXT ends the FOR loop when the first value of the FOR variable **past** the TO value is claculated, e.g.
FOR I = 1 to 10
will repeat for I = 10 and finish when adding 1 to I makes the value exceed 10, i.e. I = 11.

(b) The FOR variable does not have to be an integer value, but care is necessary in this case e.g.
FOR I = 1.1 TO 10.1

Since BASIC cannot store 1.1 exactly, the calculated value of I on the ninth arrival at the corresponding NEXT statement will be **approximatley** 10.1 : it may be just less than, equal to, or slightly greater than the TO value. The safest way is to write
FOR I = 1.1 TO 10.6
i.e. replace the TO value by (TO value + 0.5 * STEP value).
e.g.
FOR M = 0.25 To 1.502 STEP 0.05
The value of M after leaving the FOR loop will be 1.55.

8. Other uses for the FOR and Next statements.
Since the For variable is a normal BASIC identifier, it can be used

in statements executed after the FOR and before the NEXT statements, as in the examples shown. It can be treated as a more general loop - controlling statement. Consider the following iterative program for finding the square root of A:-

```
100 X = 1.0
110 X = 0.5*(X+A/X)
120 IF (ABS(A-X*X > = 0.0001) THEN 110
```

This repeats the loop of statements 110-120 a variable number of times until the terminating condition is satisfied - X is "close enough" to the answer. Now consider

```
100 X = 1.0
110 FOR Y = 0 TO 0 STEP -0.0001
120 X = 0.5*(X+A/X)
130 Y = ABS(A-X*X)
140 NEXT Y
```

This will loop until the NEXT action finds that

```
SIGN(ABS(A-X*X)-0.0001) = SIGN(-0.0001)
i.e. until ABS (A-X*X) < 0.0001
```

which is the same condition as before. The start value of the FOR statement is irrelevant. This version, although taking more statements, is more efficient because the slow process of searching for the line number to GOTO has been removed. In addition, the constant 0.0001 is calculated once only, on entry to the loop. This version can also be condensed, e.g.

```
100 X=1.0:FOR Y=0 TO 0 STEP -0.0001:X
=0.5*(X+A/X):
Y=ABS(A-X*X) :NEXT
```

as it is no longer necessary for the first statement of the loop to be the start of a line. This technique can be used for any loop which is terminated by an arithmetic comparison of values. By setting STEP to zero, it is also possible to repeat a loop until an exact zero if found, e.g.

```
100 REM READ AND TOATL NUMBERS
UNTIL A ZERO
110 T=0: FOR I=0 TO 0 STEP 0 : INPUT
I:T=T+I:NEXT I
120 PRINT "TOTAL IS";T
```

Another example is the simple sorting process introduced in paragraph 4. The loop

```
110 FOR I= 0 to N-1:IF A(I)> A(I+1)
  THEN X=A(I):A(I)=A(I+1): A(I+1)=X
120 NEXT
```

will pass once through elements O-N of the array putting adjacent values in order - but how often must the process be repeated? If, in the loop, the program records the value of I whenever it performs an interchange,at the end of the loop this value is the number of the last element which needs to be compared next time: all the elements with higher subscripts must be in order. This gives the following program:-

```
100 FOR M = N TO 0 STEP 0
110 L = 0
120 FOR I = 0 TO M-1
130 IF A(I) > A(I+1) THEN X =
  A(I):A(I)=A(I+1):A(I+1)=X:L=I
140 NEXT
150 M = L
160 NEXT
```

9. GOSUB and RETURN

The syntax of GOSUB and ON-GOSUB is the same as that of GOTO and ON-GOTO. RETURN is the entire statement. It must be followed by colon or end-of-line, otherwise a SYNTAX ERROR occurs. GOSUB and RETURN statements communicate by means of the same stack used by FOR and NEXT statements.

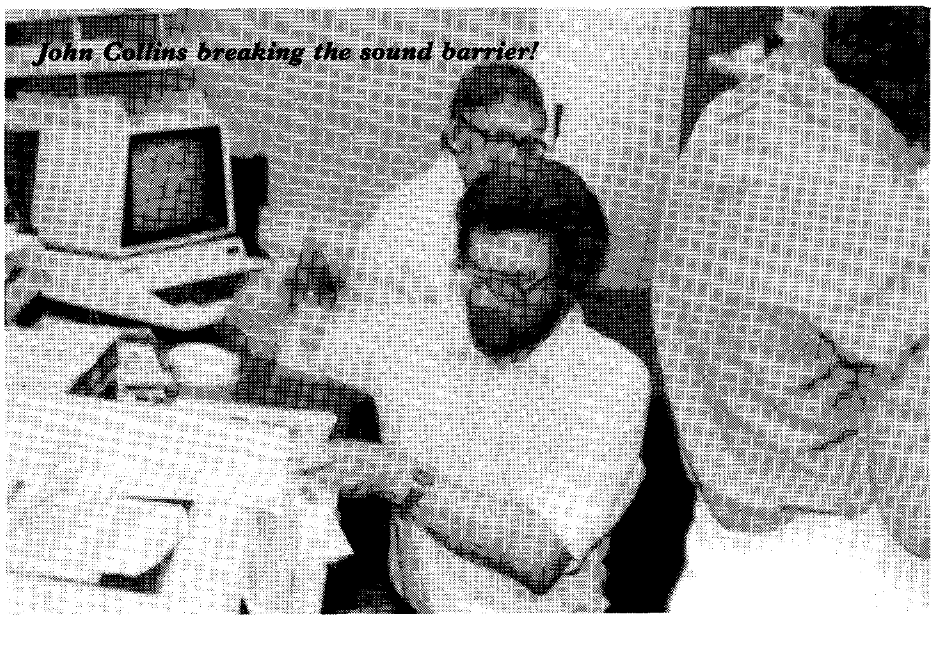
GOSUB causes a GOSUB packet to be added to the stack, containing:

The position in the line containing the GOSUB
Number of the line containing the GOSUB
'GOSUB' marker.

The normal GOTO action is then performed. Thus it is the GOSUB instruction itself which determines which pats of the program are to be used as subroutines; subroutines are not otherwise identified. If any FOR statements occur after a GOSUB, the check whether the identifier is already in use will be made only as far as the GOSUB packet; similarly, a NEXT statement will only search as far as the last GOSUB packet. Thus a FOR statement started before a GOSUB cannot be continued (by a NEXT) within the subroutine; and if the FOR variable is re-used in a FOR statement inside the subroutine, the original FOR statement will not be deleted from the stack, although the value of its variable will have been changed.

At a RETURN, the stack is searched for the latest 'GOSUB' packet, any outstanding FOR packets found being deleted. (Thus a FOR started inside a subroutine cannot be completed after a RETURN). BASIC then skips to the end of the GOSUB statement described in the GOSUB packet, and resumes execution of the next statement. It is impossible to "jump out" of a subroutine (e.g. using GOTO) bypassing Return - until a RETURN is executed, the program is still in the subroutine, wherever it may jump to.

GOSUB involves searching for line numbers just as GOTO does, where RETURN does not. As far as possible, therefore, subroutines should be included in the program with line numbers about 250 greater than the numbers of the lines containing the GOSUBS, or as near the beginning of the program as possible, unless the program calling the subroutines contains a large number of GOTOs.


John Collins breaking the sound barrier!