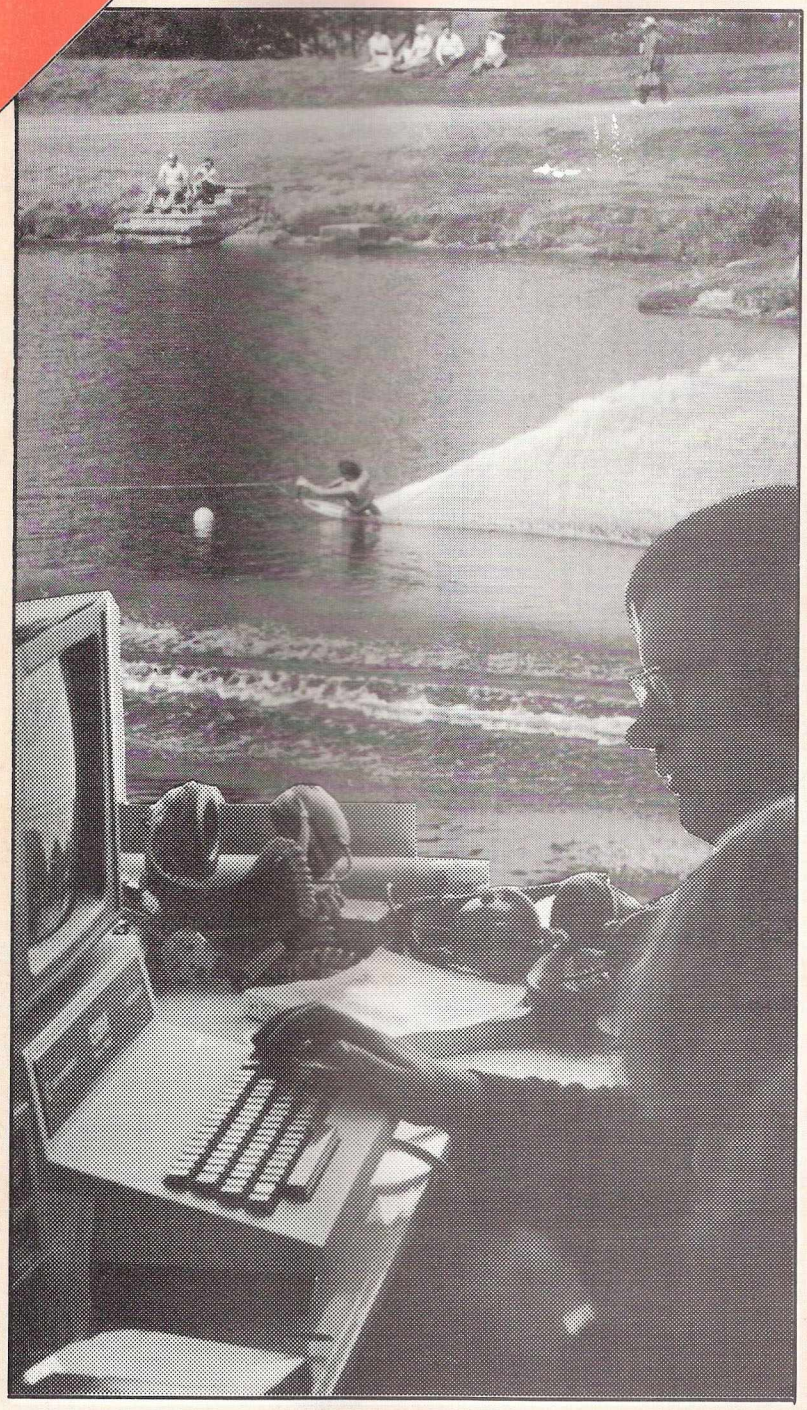
COMAL
SPECIAL REPORT
FROM THE STATES

COMMODORE CLUB NEWS

November 1981



Volume 3 Issue 10

BUTTERFIELD

Continuing his monthly forays in the PET World

USING THE USER PORT

A guide for beginners

ATTENTION OLD ROMMERS !

Upgrade to Basic 4 - how to do it

EDUCATION SUPPLEMENT

Special Pullout Section

commodore COMPUTER

"Every PET owner should read it"

Chuck Peddle, Inventor of the PET

"The PET Companion" is a fascinating collection of essential PET information from the pages of *Microcomputer Printout*. It contains **all** of the editorial from the 1979 & 1980 issues, including 105 PET programming hints and tips, 116 news reports, reviews of 54 peripherals ranging from light pens to printers and 27 major articles on PET programming. All of it written in straightforward English.

Some of the topics covered:

PROGRAMMING THE PET

Double Density Plotting
Modular Programming
Programming Style
Graphics
Subroutines
Sorting Out Sorts
Tokens
The Game of LIFE
Tommy's Tips
ROM Addresses

HARDWARE REPORTS

The New ROM Set
CBM 8032 SuperPET
CompuThink Disk Drives
Hardware Repeat Key
High Resolution Graphics
The Commodore Printer
How the Keyboard Works
AIM161 A to D Convertor
Commodore's 3040 Disk Drive
PET's Video Logic
Colour for PET: The Chromadaptor

THE SOFTWARE

Business Software Survey
Cosmic Invaders
Superchip
PETAID Do-It-Yourself Database
What's Wrong with WORDPRO?
Screen Display Aids
Keyboard Tutor
Photography Course
Who Do You Want To Be?: Fantasy Games
Commodore's Assembler Development System
Programming Aids & Utilities Survey
PET Games

THE SPECIAL REPORTS

PET in Education
PET Show Report
The Jim Butterfield Seminar
Hanover Fair Report
PET In Public Relations
Local User Groups
High Resolution Graphics
Commodore's New Technology
Future Shock: Forecasting The Future
Speech Synthesis
PET As Secret Agent
A Visit to the Commodore

plus news, letters, gossip and regular columns by leading PET experts.

To: Printout Publications, P.O. Box 48, Newbury RG16 0BD. Tel: 0635-201131

Please rush me a copy of the PET Companion

[ ] I enclose cheque/postal order/money order

[ ] Please charge my Access/Eurocard/Mastercharge or Barclaycard/Visa No:.....

[ ] UK £9.50 plus 45p postage [ ] USA $25 [ ] Overseas £12

(Credit card orders accepted by telephone on 0635-201131)

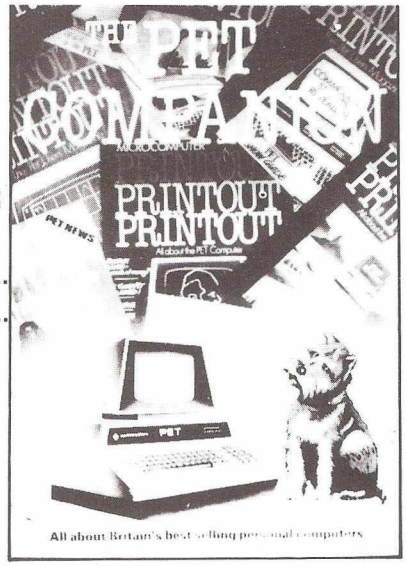
NAME: .....

ADDRESS: .....

.....

Postcode: .....

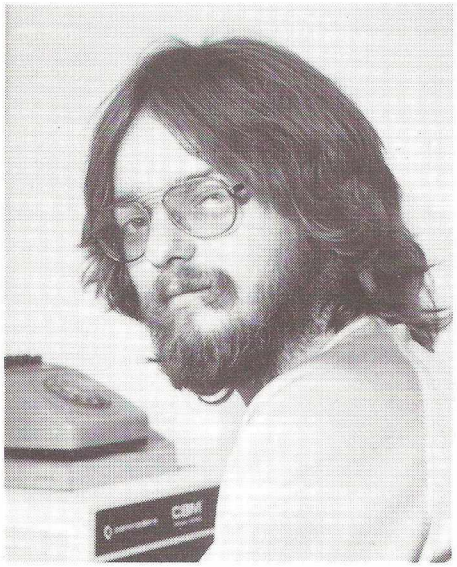
BARCLAYCARD VISA



Contents

Educational Supplement

Editorial



A number of people have enquired at one time or another : "How did you get the job of editor ?" Some have even asked "Why did you get the job of editor !?". For those of you with inquisitive minds, we present "The story so far....."

A little bit of background history. After leaving school with a variety of 'O' and 'A' levels in a somewhat peculiar mixture of subjects, my first encounter with the working world was as a lumberjack. Working in the far north of Scotland, where it was VERY cold, I eventually decided that this wasn't for me, mainly because it involved a lot of hard work. So off I went to university, University College London in fact, to take a degree in Astronomy. Despite discovering the student bar I finally emerged with a degree, and set off on the job hunt.

It took me six months to get that job. Six months in a bedsit watching the goldfish swimming around and turning the stero up to 78 r.p.m. to get the cat off it is a long six months. As soon as I mentioned that I'd got a degree in Astronomy people tended to say one of two things. Either "Oh, can you read my palm ?" Telling them that their hands needed washing didn't get one the job. The alternative response was "Ah, taking over from Patrick Moore are we ?", which didn't go down very well.

At last a job!

Ultimately my saviour arrived in the form of Nick Green, now Special Projects Manager at Commodore, but at that time in charge of the Software Department. Quite why Nick decided to take me on when others didn't I've never quite worked out : perhaps it was the drinks I bought him in the bar that evening!

After a stint of answering the telephones and sending out dealer leads, I graduated onto the cassette library and ended my days in software as software administration manager. It was then that I got the call into "The Office". We referred to it as "The Office" because it usually meant doom and gloom whenever we entered. When the conversation started I thought that this was what I had in store for me.

"Peter, you've been with us for over two years haven't you ?"

True, I thought, but is this the kind of thing that one slips into everday conversation ?

"Have you ever thought of doing another job ?"

It was around about this time that I fell on the floor. But rescue was at hand.

"How would you like to be editor of the newsletter ?"

After due thought and consideration I thought, well, why not ? Something different if nothing else. What you've seen in the last seven months has been the result : I hope you approve!

Working for Commodore

So what is it like to work for Commodored ? As you probably know Commodore is an American company, with distributors dotted around the world : we happen to be the largest one outside of the States. The usual company hierarchy exists, but (I think!) I get on well enough with the people above me. The over-riding impression of working for Commodore is that it's interesting : no two days are ever the same. It is also good fun, and at times considerably irritating, like, I would imagine, any company anywhere. But at the end of the day it's the interest factor that comes out on top.

Specifically, producing a newsletter is a very interesting job : the number of diverse application stories I see is growing all the time, and certainly on the increase is the number of letters I receive from you. All the programming hints are gratefully received : anything that will save other people "head scratching" is always welcome.

The Contents

Having mentioned the newsletter we come onto the contents of this one. The now standard features are all there, including the Jim Butterfield article, the reviews section, what the Papers Say, the peripheral spot, disk use for beginners, and so on. The Basic programming section this time covers such features as merging files, line protection, sorts and relative files, whilst on machine code we have a number of useful utilities.

Articles that go into more detail in specific areas include one on using the User Port, from A.H. Potten. Aimed at the beginner who has never encountered the user port before, it describes Mr. Potten's early attempts (and successes in using the prot, and suggests various application areas. He mentions topics like simulation of chemical process control, burglar alarms and even a PET controlled Christmas tree! (Less than fifty shopping days to go).

British Telecom leap to the rescue of Computhink disk drive owners who've been wishing to use Jim Butterfield's Cross Reference program. They've converted his program to work with these drives,and in a well documented article show precisely how it's been done.

Good news for old rom 8K PET owners who want to go to Basic 4.0. As you may be aware one of the problems facing you has been the fact that the old PETs had 28 pin sockets, and all the chips to convert to Basic 4.0 are 24 pin. In an article from Mogens Moller Nielsen of Denmark he explains precisely how one can overcome this problem, and at the same time goes into detail on how to produce your own character generator to, for instance, display pound signs on the screen.

Communications

One of our regular contributions, Pete Gabor from Datatronic in Israel, has sent in a program called Leapfrog,

and if you're an 8032 owner I've added a few notes to enable you to use the program. Although a game it does display a number of interesting programming features which will be of help if you're a relative newcomer to the Basic programming area.
A concurrent clock for the PET? Read on ....

Another item for old rom 8K users is a conversation of a program that was in an earlier newsletter, but which was written specifically for Basic 2. It's a single stroke key print routine, and thanks to Mr. Shaw of the Merseyside Microcomputer Group for providing the modifications.

Apart from the usual mixture of short listings and strange photographs, the other main feature in the main body of the magazine is an article called "Connecting the General Instruments AY-3-8910 Programmable Sound Generator to the 6502/6800 bus". More music for the PET!

The special central feature this time is once again back onto education after our forays into the world of communications. The main item of interest concerns itself with COMAL, and in particular with information from the COMAL Users Group from Madison, Wisconsin. Under the leadership of Len Lindsay they are rapidly becoming the world COMAL centre, and anything they produce I'll bring to you through the medium of this newsletter, If you've done any development work in COMAL let me know, as I'm sure Len, and other readers, would be very interested.
The address to send any contribution to is:-

The Editor
Commodore Club News
675 Ajax Avenue
Trading Estate
Slough
Berks.

If you wish to renew you subscription, or to start one up, write to Margaret Gulliford at the same address, enclosing a cheque for ten pounds (fifteen if overseas) made payable to C.B.M. (U.K.) Ltd. By doing this you will be guranteed a years supply of information, in the form of a magazine a month. Keep yourself informed!

'DIAL-A-DEMO' MOBILE COMPUTER DEMONSTRATION UNIT

Da Vinci Computers of Edgware have launched a new service for potential customers - a mobile demonstration unit.

Believed to be the first of its kind, the unit is luxuriously fitted with deep pile carpet, and is able to seat five for a demonstration in air-conditioned comfort.

The equipment includes a Pet computer, dual floppy disk drive, and two kinds of printer - dot matrix and daisy wheel - to demonstrate both ends of the price scale.

Da Vinci feel that more small businesses will have the opportunity of seeing for themselves how a computer can help in their particular business if the demonstration unit can come to them. This saves busy executive time and gives them a greater idea of how a computer can help with their particular application. The executive need never be out of contact with his office either, as the demonstration unit is also equipped with a telephone.

Apart from the hardware, Da Vinci are able to supply a wide range of software programmes suitable for all kinds of business and professional applications.

To 'dial-a-demo' (free in London and surrounding areas) just telephone the Mobile Unit on 01-882 6481 Code 1882 or Da Vinci's Offices on 01-952 0526.

INTERNATIONAL

DENSPET is the association for the exchange of original programs for the MTU (and IJJ) 200 X 320 dot high-resolution PET accessory. Send a sample of your work or £2.50/$5.00 and receive a sample in return plus subscription to newsletter and lists of available programs. Write to Frank Chambers, DENSPET, Rock House, Ballycroy, Westport, Co. Mayo, Ireland.

ITL INPUT/OUTPUT CARDS FOR LSI-11 MICROPROCESSORS

The ADAC Models 1632TTL, 1664TTL and 166ATTL, each contained on half boards (8½" x 5"), have the capability of interfacing a total of 32 or 64 TTL digital input/output lines directly to the Digital Equipment Corporation LSI-11, LSI-11/23 bus as well as the ADAC System 1000 and 2000 Series bus.

The ability to change input/output lines to be either inputs or outputs in any combination in increments of eight makes these cards unique. This is accomplished on the 1632TTL and 1664TTL by inserting pluggable components into sockets which are mounted on the printed circuit boards. On the 1664ATTL reconfiguring is accomplished by inserting pluggable jumpers.

The 1632TTL contains a 16 bit status register and provides the flexibility of operating either under program control or program interrupt. Two separate interrupt circuits are provided, one for each 16 bit input/output port. Four uncommitted read/write bits of the status register are made available to the user for control purposes. The 1664TTl and 1664ATTL contain four 16 bit registers and operate under program control. Reading and loading of the register can be handled on a word or byte basis. Both cards contain command signals useful for transferring data to and from an external device.

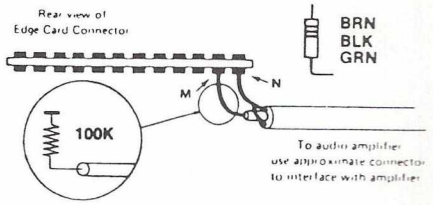
The 1664ATTL is designed with latches rather than latch/counters and, therefore, are more noise immune when driving long cables. The 1664ATTL can also be jumpered to supply fused +5V through the input cable to power opto isolation panels.

The 1664TTL has a jumper programmable reset feature which allows all output latches to be either set or cleaned upon a bus reset signal.

Priced from £190 each, these cards are an addition to the range of over one hundred LSI-11 compatible function cards available from Amplicon Electronics Limited.

MORE ON SOUND FOR YOUR PET

Last issue, an article appeared in the COMMODORE MAGAZINE which explained how to connect an audio amplifier to the CB2 line on the user port. Ordinarily, there is no harm in making the connection directly as shown in that article. However, it is possible that you may be using an amplifier which, either due to a malfunction or by an uncommon design, will present the CB2 line with an overload condition. Since there is no overcurrent protection for the CB2 line, it would be wise to provide such protection for the CB2 line yourself. This is simply done by inserting a 100K, 1/10 or greater watt resistor in series with the signal line (ie. pin M or centre conductor of cable) as shown in the following illustration. This will protect the CB2 line even if there is a short at the other end of the patch cord or even in the cord itself.



If you own an 8032 or the 12" display 4016, you will no doubt be aware the computer produces sound without the aid of any add-ons thanks to the internal 'speaker'. Consequently, running programs with sound included, will work without any soldering or screwdriving. Experiment with sound in your program, as it can be a very valuable addition e.g. it draws attention to when an input is required or accepted.

When the salesmans back is turned, try this one line instruction on an 8032 to change the 'chime time' -

```
POKE 231,150: PRINT CHR$(7)
```

The following table is of use when programming with sound as it gives the musical note equivalents to the poke values.

POKE 59467,16 to enable sound.

POKE 59466,51 generates a pleasant square wave. Also try values of 10, 14, 15, and 85.

POKE 59464,0 is silent. The following values are for the musical note equivalents listed.

```
Bb=251 (B below first C)  B  =124
C =237 (first C)          C1 =117
C#=224                    C1#=111
D =211                    D1 =104
D#=199                    D1#= 99
E =188                    E1 = 93
F =177                    F1 = 88
F#=167                    F1#= 83
G =157                    G1 = 78
G#=149                    G1#= 73
A =140                    A1 = 69
A#=132
```

Try the following two short programs.

```
10 POKE 59467,16: POKE 59466,51
20 LET X=INT(255*RND(TI)+1)
30 POKE 59464,X
40 FOR I=1TO250: NEXT I
50 POKE 59464,0
60 FOR I=1TO250: NEXT I
70 GOTO 20
```

```
10 POKE 59467,16: POKE 59466,10
20 FOR I=1TO10
30 FOR U=255TO1STEP-1
40 POKE 59464,U
50 NEXT U
60 NEXT I
70 POKE 59464,0
```

NOTE: Always remember to restore the cassette functions by POKEing everything back to 0.

Arcadaphobia

Thanks to the people who've written in lately with their highest scores. So here we have the latest list of superstars: some of these scores are truly quite amazing. What else do these people do ?!

Invaders	—	57,790	by C. Douglas-Fairhurst
Acrobat	—	47,460	by Patrik Albertsson
Breakthrough	—	4,477	by C. Douglas-Fairhurst
Night Drive	—	1,528	by Patrik Albertsson
Car Race	—	9,560	by Pete Gerrard
Crazy Balloon	—	8,540	by David Pocock
Cosmic Jailbreak	—	128,140	by Giles Murcott
Cosmaids	—	388,290	by Matthew Sargaison

So that means that only two records have survived since issue 4, those being for Car Race and Crazy Balloon. Keep 'em coming!

Missile Attack

Missile Attack is an unusual and highly entertaining new game for PET systems. Players must demonstrate both manual dexterity adn decision making in order to win.

The object of the game is to defend three cities from decending enemy missiles. This is achieved by moving cross-hair sights into the path of the decending missile and then launching an anti-missile from one or three launch sites.

The skill of the game is to decide from which base to launch the missile, where to position the sights, and when to launch. If a missile is launched at the right time a number of the decending missiles can be caught in the same explosion thus conserving the limited number of anti-missiles available at the three bases.

As each attack wave is defeated bonus points and cities are awarded depending on the difficulty level and how much cities and anti-missiles have been conserved.

Getting used to using the controls may prove quite a challenge for the beginner as the cross-hairs seem to have a life of their own.

As the player becomes more experienced, different techniques need to be mastered to destroy the decending hordes and the high score can always be pushed a little higher.

At present the game retails at 15.95 for diskette and 13.95 for cassette (inclusive fo VAT, postage and packing). Attractive dealer discounts are also offered for bulk purchases.

For further details please contact:-

Softprint
11 Wyverly Crescent
New Malden
Surrey

Second Hand PETs

2040 disk drive, 3020 PET (complete with Toolkit) and a 3040 disk drive, all in good condition, with an asking price of 1,450 pounds. Name and address to contact :-
Peter Smith
48 Woodham Ways
Woking
Surrey
GU21 5SJ
Tel. Woking 67839

3022 printer, 19 months old with light use, for 395 pounds. Free delivery if you live in the Republic of Ireland. Name and address to contact:-
Frank Chambers
Rock House
Ballycroy
Westport
Co. Mayo
Ireland

Reviews

Barry Miles

Faster Basic - Supersoft

There is considerable interest in the various compilers becoming available for the PET. These offer considerable speed improvements compared with interpreted programs.

However, the programs available are not cheap, particularly for the hobbyist user. It is therefore a welcome step for Supersoft to make available another of their enhancement chips; this time called "Faster Basic". As is now usual for Supersoft chips versions are now available for any PET or CBM machines, and the program will reside in any spare socket : the purchaser merely states his choice at the time of purchase.

As the program is less than 2K, a small extra fee will enable you to have this chip and another 2K Supersoft product incorporated into a single 4K chip. This is both convenient and cheaper than using a Spacemaker, Socket-2-Me, or Rompager for the same purpose.

The program speeds up the execution of GOTO, GOSUB or THEN followed by a line number. Variables and integers are interpreted more quickly than usual, and variables are found in memory instaneously.

The program actually modifies each program statement the first time it is executed, but any changes are reinstated when listing or saving is carried out.

A simple SYS toggles Faster Basic on and off, even while a program is running!

The program is fully compatible with all BASIC commands, and places no constraints on how the program is written (unlike some compilers).

The effects seems to be about 50% to 100% speed improvement in Basic programs. Of course, machine code programs or subroutines will not be improved.

The only disadvantage likely to be encountered is that programs which modify themselves (a somewhat exotic rarity) will probably fail to function.

Finally, and this is a major aid to producing programs that are well structured, if you are using Faster Basic, many of the tricks which you are used to using in order to speed up program execution, e.g. declaring variables at the beginning of the program, most frequently used ones first, and so on, will help you even more.

All in all a well designed programmers aid.

Instant Rom - Greenwich Instruments

This product is unique compared with its competitors, in the range of products offered, in the mode of operation and in the variety of low-priced gadgets offered, which considerably enhance its capabilities and ease of use.

The concept is simple : a combination of ROM (Read Only Memory) and RAM (Random Access Memory), in a single package, capable of being plugged into any socket of the CBM machine, be it a Basic slot or a spare slot. More correctly, it is RAM which can be made to look like ROM.

The special feature which make this range of products unique is the internal battery, which makes the data capable of being retained for up to 10 years! This means one has the equivalent of an EPROM (Erasable Programmable Read Only Memory), which can be reprogrammed at high speed. A program can be wirtten into Instant Rom using any Assembler or Monitor, and run immediately! This is accomplished by connecting a lead to the Write Enable pin of the memory expansion port. Removal of the lead makes the program completely secure.

An additional lead may be taken from the Instant Rom to the System Reset pin. This saves one having to remove a lead on switch off, and prevents garbage being written at the moment of switch on! However, your program may be erased accidentally!

These handy devices come in 2K, 4K and 8K sizes, and the latter enables two 4K blocks of memory to share one 4K socket, and, using one of the many adaptors also offered, switching between the blocks is achieved by software, using the 8 bit output port, also available from Greenwich Instruments. This ingenious device plugs into any ROM socket, and the ROM is then plugged into the port. The ROMs behaviour is not affected in any way.

The port has 8 output pins, corresponding to bits D0 to D7 on the data bus. Writing or Poking to the port sets any pin to 1 or 0, so that each pin may be used to select a ROM by connecting it to any other adaptor, or to an Instant Rom directly. A single Poke can activate up to 8 ROMs, and the adaptors may be daisy chained!

Other adaptors enables two ROMs, EPROMs or Instant ROMs to occupy the same socket, and they are then selected by means of a link, or from software!!

The breaking of pins on an Instant Rom is not to be recommended, the 8K version costs 89 pounds, so it is a relief to know that the company supplies, for a mere 3 pounds, a carrier that will stand many insertions and removals without the risk of damage.

The whole range is extremely versatile, and the non-volatility of the memory is extremely welcome. What you get with Instant ROM is the convenience of an EPROM, without the inconvenience or cost of an EPROM programmer, or of an eraser. Anyone interested should study very carefully the excellent documentation provided, because the possible permutations of the various adaptors are numerous, and will repay careful thought.

The company also suppliers "Charger", which is a tape based program for setting up your own character generator and putting it into Instant ROM. If you use the 8K Instant ROM you can choose between four new fonts and the original one! Using the output port, you can switch between them from software! You can also modify the characters at will.

The range includes a VIC system, 8K of Instant ROM, volatile or non-volatile, for memory expansion and ROM emulations.

So what are these Instant ROMs to be used for ?

The most obvious use is for rapid program development, particularly where it is intended that the

program should reside in one of the spare ROM sockets. It will be a much quicker process to use these devices than it will be to blow and re-blow a series of EPROMs. Apart from this, the simplicity of changing code, in its eventual location, is attractive!

Autoboot systems are also a possibility, and Basic programs can be made to load and run in expansion sockets, by the use of simple techniques.

The owners of programs protected by chips may find it much more convenient to save the contents of the chips onto disk, and load those into instant ROM as required. It is possible to save the program plus security chip, so that it loads in one go, although the loading then becomes a lengthier business than usual. Similarly, if you own a variety of propriety chips, and have no wish to lay out cash for Spacemakers, Rompagers, and the like, you may use the same technique.

If you are an inveterate chip collector, you may have a special, disk, with a menu asking you which chip to load!

Altogether, the entire range of products is very attractive, and well worth consideration by the serious user.

STOLEN.....

The following units have been stolen from Eastbourne College of Education, Ames Road, Eastbourne, Sussex:-

1 4032 Ser No. 100525

1 3040 Ser No. 608564

1 3022 Ser No. 115399

If found, contact Detective Constable Haddow of Eastbourne C.I.D. on (0323) - 22522 X 245/7

Review

D.J. Pocock

Assembler Tutorial

A new product just released by COMMODORE, which is a must for all would-be machine code programmers is the ASSEMBLER TUTORIAL. This package, developed by W.Owen Murcott, will help you learn how to write Assembly Language programs.

The tutorial is split into three modules, these modules consist of an introuction, a self test (in the case of module three a set of Programming Exercises) and several topics. As you can see below, each lesson deals with a different topic. The self-test is a progress check. It enables you to see which topics you need to study further.

The outline for the package is given here.

OUTLINE

```
0.0  GENERAL INTRODUCTION

Module 1 - INFORMATION REPRESENTATION

1    Module Introduction
1.0  Self Test
1.1  Binary Numbers
1.2  Sign Conventions
1.3  Binary Arithmetic
1.4  Logical Operators
1.5  Magnitude and Fractions
1.6  Binary Coded Decimal
1.7  Floating Point
1.8  Character Codes
1.9  Hexadecimal

Module 2 - HARDWARE AND PROGRAMMING

2    Module Introduction
2.0  Self Test
2.1  The PET
2.2  Memory
2.3  Stack
2.4  6502 Microprocessor
2.5  Accumulator and Arithmetic
2.6  Addressing Techniques
2.7  Indexed and Indirect Addressing
2.8  Interupts

Module 3 - SOFTWARE AND PROGRAMMING

3    Module Introduction
3.0  Programming Exercises
3.1  Assembler and Machine Code
3.2  Program Building and Assembly
3.3  Linking with BASIC
3.4  Software Aids
3.5  Assembler
3.6  Machine Code Saver/Loader
```

An early version of the ASSEMBLER TUTORIAL was on show at the PET SHOW this year and generated a great deal of enthusiasm among the representatives of schools and colleges. The final version is now being marketed.

The ASSEMBLER TUTORIAL can be used as an aid to classroom teaching or as an individual 'teach yourself' course. Demonstrations of some of the more complex concepts involved are included in the lessons. Repetition of individual sections is easy if you did not fully understand them the first time. This facility enables you to progress at your own pace.

The ASSEMBLER TUTORIAL is an excellent aid to those wanting to learn Assembly Language programming. The ASSEMBLER TUTORIAL can also teach a thing or two to those who already know something about Assembly Language Programming. (as it did to me). Of course if you are up to the same standard as Jim Butterfield, you will not need it.

The ASSEMBLER TUTORIAL will be available on tape or disk from your local dealer. A users handbook which outlines the course is supplied with each package.

The cassette version will run on any 40 column PET with 8K RAM or more, however users of BASIC 1 PET's (the original ones) will have to put up with mixed upper and lower case. It is a set of four cassettes containing the 30 programs. This version will be sold for approximately £50.

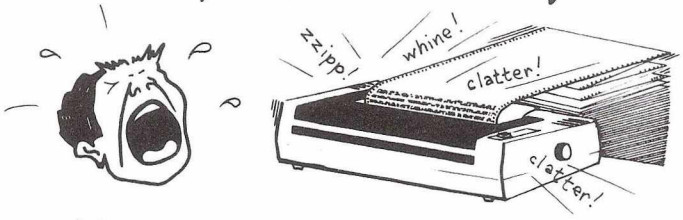
The disk version will have the 30 programs on two disks. These will be available at about the same price.

The following table gives the order number for disks that will run on the different COMMODORE systems.

Processor	Disk	Order Code
3016	2040	
3032	3040	AST3140
	4040	
4016	4040	AST4140
4032	4040	
8032	4040	AST8340
8032	8050	AST8350

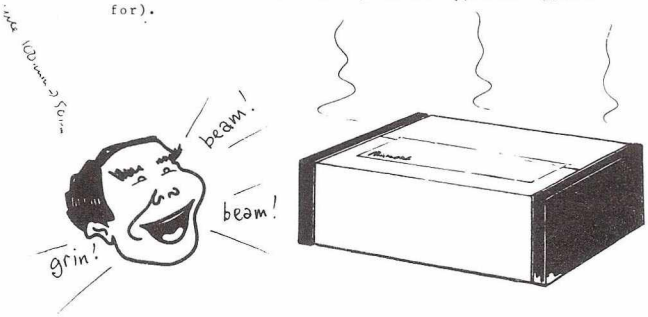
Printerproof Acoustic Covers

tell everybody - keep it quiet!



With 'Printerproof' - the strong quiet type!

Now! AN ADDITION TO THE PRINTERPROOF RANGE OF SOUND ENCLOSURES. Following the success of the CBM-3022 size model, the Acoustic Specialists at PRINTERPROOF have designed a Cover for the new 4022 Printer from Commodore. (Product Approval Applied for).



NO NOISE IS GOOD NOISE! 'PRINTERPROOF' IT FOR YOURSELF!!

The efficient sound reduction by the latest addition to our range brings the noise-level of the printer to well within the Noise Criteria for General Offices (ISO Rating 55) and is achieved without sacrificing accessibility to paper or switches.

Fully-tested, precision-made and built to last in sound-deadening heavy-gauge aluminium with perspex window and foam-lined for maximum noise-absorption the PRINTERPROOF cover is attractively texture-finished in Black and Cream to please the eye as well as the ear.

And you can say goodbye to paper spillage thanks to the ingenious two-option paper-feed guide devised and incorporated by our designer to control your paper. Illustrated instructions are enclosed with every unit.

The internal dimensions of 41cms wide by 37cms. deep by 15cms. high, mean that other printers similar in style to the CBM 4022 can be accommodated. External dimensions are: 46cms. by 41cms. by 17.5cms.

The price of £129 + £2 carriage + VAT buys, we claim, the most effective Acoustic Cover on the market. You will be delighted along with all our users.

You'll ENJOY THE NON-SOUND OF IT!

TRY NOT HEARING IT FOR YOURSELF!

Printerproof

A Division of Bradton Ltd.

Your cheque with orders, please, to PRINTERPROOF Sales Office, 12 Wokingham Road, Reading RG6 1JG, or contact Peter Stayne on Reading (0734) 661432 for further information.

Let us know the Make and Model of any noisy printers and we will be pleased to quote by return of post for a suitable cover.

Trade enquiries welcome.

12 WOKINGHAM ROAD READING RG6 1JG Telephone: Reading (0734) 661432

FREE PET/CBM COMAL

"The excitement in Europe (over COMAL) seems to be growing by the hour and we look forward to America being able to share in the good fortune of having an easy-to-use, structured, planning language at last."

The power of PASCAL and the ease of BASIC can now be yours with Commodore COMAL, a new programming language from DENMARK. It is being distributed in the USA by the COMAL USERS GROUP. To find out more about COMAL and how you can get a free disk copy of Commodore COMAL, send a large self-addressed stamped (35 cents) envelope to:

COMAL USERS GROUP
5501 GROVELAND TER., MADISON, WI 53716.
Outside USA please add $2.00 for airmal and handling.
*PET & CBM are trademarks of Commodore Business Machines.

GBS Software

23 Park Hall Road, London N2 9PT. Telex: 298951.
Telephone: 01-444 5104 & 01-404 5011.

5¼" FLOPPY DISKS:

All Double Density for Commodore

WABASH Single Sided 70 track £16.50
VERBATIM Single Sided 40 track £18.00
VERBATIM Single Sided 77 track £28.00

We have a few shop soiled commercial programs in stock - up to £100 off list price (diskette & security chip are mint - only the manuals are well thumbed). Send s.a.e. for details.
Computer books by mail order - no extra to pay for postage or VAT. Send s.a.e. for details.

GBS SOFTWARE - EVERYTHING COMMODORE

PETs take to the Waves

Gail Wellington

What was Commodore's David Pocock doing at the World Waterski Tournament on the shores of Thorpe Park's Brittania Arena in the middle of the working day? Around the office the location was more of a surprise than the timing because we refer to David as "the square format man". The author of Club News' series of articles on disk programming spends most of his working hours in front of a V.D.U. His favourite outside pastimes are watching television, going to the cinema, or taking photographs - all square formats!

So how did he happen to be sitting on the grass in the September sunshine watching people hurl themselves through the air or gyrate on the surface of the water whilst being towed behind a speedboat? Well, it all started about a week before the tournament when one of the tournament organisers rang Commodore and asked for some help with his software. He wanted to add disk file access to his existing program for scoring the tournament events. David was assigned to help and went to the tournament in case any last minute fixes were required.

The waterski competitions consist of three events; jumping, trick sking and slalom sking, with the top men and the top women in each event being awarded prizes. By a complex calcuation points are given in each event which are then totalled to determine over-all standings. COMMODORE computers calculated all of these results.

The 1981 Men's overall Champion this year was Sammy Duvall, and the Women's honours went to Karin Roberge. Both are Americans, Duvall unseating British Champion, Mike Hazelwood. Ninety men and thirty-nine women competed in all, representing 30 countries, including such distant ones as the Phillipines and Argentina.

One COMMODORE computer was dedicated solely to the task of keeping track of the individual scores. Slalom scores are based on the number of bouys succesfully rounded at a set boat speed and with a specified rope length. If a pass is successful, the rope is shortened.

Women's champion in the Slalom was Cindy Todd of the U.S.A. with a final score of 31.25 buoys. Cindy completed 1.25 buoys on an 11.25 metre rope. Although before the Thorpe event no woman had ever scored with such a short rope and Miss Todd's combined score for two rounds won her the championship, she did not turn in the best performance of the day. Sue Fieldhouse of Australia set the new world record with 1.50 buoys at 11.25 metres.

The Men's Slalom title went to Andy Mapple of Great Britain in an unexpected victory. Mapple, an 18 year old from Lancashire, scored 4.25 buoys on an 11.25 metre rope, for a new European record. Carl Roberge of the USA was second with John Battleday of Great Britain a close third. The high placings of the two British lads made up somewhat for Hazelwood's disappointing fourteenth place.

Trick sking requires performing a series of turns on the surface of the water and low jumps and turning jumps off the boat's wake. Two runs of 20 seconds duration each constitute a round. The better skiers make at least one run per round on one ski while holding the rope with the other foot. The tricks are assigned points according to their difficulty and the object is to complete as many different tricks as possible in the alloted time.

The large crowds on the shores of Thorpe's Britannia Arena saw Corey Pickos of the USA score an outstanding 17,260 points in the two rounds. The greater excitement came, however, in the women's event when Ana Maria Carrasco of Venezuela edged out Natalie Roumiantseva of Russia who was ahead at the end of the first round.

In close competition like this, the second COMMODORE computer was put to use. The commentators both for the crowd at the sight and on live television wanted quick projections, regarding the points required to upset the standings as well as fast results.

This was particularly true for the overall standings. "If Ana Marie does better than yesterday by say 100 points, how many trick points does Karin need to be first overall?" "How far does Mike Hazelwood have to jump to take the overall title from Sammy?" These were the questions that the COMMODORE people were able to answer in seconds using the computer.

The jumping event is simply scored. The longest of three jumps is the competitor's score for that round. As you might imagine, determining the distance is not so simply accomplishied! Two sighting towers are located along the shore, and judges are positioned behind a sighting instrument at two levels on each tower. The angle from each instrument to the jumps is measured at the start of the event, and the angle to the point where the skier lands is also measured. Since the distance between the two towers is known, by geometry it is possible to calculate the location of the jumps and of the loading point and thus the distance between them. Needless to say, jump calculations are a natural computer application.

Predictably, Mike Hazelwood of Great Britain won the men's jumping trophy. The crowd had hoped to see the elusive 200 foot (61 metre) mark fall, but 57.9 was the best Mike could manage. Deena Brush of the USA won the women's event, with a best distance of 39.60 metres.

The determination of overall standings is perhaps the most complex of all of the calculations and therefore really shows the advantage of the computer. A "combine" score (pronounced "com-bin-ay") is calculated for each event and the three combines for each competitor are added together to give that person's overall score.

The combine is determined by assigning 1000 points to the competitor with the best single score in any round. A proportion of 1000 points is then assigned to the other competitors based on the ratio of their score to the best score. For example, if the best trick score were 2000 points, this would be worth 1000 combine points and a skier with 1000 trick points would be awarded 500 combine points.

It is easy to see from this that if anyone in the final round outperforms the best of the previous round,

all of the combine values change. The COMMODORE computer system not only made the initial calculations simple but when a new top score was achieved, updated the standings almost instantaneously. London Weekend Television positioned a remote camera to overlook the screen display for these timely results.

The program to perform the scorekeeping and jump calculations was written by M.C. Forsdyke and Trevor Hill who themselves are Waterski enthusiasts. Some modifications were made by Dave Pocock of COMMODORE to include disk access.

After the tournament was underway, the officials approached the COMMODORE representatives and asked if it was possible to calculate team standings as well. Modifying the existing software at such a late date was considered unwise so a third COMMODORE computer system was brought in. THE MANAGER, their new data base management system was run on this machine.
The program created a record for each country and the top three scores for each event irrespective of gender, were entered. THE MANAGER totalled each countrys points and sorted them in descending order. A report was then generated in the format required by the World Waterski Union officials. COMMODORE was able to design the input and report formats for this task in about half an hour thanks to the flexibility of THE MANAGER.

Throughout the finals on Saturday and Sunday, COMMODORE personnel updated the standings as required. Within minutes of Mike Hazelwood's gold medal winning jump, results calculated by THE MANAGER were on international television broadcasts and were being telexed to press bureaux throughout the world. These were the final results:-

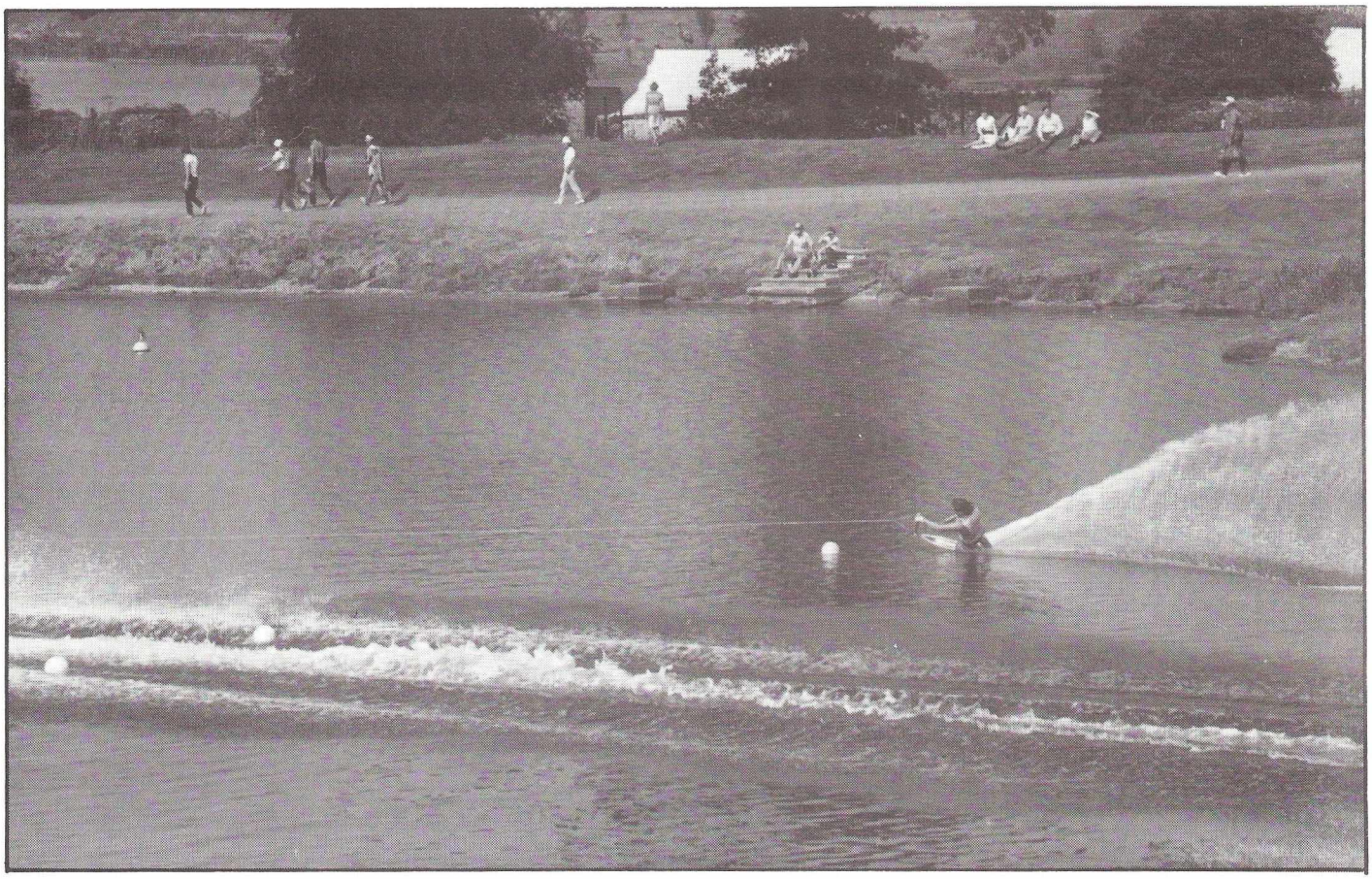
USA	8519.86
AUSTRALIA	7777.03
GREAT BRITAIN	7665.97
CANADA	7521.82
USSR	7000.17

As with any special event, all did not go as smoothly behind the scenes as it appeared to the more than 10,000 spectators lining the shores. Power overloading resulting in outages was a continual problem.

As anyone who has been around computers knows, this could have been disasterous. However Edcel Electronics provided COMMODORE with one of its Centaurus Linebackers. This simple-appearing grey box sat unobtrusively under the table. When a power cut was experienced, the Linebacker came on automatically. The instantaneous AC power from its battery meant no outage on the COMMODORE computers and therefore no loss of records.

The use of computers in watersking was not a one-off for this international event. Messrs. Hill and Forsdyke had already used their program at such events as the British National and Junior Competitions. The plan for next season is to make the program menu-driven and introduce additional handholding and checks. This will reduce the special knowledge required of the operator. Moreover, some of the waterski group with electronics backgrounds are working to perfect an interface that will enable the jump judges to align their instruments and have the readings automtically transmitted directly into the computer.

Computers, particularly microcomputers, have long been a part of the hobby scene. Now, as was evidenced at the World Waterski Competition by COMMODORE, they are finding a place in other hobbies and sports. The relative ease of transporting them, their low power requirements and the almost infinite flexibility of their software means that such use of microcomputers can only expand.



Everybody Knows....

Jim Butterfield, Torronto

Some things are so well known about the PET/CBM that nobody thinks to say them any more. In fact, if you're a new PET owner or just happen to be around when everyone else found 'obvious' features, chances are that you still don't know them. These facts are so well known that nobody has bothered telling you about them.

All of the following items apply to all makes of CBM/PET unless specified otherwise.

All PET/CBM computers have both: upper/lower case characters, called 'text' mode; and graphics/lower case characters, called 'graphics' mode. You can select text mode with POKE 59468,14 or graphics mode with POKE 59468,12. On newer machines (80 column and Fat 40) you can select text mode with PRINT CHR$(14) and graphics mode with PRINT CHR$(142); in this case the screen display is trimmed up slightly. Not bad when you consider that on some computers, upper/lower case is an expensive extra.

If you buy a 32K PET/CBM computer, you might think that you are getting 32,000 memory locations (called 'bytes'). Wrong on several counts. First of all, a 'K' represents 1,024 bytes of memory, so your total goes up to 32,768. Next, Commodore throw in the screen memory for free - that's an extra 1K on 40-column machines and 2K on 80-column machines that hasn't even been counted. So if you have all that memory, how come a fully fledged 80-column machine with a total of 34, 816 bytes tells you '31743 BYTES FREE'? Answer: Because it's telling you how much space is available for your Basic program; and that doesn't count either screen memory or a number of 'overhead' locations that have been set aside. Final puzzler: in that case, how come if you ask for the computer's Basic free space right away, by typing PRINT FRE(0), you get an answer of 31741? Where did those two bytes go ? Hint: Basic uses a two-byte marker to say, "There's no program beyond this point".

When you press the RETURN key, go to the next line. More than that: everything on the line you're leaving is received by the computer. That means you can go back to a line on the screen, change it if you wish, and then press RETURN and the whole line will be used. If you want to go to the next line without having the previous line actioned by the PET/CBM, hold down SHIFT as you press the RETURN key.

The PET/CBM has a built-in timer/clock. Type PRINT TI$ and you'll be told how much time has elapsed since you turned the computer on. The TI$ clock works in hours, minutes and seconds, which means you can set the correct time into it. If it's 3.25 pm, just type TI$="152500" and it will keep reasonably good time from that point on. Since it's hard to do arithmetic in hours, minutes and seconds the computer gives you a second timer that counts in 'jiffies': a jiffy is 1/60 of a second. The clock and the jiffy timer are really the same timing device: they both reset to zero when TI$ reaches the 24 hours mark.

Most Basic commands can be abbreviated. The question mark is a quick substitute for PRINT, but other commands can be abbreviated using a method along the following lines: to print VERIFY the short way, type the V and then hold down the shift key and press the E key. The result may look a little odd, but when you press RETURN, the computer will behave just as if you had typed the full word. Some keywords are not worth abbreviating: IF and TO are too short, for example. Others will give trouble because they start out similarly: GOTO and GOSUB start with the same two letters, as do NEXT and NEW, POKE and POS, or READ, RETURN and REM. The computer doesn't know which you want, so might pick the wrong one.

The PET/CBM will receive from the keyboard even when there's a program running. It will normally store up to nine characters, holding them in a 'keyboard buffer' until they are needed. If desired, you can pick up keystrokes one at a time while your program is running with a statement like GET X$ ... X$ will be equal to the key pressed, or to nothing (the null string, written " ") if no key has been pressed.

You can clear the screen, home the cursor and/or move the cursor in any direction as part of a PRINT statement. When you type in the statement, the desired screen activity is stored as an odd-looking reversed graphic. Later, when the PRINT statement is executed, the movements take place as planned. The mechanism is called "programmed cursor" and it's very good for screen animation.

The newer PET/CBM units contain a sounding device, but in any case you can get sound signals from the Parallel User Port. That's the middle of the three connectors at the back, and you should connect pins M and N to your amplifier - your stereo will do - since the signal isn't strong enough to drive a speaker directly. You generate sound from your program in the following manner: POKE 59467,16 turns the sound capability ON - before your programs stops, you must turn it back off with POKE 59467,0. POKE 59466,15 sets the tone - 15 is a good number but you can try others. POKE 59464,100 sets the pitch - any value from 10 to 255 is audible. Don't forget: turn it off when you're finished with POKE 59467,0 or you may have trouble with input/output.

All machines except the first (Original ROM) have a built-in Machine Language Monitor. You can get to it with SYS 4. This should produce about three lines of cryptic information. Return to Basic by typing X and RETURN. Later, you may learn to do marvellous things with the MLM - to start, just know it's there.

It's sometimes useful to know if someone is holding a key down. You can discover this by using the function PEEK(151) ... on Original ROM machines it's PEEK (515). If the value you get is 255, nobody is touching a key at the moment. If it's anything else, a key is being held. Don't try to use the value if it's lower than 255 - the numbers will vary from machine to machine - it's better to use GET to find out which key has been pressed.

Are there other features that everybody knows? I'm sure there are lots of others... the problem is, they are so obvious I can't think of them...

WordPro 4-Plus

Word Processing Software Program

Turns your Commodore CBM™ 8032 computer into a highly sophisticated, 80 character screen word processor loaded with the same inventory of features found in other professional word processing systems costing thousands of pounds more.

For Use With:

- Commodore CBM 8032 Computer with 80 column display.
- Commodore CBM 2040/4040/8050 Dual Drive Floppy Disks.
- NEC Spinwriter, or any other properly interfaced ASCII printer.

User Advantages:

WordPro 4-Plus™ offers an exceptional text editing, document storage, and typewriter quality printing capability to any business whose needs would benefit from the increased productivity inherent to word processing. It includes every entering, editing, memory, and printing feature considered important to sophisticated word processing. Compared with other available systems, many of which cost up to twice as much as the system that WordPro 4-Plus creates, it is unusually easy to learn, and just as easy to operate.

Update your Character Generator

Mogens Moller Nielsen

This article will show you how to "roll your own" screen characters, as f.ex a £-sign.

If your PET is of the 2000-series it will also help you to be able to use the advertised special character generators with mathematical symbols or foreign letters or just updating to the newer Commodore char.gen.'s where you shift to get capitals.

The trouble with the 2000-PET is that it uses a 28-pin char.gen type 6540, and the newer char.gen's are 24-pin 6316's. (which are pin-compatible with 2516/2716 EPROM's).

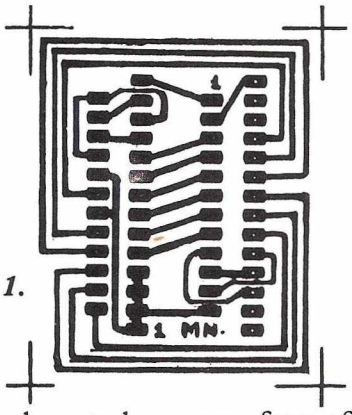


Figure 1.

So you have to have some form of adapter to use the 24-pins IC's in the old PETs 28-pins socket. Fig. 1 shows a single-sided print lay-out which together with an ordinary 24 DIL socket and a Pick-A-Back socket (EURO-DIP part no. HST 28 OG) will do the job. Instead of the pick-a-back socket, which may be a little hard to get, you could use some stiff wire of a size that will fit into an IC-socket. Notice, that when you mount the adapter with its 24-pins char.gen, its notch (Pin 1) points to the left whereas the old 28-pin char.gen ROM has its notch to the right. (Fig. 2). (Pin 1 on the new char.gen. goes via the adapter to pin 14 in the PET-socket.).

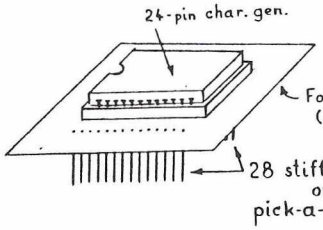


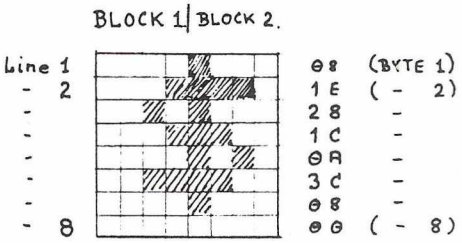
Figure 2.

So if you are satisfied with using ready-made character generators you can make the adapter and then stop. If you want to modify one or maybe design your own read on.

I teach on a public school where we have 5 old PETs. The problem was that we wanted to have some special Danish letters and the language teachers would like to have some German letters too. We could buy a 24-pin char.gen. for the newer PETs with Danish letters only, so I started finding out how the screen characters were made. I made the adapter first to find out if it would work with one of the newer standard Commodore char.gen.'s. It did, so my next task was to find out how the characters are stored in the ROM. After that I substituted some of the characters with my own and burned them into a 2516 or 2716 EPROM - and we had what we wanted. So if you can get to an EPROM-burner you can do the same.

Let's say you want to have a real £-sign instead of the $-sign. The $-sign is made in an 8X8 matrix like in figure 3.

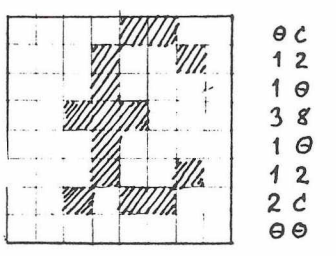
Figure 3.



In the character generator it is stored in the form of 8 successive bytes. The first byte represents the upper line and byte no.8 the lowest.

Look at the individual lines as consisting of two blocks of 4 "cell's" - each "cell" representing a binary digit. If you look at line 1, block 1 is: 0000 and block 2 is: 1000. Translated into HEX that is 0 and 8 so line 1 is stored as 08. Line 2 is 0001 and 1110.

Figure 4.



That's 1E ... and so on. The whole character is stored as: 08 1E 28 1C OA 3C 08 00.

What does a £-sign look like in an 8 by 8 matrix? You may have your own idea about it, but let's say you want to look like this: (Fig. 4.) After the same rules as before it has to be stored as: OC 12 10 38 10 12 2C 00

Now we are able to design all the characters we like.

To replace the $ we have to find it first, so let's look at where in ROM it's stored.

The characters generator uses a total of 2K. It is divided in four equal parts. The first fourth (from 0000HEX to 0200HEX) contains the characters you see on the screen when you turn on your PET and presses one key. The next fourth contains all the shifted characters, the graphics. The last half is what you use if you POKE59468,14 - small letters and (shifted) capitals.

In the following all addresses are in HEX.

If f.ex. we take the capital letter 'A', it is stored in the 8 memory cells starting with 0008. Shifted 'A' (=spade) is 0200 higher and starts at 0208. 'a' starts at 0408 and you have 'A' again starting from 0608.

The $ starts at 0120 but you find it again from 0520 (POKE 59468,14-mode).

To burn your own EPROM char.gen. you load the data from the CBM char.gen. into the EPROM-burners RAM-area and change the addresses mentioned above to what you decide is a £-sign. Burn the EPROM. If you have a 3- or 4000-series PET just plug it in. If you have a 2000-series use the adapter.

If you prefer to keep the $ and sacrifice some other characters, here are some other possibilities:

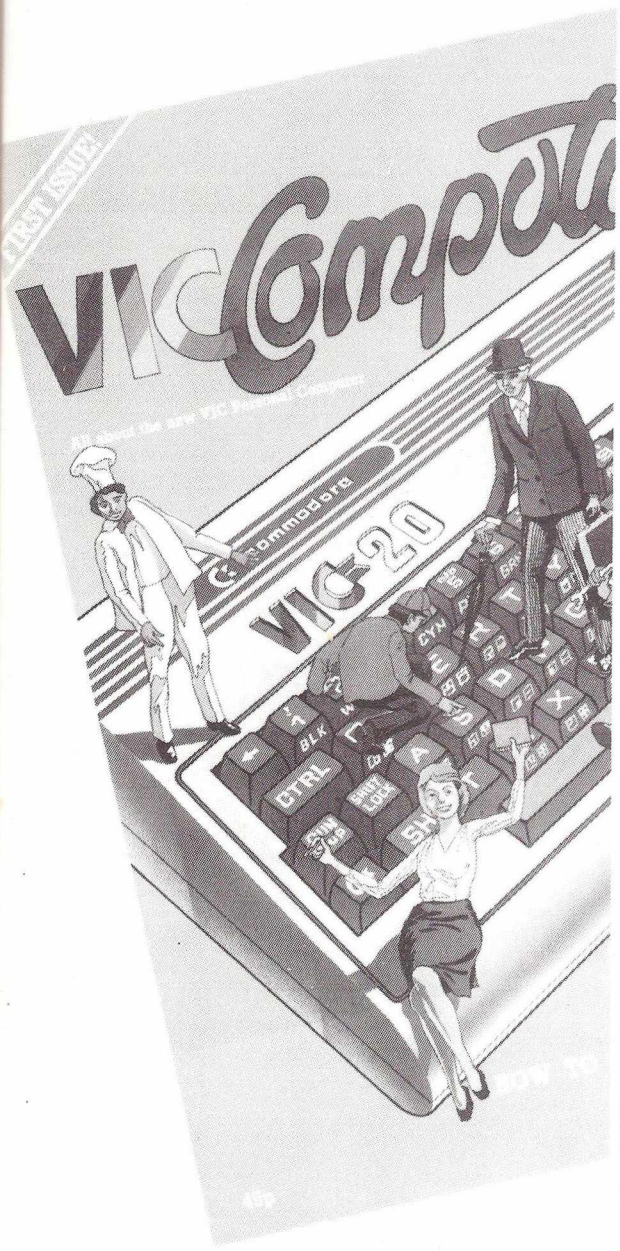
[: 00D8 and 04D8
\ : 00E0 and 04E0
] : 00E8 and 04E8
← : 00F8 and 04F8

The order in which the characters are stored in the ROM is:

First come the ASCII-characters from 40 to 5F. That is @,A,B,C,---- X,Y,Z, , , , . Then come the ASCII-characters from 20 to 3F which include punctuation marks, space, the ciphers etc.

VIC Computing

The 'how-to' magazine all about Commodore's VIC computer



VIC Computing is a great new magazine for users of the VIC. Each issue is packed with valuable programming hints, software reviews, 'how-to' articles and program listings.

You don't have to be an expert to enjoy *VIC Computing.* It is written in straightforward English for beginners – not computer experts.

Features in the first colour packed issue included: "Anyone can Program", an article to teach you to program in one hour; "But What Can It Do?", an introduction to VIC's capabilities; "Expansion of the VIC", a guided tour of its add-on capabilities; "Using Graphics" covering programming in colour; "Converting Software for VIC" – how to convert PET programs; "VIC Sound", data sheet on sound generation; plus "Dear VIC", "Beginners Queries" and "VIC hints".

It costs just £6 a year to subscribe to *VIC Computing.* Can you afford not to?

To: Printout Publications, PO Box 48, Newbury RG16 0BD

Please enter my subscription to VIC Computing.
I enclose a cheque/Postal Order for [ ] £6 UK [ ] £IR8.50 Eire
[ ] £9 Europe [ ] $20 USA Surface [ ] $30 USA Air
[ ] £9 Rest of World Surface [ ] £16 Rest of World Air
or Charge my Access/Mastercharge/Eurocard or Barclaycard/Visa card
No: .....

NAME: .....

ADDRESS: .....

.....

POSTCODE: .....

VIC Computing

Basic Programming

"CROSS REFERENCE" CONVERSION FOR COMPUTHINK DISK DRIVES

Another fine program contributed by Jim Butterfield in CPUCN Vol 2 No 8 prompted me to tweak it here and there in order to get it to work with Computhink drives. Other users may like to carry out this program conversion; the listings given here now work with the following system:
8K PET 2001, Old ROMs
32K EXPANDAMEM extension memory
COMPUTHINK 400K (single-sided dual-density) drives
DIABLO 1640 daisy wheel printer

As Computhink users will already know, it is not possible to read a program off disk as a file. Neither is it possible to use the GET instruction to read in one character at a time. However, a program in memory can be converted into a file on disk, the records being written as decimal strings, and the file can then be read back one string at a time.
First, save the two programs CROSS REF and PROGFILE on the same disk, and take a protection copy.
See figures a and b

PROGFILE is a short program which has to be linked to the progam which is to be cross-referenced, therfore the latter must not contain any lines numbered above 59999.

If you examine the Basic text area from $0401 upwards, either by the $MEM command, or by MLM or Petsoft's PDAS, you will see that the end of a Basic line is signalled by the hex byte 00. The end of program is indicated by three zero bytes: 00 00 00. But, when PROGFILE is linked to the main program, the last two zero bytes of the latter are overwritten; they are, in fact, the low and high bytes of the link for the next line of Basic. In line 60210 you will note that once this section of program is run, the zero bytes are POKEd back in again. The file created terminates at the end of the main program (the one to be cross-referenced), thus lines 60000 upwards are excluded from the file, and PROGFILE is self-destructed by the zero POKEs.

A few words of explanation about PROGFILE:

60040 sets the index variable I to the start of the Basic text area ($0401)

60050 commences PEEKing from 1025 upwards until a zero byte is detected, which signals the end of a Basic program line

60060 skips the next two bytes, which are the low and high bytes of the link address to the next line of Basic, and tests to see if the next byte is $60 (the low byte for line number 60000). If not $60 (96 decimal) go back and

Figure a

```
PROGRAM:                CROSS REF


 100 DIMA$(15),B$(3),X$(500),C(255)
 110 PRINT"[CLR]"
 120 Q$=CHR$(34):S$="     ":B$(1)=Q$:B$(3)=CHR$(58):E=-1
 130 INPUT"VARIABLES OR LINES";Z$:C2=5:IFASC(Z$)=76THENC2=6
 140 FORJ=1TO255:C(J)=4:NEXTJ:FORJ=48TO57:C(J)=6:NEXTJ
 150 IFC2=5THENFORJ=65TO90:C(J)=5:NEXTJ:FORJ=36TO38:C(J)=7:NEXTJ:C(40)=8
 160 C(34)=1:C(143)=2:C(131)=3
 165 PRINT"[CUD][CUD]"
 170 INPUT"PROGRAM NAME";P$:$O,1,"R",P$,DI$
 174 PRINT"[CUD][CUD]"
 175 REM DISCARD LINK ADDRESS
 180 $R,DR$:IFDR$="EOF"GOTO530
 185 DR$="":$R,DR$:DR$=""
 190 IFB=0GOTO240
 200 PRINTL$;:K=X:FORJ=BTO1STEP-1:PRINT" ";A$(J);:X$=A$(J)
 206 X$=X$+L$
 210 IFX$(K)>=X$THENX$(K+J)=X$(K):K=K-1:GOTO210
 220 X$(K+J)=X$:NEXTJ:X=X+B:PRINT:B=0
 230 REM GET NEXT LINE, TEST END
 240 $R,DR$:A$=DR$:$R,DR$:B$=DR$
 245 L=VAL(A$):A=VAL(B$):IFL+A=0GOTO530
 250 REM COMPUTE LINE NUMBER
 280 C=C2:C1=-1:L=A*256+L:L$=STR$(L):IFLEN(L$)<6THENL$=LEFT$(S$,6-LEN(L$))+L$
 290 REM GET BASIC STUFF
 300 IFD=1THEND=0:A=36:GOTO310
 302 IFE>-1THENA=E:E=-1:GOTO310
 304 $R,DR$:A$=DR$:A=VAL(A$)
 305 IFA<>36GOTO310
 306 D=1
 307 $R,DR$:A$=DR$:A=VAL(A$)
 308 IFA<65ORA>90THENE=A:GOTO300
 309 D=0:GOTO307
 310 C9=C(A):IFC9>C1GOTO380
 320 IFC2=6ANDLEN(M$)<5THENM$=" "+M$:GOTO320
 326 K=0:IFB=0GOTO360
 330 FORJ=1TOB:IFA$(J)=M$GOTO370

PAGE NO: 02

 340 IFA$(J)<M$THENNEXTJ:K=B:GOTO360
 350 FORK=BTOJSTEP-1:A$(K+1)=A$(K):NEXTK
 360 B=B+1:A$(K+1)=M$
 370 C=C2:C1=-1:M$=""
 380 IFC2=5GOTO420
 390 IFA=137ORA=138ORA=141ORA=167THENC=6:GOTO470
 400 IFA=44ORA=32GOTO470
 410 IFC9<>6THENC=9:GOTO470
 420 IFC9=CTHENC=-1:C1=4
 430 IFC>6GOTO470
 440 IFC<0ANDC9>C1ANDC9>6THENC1=C9:GOTO460
 450 IFC2=5THENIFLEN(M$)>2ORC>0GOTO470
 460 M$=M$+CHR$(A)
 470 ONC9+1GOTO180,480,480,480:GOTO300
 480 B$=B$(C9):C$=""
 490 $R,DR$:A$=DR$:N=VAL(A$):IFN=0GOTO180
 500 IFCHR$(N)=B$GOTO300
 510 IFN<>34GOTO490
 520 A$=B$:B$=C$:C$=A$:GOTO490
 530 $C:INPUT"PRINTER";Z$
 535 PRINT"[CUD][CUD]"
 540 C=3:Z=6:IFASC(Z$)=89THENC=4:Z=12
 550 OPEN2,C,1:PRINT#2:PRINT#2,"CROSS REFERENCE - PROGRAM: ";P$
 560 X$="":FORJ=1TOX:A$=X$(J)
 565 IFC2=6THENK=6:GOTO580
 570 FORK=1TOLEN(A$):IFMID$(A$,K,1)<>" "THENNEXTK:STOP
 580 B$=LEFT$(A$,K-1):C$=MID$(A$,K+1):IFX$=B$GOTO600
 590 PRINT#2:Y=0:X$=B$:PRINT#2,X$;LEFT$(S$,6-LEN(X$));
 600 Y=Y+1:IFY<ZGOTO620
 610 Y=1:PRINT#2:PRINT#2,S$;
 620 PRINT#2,LEFT$(S$,6-LEN(C$));C$;
 630 NEXTJ:PRINT#2:CLOSE2:END
```

Educational Supplement

A great deal of interest has been aroused by the (relatively) recent introduction of COMAL onto the scene. Lying somewhere between BASIC and PASCAL, in the sense that it is as easy to learn as BASIC, but has some of the more useful aspects of PASCAL, like long variable names, a more defined structure, and so on.

Two rules were followed in defining COMAL :-

1) Say what you mean, simply and directly.

2) Choose variable names that won't be confused.

These were taken from the book The Elements of Programming, by Kerninghan and Plauger.

To learn a little more about COMAL, quite a lot of this month's supplement is devoted to it. All the material is taken from the newsletter put together by the COMAL Users Group in Wisconsin, a group who have been extremely active in getting COMAL off the ground in the States. Thanks to Len Lindsay for sending me the information, with the promise of a lot more to come, and I hope you find it useful.

If you've developed any of your own programs in COMAL please send them along, as both Len and myself would be interested in seeing them. This educational section is the ideal medium for publishing these programs, so let's make use of it. Len's promised to send me copies of anything they produce, so between us let's give COMAL the support it deserves.

BASIC 2 version

As a footnote to the COMAL saga, work is being done in this country to produce a BASIC 2 version, thus giving even wider market coverage. When this has been done, we'll let you know. My latest information tells me we're 98% of the way there : details will appear as and when. Len in the States is working on his own COMAL Users Handbook, superior to the one originally produced, and again we'll let you know when that appears.

A few notes on some of the other articles in this section. There are a large number of colleges producing courses for beginners to the world of microcomputers, and many of these colleges are part of our educational workshop scheme. One such college is Bradford College, and inside we find details of their courses for which we're still in time.

One of the great advantages of the Commodore printers is their ability to produce programmable characters not accesible on the PET's keyboard. Of particular usefulness to schools and colleges would be the ability to have Greek characters, as used in maths, physics, chemistry etc. Well, here we present the answer, in an article from Dr. Heathcote of University College, Cardiff. As a footnote to Dr. Heathcote : I know it's taken a long time to get this printed, but in moving into this job I inherited various filing cabinets etc., and in our recent move to a new building (more of that next month!) some more material came to light, and the good doctor's article was one of them! So thank you, and I'm sorry it took so long.

Further to a reader's request for a program that produces formatted listings, here we have the answer. E.g. a cursor down appears as crd , and so on. All kinds of features are also built into the program, which is an enhancement to one that appeared in an earlier newsletter, including such things as title and date, page numbers, right justification of line numbers, and prints a listing as it would appear on the screen, i.e. upper and lower case is upper and lower case. As with any program, I suggest you save it before attempting to run it. Disasters have happened!

Next we come to a couple of articles by Nick Green, Special Projects Manager in Commodore, and with a major interest in PETs in education. The first is a report of a visit to a comprehensive school using PETs, and the failures and successes they've experienced. Secondly a visit to a major booksellers, to take a look at the kind of aids available.

Programmable Sound Generator

The next article is entitled "Connecting the General Instruments AY-3-8910 Programmable Sound Generator to the 6502/6300 bus". The what ?! To quote from the article, the AY-3-8910 sound generator is .... capable of generating three simultaneous tones, each of which can be separately controlled in amplitude and/or mixed with noise to produce a wide range of sound effects. Entirely digitally controlled, this makes it very suitable for use with a micro processor. So now you know! I've included it with this section because I think it would make an interesting project for a school running PETs. If any of you attempt it, I'd appreciate a report of how you get on, and I'd reproduce that in a future newsletter : encouragement for others is always vital.

The COMAL section appears next, and to sum up for the month we have the latest additions to the educational workshops list. I know I said in September's educational editorial that there were no amendments for that particular month, and of course there were. Printing deadlines strike again!

VIC

As I request in the main editorial for material, I'd like to put in a request here for any VIC-based material you find out. The main purpose of a magazine about microcomputers ought to be to pass on information about that microcomputer. In other words, to help repay your subscription in terms of information gleaned

and thus time saved in your own program development. So, whatever you find out, pass it on! If it saves someone time, it saves them money.

I.P.U.G., the Independent PET Users Group, who have a couple of VICs, are doing all kinds of interesting work on the beast. Rumours abound about a 40 column machine, but don't write in yet. We'll let you know! Anything else they produce will also, of course, be passed on.

One thing I can tell you about I.P.U.G. is that they're forming a VIC Users Group in the not too distant future. Again, as soon as I find out what kind of form this will be taking, I'll let you know.

So that's it then, possibly the two most important developments in education in recent times : the VIC and COMAL. The former for inexpensive access to the world of microcomputing, and the latter for an easy introduction to producing structured, readable programs. Whether the two will ever be compatible we just don't know, but that's something for the future.

Whatever is happening, via the "yellow pages" we'll keep you in touch!

Megapalm Software

Megapalm Limited specialises in the analysis, design and installation of computer based business systems.

They have been dealers in microcomputer equipment since 1979 and the company exhibited at the '2nd Personal Computers World Show' in London in that year. Here are a few extracts from their brochure:

Megapalm Limited operates a local microcomputer bureau service. We are situated in the North West of England and able to supply complete computer systems (hardware and software) in Lancashire, Cumbria and West Yorkshire. We have clients in such places as...

Carnforth
Caton
Dalton in Furness
Galgate
Kendal
Lancaster
Liverpool
Morecambe
Piling
Salford
Sedbergh
Ulverston.

Both in the North West of England and throughout Great Britain, we are able to supply Megapalm Software. You may contact us either directly or through your local microcomputer dealer.

Our Software Editor.

All Megapalm Software has been written by or under the supervision of J.R.Mace B.A., M.Sc, F.C.A.
Roger Mace is a Senior Lecturer in Accounting and Finance at the University of Lancaster.

He is author of several articles and other publications including the book, 'Management Information and the Computer', (Prentice Hall 1974).

He is editor of the 'Lancaster Journal of Computer Applications in Finance and Accounting'. (Details from Department of Accounting and Finance, University of Lancaster, Lancaster LA1 4YX).

One of his forthcoming publications (1982) is a book provisionally entitled 'Basic Program Design for Accountants'. (Details from Philip Allan Publishers Ltd., Market Place Deddington, Oxford OX5 4SE.)

Features of all our Software (subject to change without notice)

1. All Megapalm Software for the Commodore 8000 Series of Microcomputers is supplied in compiled form to obtain the speed and sophistication of DTL BASIC. (DTL BASIC is a Basic Compiler by Drive Technology. The main advantages of DTL BASIC are that compiled programs run much faster than on the interpreter, and occupy less store than uncompiled programs).

2. All Megapalm Software is believed to be cost-effective and user friendly.

3. End-user prices described in this leaflet include introductory on-site training of a user's existing personnel.

4. All Megapalm Software is interactive and is personalised for each end-user.

5. Hardware for which Megapalm Software is designed can also be used for other purposes, including word processing.

6. Pre-installation consultancy on system design is available from Megapalm Limited on request.

7. Dealer Demonstration packs are available.

Established Megapalm Software

G-PAC
(General-Purpose Accounting paCk)
Programs in the G-PAC suite include those for:-

1. *Invoicing* with optional link to maintenance of *stock records* and to *sales ledger* maintenance.
2. *purchase ledger* maintenance with optional link to automatic *cheque writing.*
3. *nominal ledger,* trial balance and production of 'on demand' *financial accounts.*

A user may elect to integrate these applications with one another in almost any combination and with simultaneous allocation of costs and revenues to 'cost centres' if required.

Features of G-PAC Suite.

1. Full audit trails and elaborate programmed controls to ensure that invalid data will normally be rejected by the system.
2. Over 3100 transactions recorded on each disk.
3. Up to 99 financial codes, any of which may be analysed to greater depth by use of detailed account codes, up to a maximum of 250 detailed codes (and overall maximum of about 290 'live' codes on each disk, with the remainder left untitled).

Typcial end-user price....
around £1000 + VAT

Property sales and mailing list service for estate agents.

* Large capacity - 1800 properties and 1800 applicants on a single data disk.
* Data recorded for each property apart from its address and the asking price includes.....

date of entry and selling status document file reference
property type - semi semi-det etc.
property type - hse bung flat etc.
age and state of modernisation construction material
location - village or town, n or s
location - distance from centre
location - detailed area code size of garden
number of garages, beds, baths, reception rooms & habitable rooms
type of central heating
'granny-flat' potential
'with land'
'scope for extending'

amenity catchment area.
* Data recorded for each applicant includes the applicant's attitude to each of the above factors, as well as his name and address and the price of properties in which he might be interested.
* Printed output includes... full or selective listings of property.
* Judgement or standardised matching procedures for any combinations of the available criteria.
End-user price ..... £800 + VAT

Courses at Bradford College

Their leaflet describes these courses as "intended to be part of an educational service to the community of Bradford and as such are aimed at individuals and at commercial, industrial and other educational institutions". If you belong to an industrial or commercial concern, then they can discuss courses for employees tailores specifically to their needs : all you have to do is write to the college.

As part of their commitment to the Commodore Educational Workshop scheme, the college provide on every Friday evening up until 8.30 p.m. (but not during college holidays), an opportunity to get a hands-on session with computing (specifically PETs). Options to be presented will be exploring available software and its suitability for an educational environment, amend and develop software, gain experience in the use of computer hardware, and develop and discuss ideas on using computer based learning techniques with other teachers. This is free for teachers employed within the Bradford Metropolitan District, and other teachers have to become an associate member of the college :this costs just 10 pounds.

As well as holding a number of two and three year courses, there are a couple of evening classes which could be of interest to those of you in or near the area. One on information and word processing : telephone Mrs. B. N. Holmes (Bradford 34844, extension 223), who is the course tutor, for details. The other is on computer based learning : telephone Mr. F.P. Marrow (same number, extension 163) for details of this one. Both these courses start in January next year, so we're in plenty of time.

If any other college would like to pass onto me details of courses they're holding, I'd be pleased to publish them.

Educational Workshops

Here are the latest additions to the list of educational workshops.

Establishment	PETs	Courses
Mr. B. Jones Slough College of Higher Education Wellington Street Slough Berkshire	8	Yes
Mr. R. Broadley Dept. of Social Studies Trent Polytechnic Burton Street Nottingham	15	Yes
Mr. R. A. Clarke Computer Centre Birmingham Polytechnic Franchise Street Birmingham B42 2SV	4	Yes
Mr. J.R. Park Merchant Taylors School Liverpool Road Crosby Liverpool L23 0QP	9	Yes
ASCENTS Shiremoor Teachers Centre Park Estate Shiremoor Newcastle Upon Tyne NE2 1HN	2	Yes
Dr. J.P. Chester Dept. of Applied Science Moylish Park Limerick Ireland	20	Help and Advice
Mr. R.C. Bennett Whitland Grammar School North Road Whitland Dyfed Wales	3	Help and Advice
Mr. D. Voulgaridou Vas. Georgiou 16 Thessalonika Greece	5	Possibly
R.E. Smith Mayfield C.E. Middle School St. Vincents Road Ryde Isle of Wight	1	No
Dr. K.W. Glasson 11 Lancaster Cottages Lancaster Park Richmond Surrey	1	Help and Advice esp. remedial
Mr. J. Aspinall West Midlands College (H.E.) Gorway Walsall	12	Yes
J. Battisti City of Liverpool College (H.E.) Liverpool Road Prescot Merseyside L34 1NR	3	Yes
Mr. W.R. Houston Peterlee Technical College Peterlee Co. Durham	13	Yes
Mr. R. Bird King Edward's School Witley Godalming Surrey GU8 5SG	3	Yes

And there is also one correction to an entry: apologies to all concerned.

J.I. Meardon Adams School Lowe Hill Wem Shropshire SY4 5UB	3	Internal

Comal Introduction

I make no apologies for reproducing here the almost complete contents of the latest information package from the COMAL Users Group in the States. This is a body of people who are very keen to get COMAL well and truly established as THE programming language of the '80s, and probably beyond as well! Read on ...

Anyone using COMAL today, we consider a COMAL pioneer. We are breaking the way for COMAL. So, let's make it easier on ourselves. We are sponsoring a COMAL Program Exchange - featuring USER GROUP DISKS. The idea behind this is simple. It provides a way for each COMAL pioneer to share their programs, and it provides a way for a brand new COMAL pioneer to get some already working COMAL programs to analyze, to figure out how things work. Simply send us a disk (please mail it in Floppy Armour, or stiff cardboard for protection) of your substantial COMAL programs (also include any nifty procedures as well - listed to disk, ready to merge into another COMAL pioneers program). In return, we will send you a COMAL USER DISK, of user submitted COMAL programs - FREE of charge. Or, if you don't have any suabstantial programs to submit, you still can take advantage of this exchange by ordering the disks you want (see our order form). Please, no copyrighted programs accepted for the exchange. We are contributing our order processing system to User Group Disk #2. We also would like our User Group Disks to contain useful procedures (routines) that accomplish specific tasks. We are contributing some procedures that can be used with the STARWRITER printer. One will double strike any string passed to it. The other will bold face any string. Both procedures are handy for anyone producing reports or printouts on a STARWRITER. We hope COMAL pioneers with other types of printers will send in their routines to take advantage of each printers special features.

We will pay you to talk about COMAL

COMAL is in dire need of assistance. It is battling BASIC (already firmly entrenched) and PASCAL, the latest language craze. We hope you agree that it is better than both, but the better languages don't necessarily become popular. It takes wide spread support. We hope that you are introducing COMAL to your friends. And to thank you for spreading the word, we will give you $5.00 for anyone who orders a STARTER KIT and say they heard about it from you. We wish we could do more to help you, but we can barely pay our present printing and postage bills as it is. You may notice that our Starter Kit now includes a FREE subscription to our COMAL Newsletter as well as FREE updates to both the COMAL MANUAL and HANDBOOK. Since COMAL is just getting started, these three additions are sorely needed. We are including these additions with your previous purchase of the STARTER KIT at no extra charge to you.

A condensed order form and price list is enclosed. Make as many copies as you wish. Notice the new prices and expanded KIT. And remember - $5.00 for each COMAL convert, the more COMAL pioneers the better. Look out BASIC, COMAL is coming!

COMAL USERS GROUP,
5501 GROVELAND TERRACE,
MADISON WI 53716

What is COMAL?

If this has crossed your mind, our group can help. We are gathering information on this new programming language. This information is available to anyone who requests it and includes a large 35 cents SASE. But as of this date, information on COMAL is extremely hard to come by. Thus far our search for information on COMAL has been very interesting. We contacted the PROGRAMMING LANGUAGES Special Interest Group of ACM - their reply was "Sorry - this is the first I've heard of COMAL. "We talked with almost every Publisher at the National Computer Conference - none had any books on COMAL, none even had heard of it. One publisher told us that it would be foolish to publish a book on a language until there was a following of some size - and from a business standpoint, we must certainly agree. So, the need for a COMAL central information source is genuine. We hope we can fill this need. If you are aware of any information on COMAL that we do not mention, please let us know so we can add it to our sources (if we missed an article, send us a copy for our files). Also please let us know if there are any errors in our information here, so that they may be corrected. Hopefully the release of this information package will turn up leads to better and more complete sources for our next update.

This information was processed with WORD PRO 4 on a CBM 8032. We also have access to a CP/M system, and just received METANIC COMAL-80 for CP/M.

The information presented here has been reviewed and is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies. The material presented is for information purposes only and is subject to change without notice.

* PET and CBM are trademarks of Commodore Business Machines
* CP/M is trademark of Digital Research
* METANIC COMAL-80 is trademark of Metanic Aps
* COMAL USERS GROUP is trademark of COMAL Users Group
* COMAL CATALYST is trademark of COMAL Users Group
* FLOPPY ARMOUR is trademark of Square 1
* Z-80 is trademark of Zilog
* APPLE is trademark of Apple Computer Co
* TRS-80 is trademark of Radio Shack
* SOFTCARD is trademark of Microsoft
* STARWRITER may be trademark of C Itoh
* WORD PRO is trademark of Professional Software

Now, a little background on COMAL. Several years ago, Borge Christensen and Benedict Lofstedt designed some extensions to BASIC to make programs more readable and much safer. Then he and his associates in Denmark (including Knud Christensen and Per Christensen) modified Data General Corporations Extended BASIC interpreter. They called it COMAL (COMmon Algorithmic Language). But, events before that were responsible for the project itself. In 1972, the States Training College in Tonder, DENMARK, got a NOVA minicomputer. At that time, BASIC was the language being used for teaching new students programming. It was easy to learn and worked just fine at first. But as the short programs grew bigger, errors became more frequent, and it took quite some time to find out just where the student had made a mistake. The programs were hard to read. Borge Christensen and his associates became dissatisfied with BASIC. There were two main causes of this dissatisfaction. BASIC variable names were too short to relay information as to what they represented and the many GOTO's made it hard to identify a programs tasks. The IF .. THEN was a prime offender in this second area. When an IF was TRUE, you often had to go **SOMEWHERE ELSE,** rather than immediately executing the statements needed at that point.

Two simple and fundamental rules of programming from the book THE ELEMENTS OF PROGRAMMING STYLE by Kerninghan and Plauger are virtually impossible with BASIC:

1) Say what you mean, simply and directly.
2) Choose variable names that won't be confused.

However, there are many GOOD things about BASIC, such as an INTERACTIVE MODE, DYNAMIC EDITOR, and easy to use I/O STATEMENTS. And is definitely easy to learn. So, the answer was to design a language that kept these GOOD points from BASIC, but that also had long variable names, and a more defined structure. PASCAL was the model chosen for that. The result was COMAL, a language that should be ideal for both the beginning student, since it is easy to learn and use (like BASIC), and for the advanced, since it is structured and powerful (like PASCAL).

Now a quick taste of what COMAL is all about. For more complete and detailed information, please refer to our COMAL STARTER KIT. COMAL automatically indents the body of each structure to make the program more readable. The examples listed below illustrate the indentation provided.

IF ... THEN ... ELIF ... ELSE ... ENDIF

The IF sets up a condition that is either TRUE or FALSE. IF the expression evaluates to TRUE, the statements following the **THEN** are executed. Otherwise, the statements after the **ELSE** are executed. The ELSE and its statements are optional, and the ENDIF is not always needed. ELIF is short for ELSE IF. See example a.

The CASE sets up a multiple choice type of situation. The value of the expression is compared to each value after each WHEN (as many WHENs may be used as you wish). When it compares TRUE, all statements after that WHEN upto the next WHEN (or OTHERWISE) are executed. If none of the WHEN comparisions are TRUE, the statements immediately after the OTHERWISE are executed. Program execution then continues with the line following ENDCASE. This is similar to the ON ... GOTO of BASIC, however, it is much easier to follow and read, since the statements to be executed follow each WHEN, while the ON GOTO of BASIC sends you all over the program with the chain of GOTO's. Another advantage, is that you may include multiple expressions following a WHEN (thus you do not need to repeat a set of statements to be executed on two different sets of criteria), see example b.

This is one way to set up a loop in COMAL, and it functions just like in BASIC. STEP is optional, as in BASIC, and if left out defaults to 1. The DO and NEXT are also optional depending on the situation. Once the value of variable name exceeds the value of the expression after the TO, the loop is finished and the program continues execution with the first statement following the loop. NOTE: the condition is compared first, and the statements executed afterwards. Two examples (notice that the first is a one line example) are given, as in figure c.

example a.

```
IF <expression> THEN
  <statements>
ELSE
  <statements>
ENDIF
```

```
IF CHOICE$ = "HELP" THEN
   EXEC INSTRUCTIONS
ELSE
   PRINT "THANK YOU"
   LINECOUNT:+1
ENDIF
```

CASE ... OF ... WHEN ... OTHERWISE ... ENDCASE

```
CASE <expression> OF
WHEN <expressionsA>
  <statementsA>
WHEN <expressionsB>
  <statementsB>
OTHERWISE
  <statements>
ENDCASE
```

```
CASE LINECOUNT OF
WHEN 0
  EXEC INITIALIZE
WHEN 1, 2, 3, 4
  PRINT TEXT$
OTHERWISE
  EXEC NEXTPAGE
ENDCASE
```

example b.

FOR ... TO ... STEP ... DO ... NEXT

```
FOR <variable name> = <expression> TO <expression> STEP <expression> DO
  <statements>
NEXT <variable name>
```

Figure c.

WHILE ... DO ... ENDWHILE

```
WHILE <expression> DO
  <statements>
ENDWHILE
```

```
WHILE ERRORS=0 DO
  EXEC FINDIT
  EXEC SORTIT
ENDWHILE
```

This structure sets up a loop, similar to the FOR ... NEXT loop. With this loop, the condition comes at the beginning (after the WHILE), and the loop is executed only if the expression evaluates to TRUE. It is very handy since it will keep executing as long as the initial condition is met.

REPEAT ... UNTIL

Once you use this structure, you will wonder how you ever did without it. It is very similar to the DO .. WHILE structure, except that the condition occurs after the loop is executed. It is very handy for reading data from both disk files, and data statements. Its basic syntax is:

REPEAT
  statements
UNTIL expression

Here are two examples to show how easy it makes programming. The first keeps reading records from a sequential file on disk until it reaches the End Of File (EOF). The second keeps reading data from data statements until it comes to an End Of Data (EOD), fig d

PROC ... ENDPROC and EXEC

PROC is an abbreviation for procedure, and is similar to the subroutine of BASIC, though much more advanced. One advantage immediately noticed, is that each procedure has a name, whereas the subroutine of BASIC was called by line number. The procedure's statements are coded between the PROC and ENDPROC, and are only executed when called with the EXEC statement. Otherwise they are skipped over and thus they can be placed anywhere within the program (you can't accidently fall into a procedure with COMAL). The PROC and EXEC look like, fig e.

But this just touches the tip of the PROCEDURE iceberg. COMAL procedures allow both local and global variables, as well as true parameter passing. You can declare a procedure CLOSED, in which all its variables will become LOCAL, and will not be confused with variables with the same name in the main program.This makes it easy to write general procedures, which can be appended onto future programs without worrying about conflicts with variable names.

Figure d.

```
REPEAT                          REPEAT
  READ FILE 1 : ITEM$             READ TEXT$
  PRINT ITEM$                     PRINT TEXT$
UNTIL EOF(1)                    UNTIL EOD
```

Figure e.

```
PROC <procedure name>           PROC ERRORPRINT
  <statements>                     PRINT "AN ERROR HAS OCCURRED"
ENDPROC <procedure name>           PRINT "THE ERROR WAS ";ERROR
                                ENDPROC ERRORPRINT
-------------------------------------------------------------------
EXEC <procedure name>           EXEC ERRORPRINT
```

Figure f.

```
OPEN 1, "TESTFILE", READ        OPEN 2, "HIGHSCORE", WRITE
  READ FILE 1: NAME$              WRITE FILE 2: SCORE$
  READ FILE 1: ADDRESS$           WRITE FILE 2: NAME$
  READ FILE 1: CITY$              WRITE FILE 2: "HIGH SCORE"
CLOSE 1                         CLOSE 2
```

The ability to easily merge program segments is built into the STANDARD COMAL. Line numbers are included in the listings, simply to give you a point of reference. There is no COMAL statement that references a LINE NUMBER. Of course, editing commands allow you to use the line numbers (such as with LIST, EDIT, DEL (delete), AUTO (automatic line numbering), and RENUM (renumber).

And disk files are very easy to use. You simply OPEN a file for either a READ or a WRITE. Then simply READ or WRITE to it, CLOSEing it when you are done. And several files may be open simultaneously. For example, fig f shows how to read and write with sequential files.

It is also easy to "tack on" some information to the end of a sequential file that already exists. Simply use the keyword APPEND instead of WRITE in the OPEN statement (OPEN 5, "FILENAME", APPEND). COMAL also supports RANDOM, direct access files.

Since this is only supposed to be a brief introduction, we will stop here. For comeplete details and information, please refer to our COMAL STARTER KIT. We highly recommend COMAL as the language to learn before BASIC. COMAL is more structured and allows you to write better programs faster. It is also very easy to learn.

CP/M SYSTEMS

METANIC APS has marketed a version of COMAL for two years. It requires the CP/M operating system. The cost is about $233 and involves a license agreement. A special version is available for APPLE with Softcard, and one may soon work with a TRS-80 that has CP/M. To save our CP/M readers the hassle of overseas communication with METANIC APS (based in Denmark), we are distributing information about their system in co-operation with them. We can provide you specific information, including prices, delivery and example program listings. Just send us a Self Addressed Stamped Envelope, asking for **METANIC APS CP/M COMAL INFO.** We also have 50 copies of METANIC APS Syntax Diagrams & Examples booklet (for CP/M systems). We will give these out FREE to anyone sending a large (6 x 8.5 inch minimum size) 52 cent stamped self addressed envelope (while supplies last only). We hope to soon have a copy of this COMAL running on our CP/M system we have access to. Until then, we can only pass on information supplied by Metanic Aps (at least you don't have to write to Denmark for the information).We hope that soon our User Group Exchange Programs will be available on CP/M disks.

We put together an order processing system that we now use to process our STARTER KIT orders. We wrote it in COMAL, and it was com-

pletely working after only 2 evenings, a tribute to the ease of writing in COMAL. We are contributing this system at the start of the second Users Group Disk. The system asks for the customers name, address, and order particulars. It can then print out an invoice, UPS packing slip, Visa/Master Charge slip, business envelope, package mailing label, and filecard. It also adds that customer to a disk file, which can be used as a mailing list. It takes advantage of PRINT USING. It works with our STARWRITER printer, but any friction feed printer should work.

We also would like our User Group Disks to contain procedures (routines) that accomplish specific tasks. We put together some procedures that can be used with the STARWRITER printer. One will double strike any string passed to it. The other will bold face any string. Both procedures are handy for anyone producing reports or printouts on a STARWRITER. We hope COMAL pioneers with other types of printers will send in their routines to take advantage of each printers special features. (One user called to tell us he was going to send us routines for the Epson MX-80).

Special Note: All CBM User Group disks will include a FREE copy of the COMAL interpreter. CP/M disk will **NOT** contain the COMAL interpreter, since it must be licensed from Metanic Aps (approximate license cost is $233).

MORE ABOUT THE TWO COMAL-80 INTERPRETERS

METANIC COMAL-80 is available from Metanic Aps in Denmark. They sent us 50 copies of their booklet, SYNTAX DIAGRAMS & EXAMPLES. Send us a self addressed stamped (52 cents) envelope (at least 6 by 9 inches) for a free copy while they last. METANIC COMAL-80 is a three pass interpreter for Z-80 microcomputers running under CP/M (including APPLE with SOFTCARD). It allows long variable names as well as LABEL names, and automatically provides program indentation for easy reading. Lines are checked for correct syntax as they are entered. Rejected lines are displayed with the cursor indicating the error. The price is 1700 D.Kr (about 7.3 D.Kr per US Dollar=$233). The manual is 175 D.Kr.($24). For more information on this version of COMAL simply send a Self Addressed Stamped Envelope to the COMAL Users Group - request **METANIC COMAL info.** We are distributing the information, including current prices, in cooperation with METANIC APS, to save our readers from the bother of overseas mail.

CBM COMAL-80 for the PET/CBM was released by Commodore (UK) on May 6, 1981 - but it will not be available directly from Commodore. In fact their newsletter specifically states **"Please do not apply (for a copy of COMAL) directly to us."** Commodore has placed the CBM COMAL-80 interpreter into the public domain, thus allowing distribution by User Groups. **NOTE-we were in error in our original news release and ad by stating that Commodore was releasing a COMAL compiler. It is a three pass interpreter, not a compiler.** Three pass interpreter means that it checks the syntax of each line AS IT IS ENTERED (pass 1), and when you type RUN it first scans the entire program to verify that all structures are correct and replaces branch statements with an absolute address for the jump (pass 2). It then RUNs your program. No extra actions or commands are needed to take advantage of the three passes.

Both Commodore (USA) and Commodore (UK) have OK'ed the distribution of CBM COMAL by the COMAL USERS GROUP.

This is what Commodore (UK) had to say about COMAL in a letter to us: "The excitement in Europe (over COMAL) seems to be growing by the hour and we look forward to America being able to share in the good fortune of having an easy-to-use, structured, planning language at last." However, Commodore (USA) has informed us that they will not be supporting COMAL here in the USA (for the time being). Thus our COMAL USERS GROUP will undertake the support here. We are already developing a COMAL handbook and a COMAL HELP disk, both which should be available shortly. Plus we will distribute copies of the manual provided to us by Commodore (UK) free along with our STARTER KIT. We have started a CBM COMAL program exchange. A copy of the CBM COMAL interpreter will be included FREE on each USER DISK. This should provide an easy way for any PET/CBM user to acquire their own COMAL interpreter, along with some sample programs.

The first release will run on both the CBM 8032 and CBM 4032, with other versions in the works. With any luck, someone will be able to modify the interpreter to work on any PET/CBM. Let us know if you would like to try the conversion. REMEMBER, since COMAL is a 24K machine code program itself, it presently will only work on 32K computers, and currently only with a CBM disk. A special board is being developed in Denmark, and is nearly complete. It will plug into any PET or CBM, providing a greatly expanded COMAL upon power up, with no memory loss. This is how all computers should come, with COMAL in ROM as the language supplied with the machine.

PET & CBM SYSTEMS

The latest version of COMAL-80 to surface is for the 32K PET and CBM computers (with DISK BASIC 4.0). It is an extended version with many added features. And even more enhancements are being added for a COMAL in ROM that will simply plug into ANY PET or CBM, reagardless of memory size or version of BASIC. Prototypes of the COMAL ROM version are currently being tested in Denmark, and should be available soon. We will provide you with more information if you send us a Self Addressed Stamped Envelope - request **CBM COMAL ROM BOARD INFO.** However, the disk version of CBM COMAL is available now. Better yet it is available FREE from the COMAL Users Group. We are including it FREE on each of our CBM User Group disks. (Note: This FREE offer applies only to the CBM disks, **not** to any CP/M COMAL User Group Disks). Do not order COMAL directly from Commodore. They prefer to have a User Group distribute it. To get your FREE copy of CBM COMAL-80 simply order a User Group disk (at $12.95), or if you also would like a manual, handbook, reference card, Help Disk, subscription to our Newsletter, THE COMAL CATALYST, and updates to both the Manual and Handbook, all packaged in a custom padded 3 ring binder, order our deluxe COMAL Starter Kit (a bargain at only $47.50).

MAGAZINE ARTICLES

COMAL: STRUCTURED BASIC by Borge Christensen

Peoples Computers (Recreational Computing), JAN-FEB 1978, pgs 36-40

A very good article including a complete COMAL program listing and sample RUN, authorised by one of the originators of the COMAL language.

AN INTERVIEW WITH DR. CHIP, COMPUTE, April 1981, page 8

This page of rumours and new tips mentions COMAL as soon to be available for the PET.

COMMODORE RELEASES NEW STRUCTURED-BASIC LANGUAGE, COMAL, AS PUBLIC DOMAIN SOFTWARE

CPUCN (the official COMMODORE (UK) newsletter), Vol 3, pages 4-5

This is one of the first news releases reporting the availability of a COMAL language for the PET/CBM. It stated that Commodore COMAL will be placed in the public domain, and will be available through user groups.

PUBLIC DOMAIN COMAL TO STOP USE OF UNSTRUCTURED BASIC IN SCHOOLS

Practical Computing, April 1981, news article, pg 47

This news release reports on the COMAL language available for the Commodore computers.

CBM - COMAL-80 by Borge Christensen

Commodore Info, Vol 1 No 1 Spring 1981, page 3

This article was written in German. We are hoping to have it translated for us soon.

OF INTEREST by Dave Cortesi, Dr. Dobbs Journal, July 1981, pages 38-39

In this new products section, the COMAL Information Package was mentioned, and our address was listed as a resourse.

COMAL NICHT NUR FUR ANFANGER, Computer Journal, January 1981, page 7

This article was written in German. We are hoping to have it translated for us soon.

COMAL-80 A NEW LANGUAGE? Susan Sonderstup & Mogens Pelle

Dr. Dobbs Journal, June 1981, pages 14-15, 46-47

This article traces some of the basic structures of METANIC COMAL-80, a CP/M based COMAL implementation available from Metanic Aps. It also gave a summary of the history of COMAL. The program listing included was not explained and unfortunately was written in Danish.

EDUCATIONAL SUPPLEMENT, Commodore Club News, May 1981, page 1

It is mentioned that COMAL is now available for FREE for the PET/CBM computers.

GUEST COLUMN, Midnite Software Gazette, Summer 1981, page 4

Len Lindsay briefly explains what COMAL is.

STRUCTURED BASIC - FREE, Printout, Issue ?, page 13

An announcement of the availability CBM COMAL as distributed through User Groups.

FREE PET/CBM COMAL, COMPUTE, July 1981, page 166

An announcement of the FREE COMAL information service of the COMAL Users Group.

COMAL AND STRUCTURED LANGUAGES LEAVE PRIMITIVE BASICS BEHIND by Roy Atherton

Practical Computing, June 1981, pages 98-101

A very good article giving an overview of COMAL. COMAL flow charts are explained.

COMAL, Creative Computing, August 1981, page 175

An announcement of the FREE COMAL information service of the COMAL Users Group. (Oops! Our address was incorrect in the announcement.)

USER GROUPS, Midnite Software Gazette, Summer 1981, page 10

Jim Strasma mentions our COMAL Users Group and our User Group Diskettes.

USER'S GROUP and CLUB DIRECTORY, Computer Shopper, August 1981, page 51. The COMAL Users Group is listed under Wisconsin.

PROGRAMMING LANGUAGES FOR BEGINNERS AND THE GLOBAL CHALLANGE by Borge Christensen

Commodore Club News, July 1981, pages 9-11

This is a very interesting article by the creator of the COMAL language. The points he makes should strike home to many people. We are requesting permission to reprint this article in our COMAL NEWSLETTER. He referes to a programming ENVIRONMENT as just as important as the language. Comal was put together as a 3 pass interpreter to allow an interactive language. Below are two good updates from the article:

"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."

"...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."

COMAL Commodore Club News, July 1981, pages 2-3

This editorial by Pete Gerrard includes this beginning: "Comal has hit the world to great acclaim...". Yes, COMAL is about ready to burst onto the scene. He refers to the article by Borge Christensen in the same issue.

BRITAINS SCHOOLS TEACH THE WRONG COMPUTER HABITS by Roy Atherton

New Scientist, May 28, 1981, pages 568-570

This is a good article briefly giving a history of computers and their programming languages, pointing out the fact that BASIC is an old language that does not include the advantages of structured programming. The article includes the following:

"THE GOOD, THE BAD, AND THE UGLY, WAYS OF PROGRAMMING. These two programs, the first in BASIC and the second in COMAL, show the differences between the two languages...The BASIC program illustrates the confusing way of thinking that the language encourages...the COMAL program, on the other hand, uses more logical principles and reads more or less from top to bottom."

"...the Department of Education and Science should tell computer suppliers that from, say, the beginning of 1982, schools and colleges will be discouraged from buying computers without COMAL or other satisfactory structured BASIC. Nothing less than this kind of action will do if we are to put educational computing in Britain on to a firm footing."

Books

COMAL-80 - ADDING STRUCTURE TO BASIC a REPORT by Max Bramer
An interesting and well written report about COMAL from ENGLAND. It includes a careful analysis of the importance of operational environments to the popularity of BASIC.

METANIC COMAL-80 SYNTAX DIAGRAMS & EXAMPLES by Metanic Aps
This booklet provides information about METANIC COMAL-80 (CP/M version) including the correct synax for all commands and statements, as well as 5 sample program listings. It is very well done.

COMAL HANDBOOK by Len Lindsay & COMAL USERS GROUP
Soon to be completed, the handbook's first edition will be for CBM COMAL. It will be distributed in a deluxe padded 3 ring binder, with a built in custom note pad, shipped in a custom made container for safe transit, as part of the COMAL STARTER KIT.

COMMODORE COMAL MANUAL by Commodore Basel (Borge Christensen / Mogens Kjaer)
Reformatted to our COMAL HANDBOOK format, included FREE with each CBM COMAL STARTER KIT.

COMAL PROBLEMLOSNING OG PROGRAMMERING by Borge Christensen
A textbook on COMAL programming written in Danish. It currently is popular in Denmark, and is already partially translated into English. A publisher is still needed for the English version (possibly thru MacMillan).

RESOURCES LIST

* COMAL Users Group, 5501 Groveland Ter, Madison, WI 53716
* Commodore Club News, Commodore Business Machines (UK) Ltd, 818 Leigh Rd, Trading Estate, Slough, Berks, ENGLAND
* COMPUTE!, PO Box 5406, Greensboro, NC 27403
* Computer Shopper, PO Box F, Titusville, FL 32780
* Creativer Computing, PO Box 789-M, Morristown, NJ 07960
* Dr Dobbs Journal, PO Box E, Menlo Park, CA 94025
* Metanic Aps, Kongevejen 177, 2830 Virum, DENMARK
* Midnite Software Gazette, 635 Maple Ct, Mt. Zion, IL 62549
* New Scientist
* Practical Computing, IPC Electrical Electronic Press, Quadrant House, The Quadrant, Sutton, Surrey, SM25AS, ENGLAND
* Printout, PO Box 48, Newbury, RG16 OBD, Berkshire, UK
* Recreational Computing, 1263 El Camino Real, Menlo Park, CA 94025

USER GROUPS

COMAL Users Group, 5501 Groveland Ter, Madison, WI 53716. Please help us update our lists of Resources, User Groups, Articles, and Books. Send us a copy of any article we do not mention that would be of interest to COMAL users. Let us know of companies supporting COMAL with programs.

COMAL PROGRAM EXCHANGE

Anyone using COMAL today, we consider a COMAL pioneer. We are breaking the way for COMAL. So, let's make it easier on ourselves. We are sponsoring a COMAL Program Exchange - featuring USER GROUP DISKS (currently only available for CBM/PET - soon available for CP/M). The idea behind this is simple. It provides a way for each COMAL pioneer to share their programs, and it provides a way for a brand new Comal pioneer to get some already working COMAL programs to analyze, trying to figure out how things work. Simply send us a disk (please mail it in Floppy Armour, of stiff cardboard for protection) of your substantial COMAL programs and/or any nifty procedures (LISTed to disk). In return, we will send you a COMAL USER DISK, or user submitted programs to submit, you still can take advantage of this exchange by ordering the disks you want. Please, no copyrighted programs accepted for the exchange. The first Users Group Disk is included with the CBM COMAL STARTER KIT. It is mainly an introduction to COMAL. The programs are meant primarily as examples of how COMAL works rather than as application programs.

AND IN CONCLUSION...

Thank you for your interest in COMAL. We believe that COMAL is the logical replacement for BASIC. We believe in it so much that we financed the start-up of the COMAL USERS GROUP out of our own pockets (ouch!). We are paying for the entire cost of printing the CBM COMAL manual even though we include it free with each CBM STARTER KIT. And the first issue of our COMAL CATALYST newsletter contains the complete official COMAL-80 proposal for the COMAL-80 NUCLEUS STANDARD (over 30 pages). We want our COMAL STARTER KIT to be a first class product, and thus are going to great efforts to achieve this. We hope that you will like it. Please send us any questions or comments.

We just recently were suprised to read in the Midnite Software Gazette that two people had complained about our CBM COMAL interpreter not being FREE, but costing $13. We were disappointed that they chose to complain to a publication without even calling or writing to us first. We would gladly have clarified the point for them. Our USER GROUP DISKS will cost $13 for either CBM/PET version, or the CP/M version. The COMAL interpreter is not included on the CP/M disk. It is included on the CBM/PET version at no extra charge. Therefore, it is indeed FREE, since the $13 is for the User Group Programs Disk, and even that cost is waived for anyone sending in a disk of their own COMAL programs for our Exchange. We are sorry for any confusion. If there are any other questions - please let us know first, since we should have the answers. We are sorry for the delays that occurred at the beginning, due to the difficulty with a brand new programming language.

Enclosed is a list of our current COMAL offerings. Since we are using quality materials, we think our COMAL HANDBOOK will be a proud addition to your bookshelf. Our STARTER KIT, including both our HANDBOOK and the COMAL MANUAL (complete with FREE update service) along with our HELP disk and User Group disk, compares very favorably with PASCAL for the PET (advertised at $295). Our HANDBOOK will be in a deluxe padded three ring binder (which

Visit to a Comprehensive School

Nick Green

The school in question is St. Peter's and Merrow Grange Comprehensive School, in Guildford, Surrey, which has 830 pupils in all. They have 15 PETs, including one 8032, one disk unit and a number of printers. The finance for all this, some 12,000 pounds, was raised entirely through the support of the pupils' parents : other schools take note! The PETs are available for children in the fourth, fifth and sixth forms.

At the time when this report was written, Surrey county will not spend any more money on Pets. They will also only provide software for RML machines, and although the Inspectors are very helpful, they will help in any way except money. There is also a lack of information or guidance on the micros available : Surrey Resources Centre, for instance, has no micros. Terminals are no good ; a neighbouring school however does have a terminal link from Surrey University. Although the Department of Education's money can be used for RML software, not one school in the country has an RML machine!

Indeed, the Country Inspectors Micro Computers Study Group pushes the RML 380Z, although one inspector did come and borrow a PET from the school to assess it.

St. Peter's and Merrow Grange have taken the lead in Surrey by purchasing first 4 and then 11 machines, going on the basis of computing being a proper classroom activity. They wanted as many children as possible to have hands on experience, and they've subsequently had lots of visits from other schools in the area.

The school is voluntarily maintained by the London Education Authority : therefore there is no liason with the Royal Grammer School who have 4 machines and a computer users group.

In the initial stages there were obviously a number of problems to overcome. When they started, they were told that you couldn't use a PET with a TV interface (said by the Chief Advisor to the Country for Resources). Teaching time again was a problem, as was lack of space. To suddenly encounter that number of PETs meant a good deal of re-organisation!

Keeping Pace

The school is very keen to keep apace with developments, particularly in the Computer Assisted Learning area. CAL is used, but merely as a reinforcement at the moment. The courses currently on offer cover a number of "traditional" subjects, although at a fourth and fifth form level they concentrate mainly on computer awareness, with some 360 pupils taking this course. It's only when we got to the sixth form that we find more specialist programs beginning to appear.

The PETs are not used as a separate department, the emphasis being that it was a computing centre for the whole school. The teachers could come and plan programs with the head of the computing department, Dr. Hancock, and it was found that the only opposition encountered so far has been from the mathematics and chemistry departments. Basically they didn't want to foist PETs on everyone, but they would be in service if the staff so wished.

The main areas of use in the scholastic arena are as follows:- economics, geography, biology, history (not usually encountered in schools), remedial and technical, and physics. To go through each of them in turn, we find the following topics being studied :-

Economics - mainly multiplechoice questions, and survey programs (questions asked of local shoppers).

Geography - map games.

Biology - studying surface area to volume body.

History - As I said, not commonly encountered in schools, but here we find a large number of programs in use. Again multiplechoice questions are being used, for revision before exams. Especially helpful for kids who can't concentrate in class, the theory here is that if they have to beat the computer they'll concentrate better. They are also working on a Civil War Project, a simulation based on a game they came across and re-wrote. The multiplechoice questions will point out immediately how to answer a question, and are useful for explaining how to answer these questions in exams. They see the mistakes immediately. On the simulation front, again the immediate feedback is vital, as well as making the kids think more about the situation at the time. To quote one of the staff "We now know what did happen so with the PET kids can see what could have happened".

Physics - this one is a special case, owing to a teacher being away for a whole term. Consequently the PET is being used here for multichoice and revision programs.

And now that they've developed their TV interface, it means that with any of these subjects they can take a single machine into the classroom and let the whole class see what's going on.

Most of the software they developed themselves, although some has come from MUSE and the Schools Council Projects. Some of the sixth formers are also working on program development. The sort of work areas being covered include timetabling, schools administration, careers, word processing, purchase requisitions and ordering, and so on. Using just one machine they found insufficient memory to cover the whole timetabling, but they could divide the school into 2 or 3 sections and fit the timetable in that way.

A couple of the books they used on the early days were Peter Bishops "Computer Programming in Basic" and PCC's "What to do after you hit RETURN", and found both extremely useful in the development stages. To quote one of the teachers "one part of the problem is that we have to teach sometimes!"

Still on the software front, they found one of the problems was a lack of software in their particular study areas : there was an awful lot aimed at University and college level, but not as much for their own use. Consequently the need to develop their own. As yet they haven't formed a

computer club of their own, mainly because they don't want to establish it as a club but as a serious department.

The amount of access to the machines has to be limited : the problem has been not so much getting them on the machines but getting them off them again. Thus for the fourth and fifth formers the access is restricted to lessons only, but for the sixth fomers there is much more open access : three and a half hours weekly per pupil, and more by prior arrangement if required. They are currently trying to get this access extended down the school, the feeling being that the micro is here to stay, and kids need to be acquainted with them as soon as possible.

Limited Staff

The number of dedicated computing staff is limited : mainly they rely on Dr. Hancock for guidance, and she has three part time teachers to help her. The rest of the staff seem never to have enough time to help out : teaching 32/33 periods a week tends to take up a lot of time! They can go into the computer centre whenever they want, with inservice training being part of their teaching load.

I asked the question "What would you like to happen ?"

"Well, there's no chance of more support from the LEA : the government's nie million pounds is being put into software. This is putting the cart before the horse. The schools won't be able to afford the hardware to run it. We have comitted ourselves and are probably going to make lots of mistakes. An evaulation exercise should be undertaken by the county, by picking out a pilot school and report back in about a year. We had to go to Universities and technical colleges to see how 20/30 machines could be used. We have had to take time from other things over this, so we must make it worthwhile for the children".

"We are going into it a bit blindly, which may not suit some children. Only in the circumstances : an unsupported county : are we doing the right thing. Teachers are learning as we go. The point is that even the dimmest child will have rubbed shoulders with a computer when he has left school".

"We have to decide where our priorities lie. So far we've been carried along by our own momentum, but there is no point in learning the computer for its own sake : we have to decide where we go from here".

"A number of weeks ago we started with computers in the compulsory core of the curriculum for 4th years and over. We already see a need for a further 15 machines, as machines are used fulltime for at least 36 periods of the week. MUPET would be nice to have : sharing a disk unit between the whole class would open up many areas for us. We are looking for development in CAL".

"To sum up, we don't see the micros replacing the teacher, but spreading one teacher more thinly. The possibilities are unlimited. We are not in a position to go to someone and ask how to do something : we have to find out for ourselves. We were lucky to have Physics teacher who was very keen on computers : there is a great shortage of staff who know about computers".

"Basically there just isn't enough software. There's too much aimed at first year university level, so we've joined the Wolverhampton Exchange Library Program".

"Why the PET ? In two years we've only paid fourteen pounds for maintenance. We have no service contract because maintenance is so low. We never push people to use machines : we want them to capitalise on their own".

A school that's doing a lot of enterprising work : let's hope there are many more like them. If you're involved in anything like this, write and let me know.

Comal Introduction

Continued from page 9

allows you room to expand it with your own materials), including a unique note pad inside the binder for notes while learning COMAL syntax. Our diskettes are only top quality with HUB reinforcements. We use floppy armour to ship our diskettes, and use expensive custom made boxes to ship our Starter Kits. We do this because we think that quality seems to be fast disapearing these days. We didn't want you to get a cheaply photcopied manual folded into an envelope. We think you will like COMAL, and we hope our information will meet your standards of quality.

FINAL NOTE for PET/CBM owners: remember, Commodore is not distributing COMAL - so please do not contact them to get a copy. We will give you a copy free with any CBM COMAL User Group disk or our CBM COMAL STARTER KIT. We also are sponsoring a COMAL program exchange for PET/CBM - We will give a FREE User Group Disk to anyone contributing a substantial COMAL program for the exchange.

ABOUT THE PROGRAMS ON OUR CBM DISKS

A commented partial list of programs found on USER GROUP DISK 1, USER GROUP DISK 2, and HELP DISK is printed below. Additional programs will be on the disks.

PROGRAM NAME	LANGUAGE	LOAD ENTER	Comments
comal	BASIC	DLOAD	RUNS FULL COMAL FOR YOU (DLOAD "*" or SHIFT RUN)
comal(blue)full	BASIC	DLOAD	FULL COMAL interpretter
comal80in	BASIC	DLOAD	SPLIT COMAL INPUT Module
comal80ex	BASIC	DLOAD	SPLIT COMAL EXECUTE Module
comalerrors	---	---	File accessed by COMAL system for error messages
basic	COMAL	LOAD	to return to BASIC = CHAIN"BASIC"
getchar.l	COMAL	ENTER	Get one character
multibold'demo	COMAL	LOAD	Demonstrate BOLD FACE printing - STARWRITER only
bold'char.l	COMAL	ENTER	Prints one BOLD FACE character - STARWRITER only
bold'string.l	COMAL	ENTER	Prints a BOLD FACE string - STARWRITER only
enroll	COMAL	LOAD	Part of OLSEN MAIN system

Topic Lists - the first step for curriculum software

Nick Green

Spurred on by questioning from a group of London teachers at a conference at North London Polytechnic, I went to Foyles (a large bookshop in central London) and browsed through the various revision aids and study notes that are published.

It was my view that this material could cut many corners in drawing up topic lists that would be useful to courseware authors.

We would assist by publishing proposals from subject matter experts to avoid duplication of effort, using the medium of what seems to have become known as the "yellow pages".

Subsequently the syllabuses for various topics were examined and where "common core" topics were not a central issue I found they could be very helpful for certain subjects, e.g. geology, but it did not seem so for all subjects, and subject matter experts or expert groups are urged to consider both sources of topics.

Thus we would find by consulting the Key Facts for History 1914 - 1946 that a number of topics are listed.

CAL techniques

The use of CAL techniques in History requires a student to be able to reproduce a model (albeit as an essay) of the material under consideration. Thus, it is seen as the History teacher's role to teach the structure and function of various models to account for historical events. There are some difficulties however. For example should a contra-factual element be introduced ? : in a simulation of the causes of the First World War should a model be able to predict what the outcome of the assassination of Arch Duke Ferdinand would have been, had it occured some days later ? One might envisage a family of models in which initially a model is run with no input and pages of text, graphs, graphical animations are presented as required to the student, and then the ground is gone over again. However, an increasing number of correct responses is required by the student. While this is not my personal choice, I can see it might make the learning of such subjects more palatable.

Another method is straightforwardly asking the student to construct his own simulation, after reading books on incidents in history. For example, the Battler of Jutland, the Peasants' Revolt etcetera. Clearly this is one of the more difficult areas, together with English Literature, to deal with creatively, but if the techniques of model building and simulation are used and the expert is sufficiently knowledgable on his subject, then there is no question but that useful results can be obtained.

Moving onto the less difficult areas of numeric subjects we find that the Key Facts identify several topics for Physics, including topic number 7, which includes momentum. Here one might deal with mass and weight as a single sub-topic, and momentum has a sub-topic with lots of examples of billiard balls and railway carriages etc.

Within 'A' level maths I examined the Key Facts Pass Book, which distinguished 20 super topics from its page of contents and from its index some 200 component parts.

The super topic, Co-ordinate Geometry for example, contains the conic sections and the topic lists includes the definition of such terms as directrix. Clearly, it is not worth devoting a whole topic to the notion of directrix but a topic would be written on the parabola.

Thus, the topics taken from the index of the Key Facts Pass Book can in many cases be collapsed into a single topic and just as in Physics we encountered some sub-topics we might on occasion wish to identify some more.

Finally I had a look at English Language and considered that the contents of the two books provided an excellent map for topics in a CAL course. Such topics as spelling, direct and indirect speech, literary appreciation, essay writing, vocabulary, style in writing and language, comprehension, precis, punctuation, grammar, sentence structure and corrections are all suggested.

Exam Syllabus

The exam syllabus should not be overlooked however, and in considering a subject like Advanced level Geology the syllabus is very clear ; each subdivision can itself become a topic for a computer program to be written.

Thus, for example, looking at the University of London syllabus for 1980/1 I would propose topics on 1) Theories of Continental Drift and Plate Tectonics and 2) the transport of Detritus by Gravity, Water, Wind and Ice. Later in the syllabus we have a topic on the Petrology of Metamorphic rocks and the types of Metamorphic, Thermal, Dynamic and Regional.

Essentially the message is that some effort must be expended at this level and it is well within the grasp of teachers' software workshop to tackle particular curriculum areas with each teacher writing one or two topics and thus contributing to a super topic or breaking down his topic into sub-topics.

One could envisage a planning period, a thinking period on each topic, and then a week's actual coding to produce version one of a particular topic.

We will produce any future developments in this field in future publications.

Nick Green education supremo

Greek Alphabet Codes - PET 3022 Printer

David J. Heathcote

14	17	10	4	26	1	GIVES	α	ALPHA		66	53	41	65		GIVES	ξ	XI
	1	62	80	42	4	GIVES	β	BETA		6	9	17	18	12	GIVES	ο	OMICRON
	64	54	9	54	64	GIVES	γ	GAMMA		9	30	16	30	33	GIVES	π	PI
	22	41	41	6		GIVES	δ	DELTA		62	73	72	48		GIVES	ρ	RHO
	10	21	21	17	2	GIVES	ε	EPSILON	99	85	73	65	65	99	GIVES	Σ	SIGMA [CAPS]
	64	44	50	35	64	GIVES	ζ	ZETA	6	9	9	14	8	8	GIVES	σ	SIGMA
	64	48	65	62		GIVES	η	ETA		8	16	30	17	16	GIVES	τ	TAU
	62	73	73	62		GIVES	θ	THETA	8	6	1	1	18	12	GIVES	υ	UPSILON
		30	1	2		GIVES	ι	IOTA	48	73	14	24	40	48	GIVES	φ	PHI
17	14	4	8	30	17	GIVES	κ	KAPPA	34	36	24	22	33	65	GIVES	χ	CHI
65	66	52	12	2	1	GIVES	λ	LAMBDA	112	9	126	8	48	64	GIVES	ψ	PSI
1	126	32	32	120	4	GIVES	μ	MU	25	38	64	64	38	25	GIVES	Ω	OMEGA [CAPS]
	16	12	3	4	24	GIVES	ν	NU		6	9	2	9	6	GIVES	ω	OMEGA

```
READY.

 100 OPEN5,4,5:OPEN4,4,1:OPEN2,4,2
 110 PRINT#4,CHR$(1)" GREEK ALPHABET CODES PET 3022 PRINTER":PRINT#4:PRINT#4
 120 PRINT#2,"999  999  999  999  999  999     AAAAA    A    AAAAAAAAAAAA"
 130 A$="":C$=CHR$(29)
 140 FORI=1TO6:READA(I):A$=A$+CHR$(A(I)):NEXTI:READN$
 150 PRINT#5,A$
 160 FOR I=1TO6:PRINT#4,A(I);:NEXT:PRINT#4," GIVES"C$CHR$(254)C$N$:PRINT#4
 170 GOTO130
1000 DATA14,17,10,4,26,1,"ALPHA"
1010 DATA0,1,62,80,42,4,"BETA"
1020 DATA0,64,54,9,54,64,"GAMMA"
1030 DATA0,22,41,41,6,0,"DELTA"
1040 DATA0,10,21,21,17,2,"EPSILON"
1050 DATA0,64,44,50,35,64,"ZETA"
1060 DATA0,64,48,65,62,0,"ETA"
1070 DATA0,62,73,73,62,0,"THETA"
1080 DATA0,0,30,1,2,0,"IOTA"
1090 DATA17,14,4,8,30,17,"KAPPA"
1100 DATA65,66,52,12,2,1,"LAMBDA"
1110 DATA1,126,32,32,120,4,"MU"
1120 DATA0,16,12,3,4,24,"NU"
1130 DATA0,66,53,41,65,0,"XI"
1140 DATA0,6,9,17,18,12,"OMICRON"
1150 DATA0,9,30,16,30,33,"PI"
1160 DATA0,62,73,72,48,0,"RHO"
1170 DATA99,85,73,65,65,99,"SIGMA [CAPS]"
1180 DATA6,9,9,14,8,8,"SIGMA"
1190 DATA0,8,16,30,17,16,"TAU"
1200 DATA8,6,1,1,18,12,"UPSILON"
1210 DATA48,73,14,24,40,48,"PHI"
1220 DATA34,36,24,22,33,65,"CHI"
1230 DATA112,9,126,8,48,64,"PSI"
1240 DATA25,38,64,64,38,25,"OMEGA [CAPS]"
1250 DATA0,6,9,2,9,6,"OMEGA"
READY.
```

Programmable Sound Generator

The AY-3-8910 sound generator is a particularly versatile device capable of generating three simultaneous tones, each of which can be separately controlled in amplitude and/or mixed with noise to produce a wide range of sound effects. The particular merit of the GI chip, compared to other sound generators, is that its operation is entirely digitally controlled, making it suitable for use with a micro processor.

In addition to its sound generator functions the AY-3-8910 also features two 8-bit wide general purpose I/O ports (labelled IOA and IOB in the pin digram of Fig. 1). All functions are controlled by sixteen internal registers accessed by a combined data and address 8 bit port (DAO-7 in Fig. 1). The AY-3-8910 is designed principally for use with GI's PIC1600 and 1650 series of mcro processor with bus control pins BC1, BC2 and BDIR determining whether the DAO-7 lines are to be interpreted as address or data lines.

The combined function of the DAO-7 lines do not allow for easy interfacing to other microprocesors such as the 6502 and 6800 series. One method of interfacing that has been proposed uses a 6820 programmable interface adaptor (PIA) with the 8 lines of port A connected to the DAO-7 pins of the sound generator and three of the port B lines for the three bus control pins.

This means of interfacing makes programming the sound chip cumbersome. One needs to simulate the bus waveforms shown in Fig 2. Assuming that one is writing in BASIC, then two POKE commands are needed to set up the 6820 ports as outputs. Then one needs a POKE to send the address of the required internal register to the DAO-7 pins, another POKE to sent LATCH ADDRESS to the bus control pins, a third POKE to send BUS INACTIVE, followed by a fourth POKE to send the data to the DAO-7 pins, a fifth POKE to send WRITE DATA to the bus control pins, and a sixth POKE to return the bus control pins to BUS INACTIVE. These last six POKES must be repeated for each of the sixteen internal registers needing input.

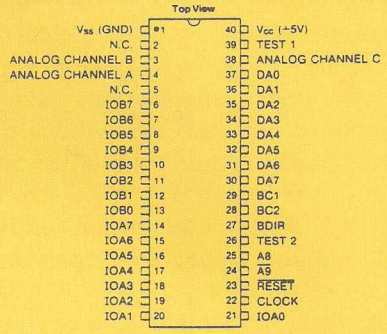
Why can one not make the sixteen registers in the sound generator part of the addressable memory of the micro processor? Then a single POKE to the relevant address would be all that is needed.

The reason that this is not straight forward is that the AY-3-8910 is too slow to respond to the 1uS processor cycle of the 6502/6800 families. Following a falling edge of the 02 clock the minimum time intervals needed are as in figure a.

The solution is to deliberately spread the writing to the sound chip over two processor cycles. The circuit if Fig.3 does this. The relevant signal waveforms are shown in Fig 4. Circuits IC1 and IC2 decode the top eight address line A15-8 and their outputs feed the additional address select pins A8 and A9 on the sound generator IC8. The quiescent state of IC5 leaves a one on its terminal Q and a zero on Q, the latter enabling the tristate output of IC3 thereby applying the lower address lines A7-A0 to the combined data/address pins of the sound generator. The first half of the dual monostable IC7 triggers off each falling edge of 02 and produces a 300 nS pulse. The back edge of this pulse triggers the second half generating a delayed 550nS pulse. This latter pulse is applied to pin BDIR of the sound generator, which together with the output Q of IC5 on BC1 (also high during this period) codes terminates 150nS before the computer address lines A15-AO change. Only the lowest four address lines A3-AO of the eight A7-A0 lines address a register in the sound generator, if the other four lines A7-A4 are not zero, of if A8 and A9 are not 10 respectively then the address is invalid and the sound generator takes no further action.

On the next falling edge of 02 (rising edge of 02) two things happen. The data on the data lines D7-D0 from the micro processor are latched into IC4, and the low, now on pin D of IC5, is clocked through to the Q terminal. This latter signal enables the tristate output of IC4 at the same time as the output of IC3 is disabled. Data, not address, is now applied to the combined data/address lines of the sound generator IC8. The monostables continue to produce delayed 550nS pulses and when BDIR is high again BC1 is now low giving a READ DATA command to the sound generator.

Circuit IC6c is an exclusive OR which inverts 02 whenever the monostable output is high. The result is to convert the 1 MHz 02 into a 2MHz clock which is fed to the clock terminal of the sound generator. This 2MHz clock enables the generation of more precise tones at the higher frequencies than a 1MHz clock would



AY-3-8910 Pin diagram

Figure 1

Figure a.

delay until the processor address is valid	300 nS
AY-3-8910 address set up time	400nS
AY-3-8910 address hold time	100nS
AY-3-8910 data set up time	50nS
AY-3-8910 data pulse width	500nS
AY-3-8910 data hold time	100nS
	1.45nS

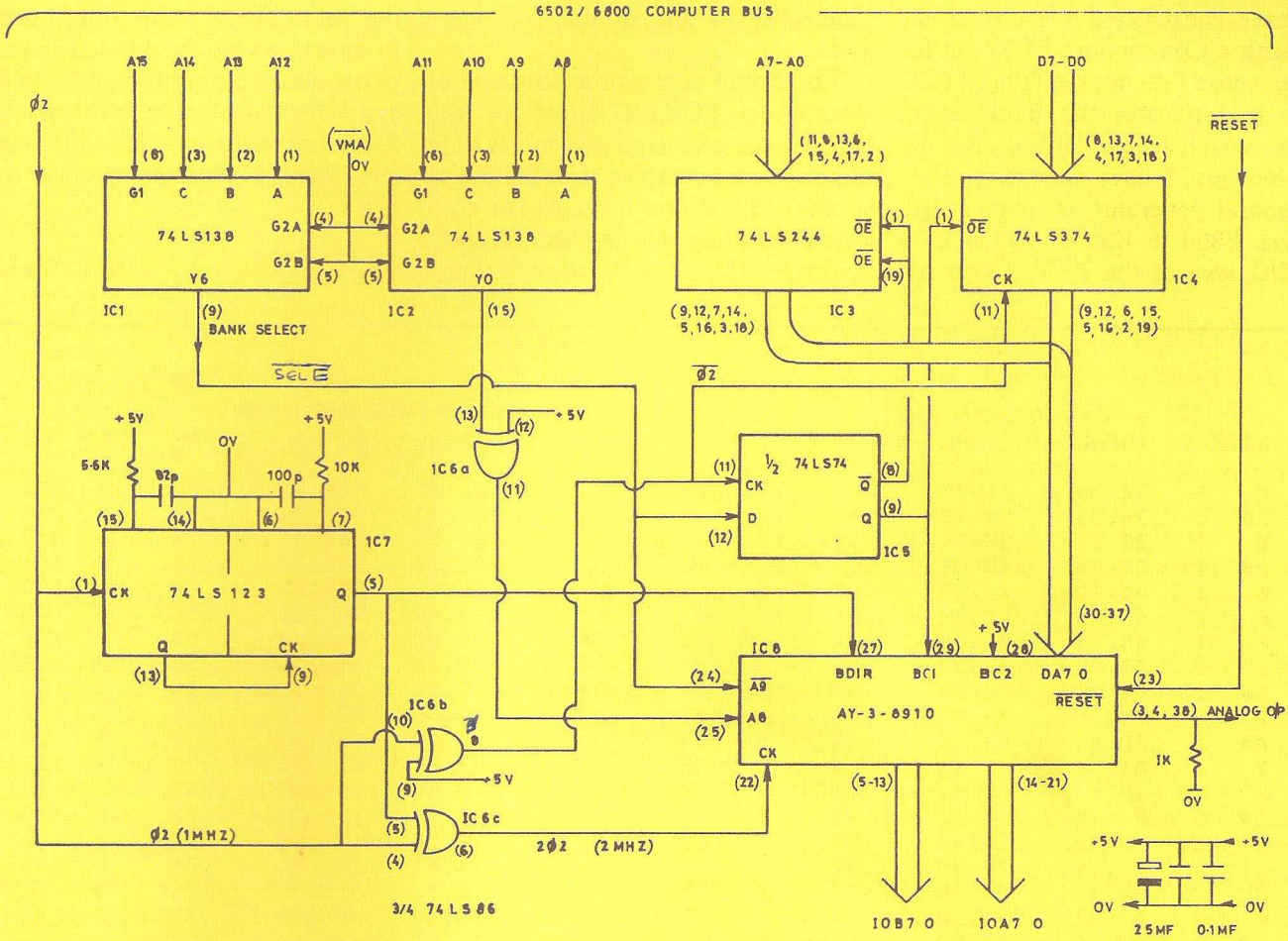


Figure 3 AY-3-8910 Memory mapped on the 6502/6800 Bus

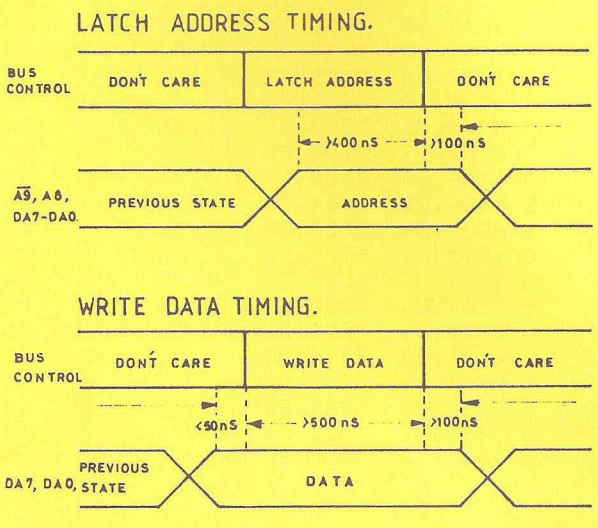


Figure 2 Sound generator timing (Write mode)

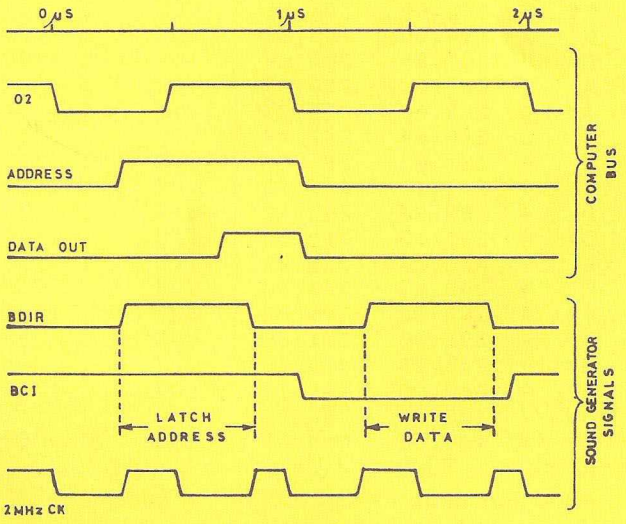


Figure 4 Signal timing

allow, see Table 1. Circuits IC6a and IC6b are simple inverters and could be replaced by a 74LSO4 if the frequency doubling function of IC6a is not required.

I have constructed the circuit for use with a Commodore PET and for this purpose I do not really need IC1, since it duplicates the Bank Select signals which are available on this expansion bus. I have chosen to place the sound generator at memory addresses E800 to E80 F, which is in the I/O area of the PET. Other addresses are possible by choosing different outputs of IC1 and IC2, and possibly the use of the G2 chip select pins. If the circuit is used with the 6800 processor, then VMA should be connected to one of the G2 select pins.

The circuit achives the objective of permitting POKES direct to the sound generator registers and is only slightly more complex than the use of a 6820 PIA if one includes the chip select decoding that is also necessary with the PIA. One function that my circuit does not allow is the reading of the register data. To add this feature is not difficult, although it will require a double, PEEK, one to strobe in the address, and the second to read the data. There seems little purpose in having a read function, I have yet to use the two output data ports IOA and IOB, but plan them for two D to A convertors to provide, additionally, two directly synthesised tone channels.

M.C.Stevens

		--- FREQUENCY (HZ)---			POKE				--- FREQUENCY (HZ)---			POKE	
NOTE		IDEAL	ACTUAL	ERROR	HI	LO	NOTE		IDEAL	ACTUAL	ERROR	HI	LO
C	1	32.70	32.71	.0%	14	238	C	5	523.25	523.01	.0%	0	239
C#	1	34.65	34.65	.0%	14	24	C#	5	554.37	555.56	.2%	0	225
D	1	36.71	36.71	.0%	13	77	D	5	587.33	586.85	-.1%	0	213
D#	1	38.89	38.89	.0%	12	142	D#	5	622.25	621.89	-.1%	0	201
E	1	41.20	41.20	.0%	11	218	E	5	659.26	657.89	-.2%	0	190
F	1	43.65	43.66	.0%	11	47	F	5	698.46	698.32	.0%	0	179
F#	1	46.25	46.24	.0%	10	143	F#	5	739.99	739.64	.0%	0	169
G	1	49.00	49.00	.0%	9	247	G	5	783.99	786.16	.3%	0	159
G#	1	51.91	51.91	.0%	9	104	G#	5	830.61	833.33	.3%	0	150
A	1	55.00	54.99	.0%	8	225	A	5	880.00	880.28	.0%	0	142
A#	1	58.27	58.28	.0%	8	97	A#	5	932.33	932.84	.1%	0	134
B	1	61.74	61.73	.0%	7	233	B	5	987.77	984.25	-.4%	0	127
C	2	65.41	65.41	.0%	7	119	C	6	1046.50	1050.42	.4%	0	119
C#	2	69.30	69.29	.0%	7	12	C#	6	1108.73	1106.19	-.2%	0	113
D	2	73.42	73.40	.0%	6	167	D	6	1174.66	1179.25	.4%	0	106
D#	2	77.78	77.78	.0%	6	71	D#	6	1244.51	1250.00	.4%	0	100
E	2	82.41	82.40	.0%	5	237	E	6	1318.51	1315.79	-.2%	0	95
F	2	87.31	87.29	.0%	5	152	F	6	1396.91	1404.49	.5%	0	89
F#	2	92.50	92.52	.0%	5	71	F#	6	1479.98	1488.10	.5%	0	84
G	2	98.00	97.96	.0%	4	252	G	6	1567.98	1562.50	-.3%	0	80
G#	2	103.83	103.82	.0%	4	180	G#	6	1661.22	1666.67	.3%	0	75
A	2	110.00	110.04	.0%	4	112	A	6	1760.00	1760.56	.0%	0	71
A#	2	116.54	116.50	.0%	4	49	A#	6	1864.66	1865.67	.1%	0	67
B	2	123.47	123.52	.0%	3	244	B	6	1975.53	1984.13	.4%	0	63
C	3	130.81	130.75	.0%	3	188	C	7	2093.00	2083.33	-.5%	0	60
C#	3	138.59	138.58	.0%	3	134	C#	7	2217.46	2232.14	.7%	0	56
D	3	146.83	146.89	.0%	3	83	D	7	2349.32	2358.49	.4%	0	53
D#	3	155.56	155.47	.1%	3	36	D#	7	2489.02	2500.00	.4%	0	50
E	3	164.81	164.91	.1%	2	246	E	7	2637.02	2659.57	.9%	0	47
F	3	174.61	174.58	.0%	2	204	F	7	2793.83	2777.78	-.6%	0	45
F#	3	185.00	184.91	.0%	2	164	F#	7	2959.96	2976.19	.5%	0	42
G	3	196.00	195.92	.0%	2	126	G	7	3135.96	3125.00	-.3%	0	40
G#	3	207.65	207.64	.0%	2	90	G#	7	3322.44	3289.47	-1.0%	0	38
A	3	220.00	220.07	.0%	2	56	A	7	3520.00	3472.22	-1.4%	0	36
A#	3	233.08	233.21	.1%	2	24	A#	7	3729.31	3676.47	-1.4%	0	34
B	3	246.94	247.04	.0%	1	250	B	7	3951.07	3906.25	-1.1%	0	32
C	4	261.63	261.51	.0%	1	222	C	8	4186.01	4166.67	-.5%	0	30
C#	4	277.18	277.16	.0%	1	195	C#	8	4434.92	4464.29	.7%	0	28
D	4	293.66	293.43	-.1%	1	170	D	8	4698.64	4629.63	-1.5%	0	27
D#	4	311.13	310.95	-.1%	1	146	D#	8	4978.03	5000.00	.4%	0	25
E	4	329.63	329.82	.1%	1	123	E	8	5274.04	5208.33	-1.2%	0	24
F	4	349.23	349.16	.0%	1	102	F	8	5587.65	5681.82	1.7%	0	22
F#	4	369.99	369.82	.0%	1	82	F#	8	5919.91	5952.38	.5%	0	21
G	4	392.00	391.85	.0%	1	63	G	8	6271.93	6250.00	-.3%	0	20
G#	4	415.30	415.28	.0%	1	45	G#	8	6644.88	6578.95	-1.0%	0	19
A	4	440.00	440.14	.0%	1	28	A	8	7040.00	6944.44	-1.4%	0	18
A#	4	466.16	466.42	.1%	1	12	A#	8	7458.62	7352.94	-1.4%	0	17
B	4	493.88	494.07	.0%	0	253	B	8	7902.13	7812.50	-1.1%	0	16

Table 1
Course (HI) and fine (LO) Tuning register values using a 2MHz clock.

Figure b

```
PROGRAM:                PROGFILE

60000 REM **PROGFILE**
60010 PRINT"[CLR]":CLR
60020 PRINT"[CUD][CUD]PUT A FRESH FORMATTED DISK IN DRIVE 1.[CUD][CUD]"
60030 INPUT"PROGRAM NAME:";P$
60040 I=1025
60050 A=PEEK(I):IFA<>0THENI=I+1:GOTO60050
60060 B=PEEK(I+3):IFB<>96THENI=I+1:GOTO60050
60070 C=PEEK(I+4):IFC<>234THENI=I+1:GOTO60050
60075 D=PEEK(I+5)
60080 IFA=0ANDB=96ANDC=234ANDD=143THENA=I:GOTO60100
60090 PRINT"[CUD][RVS]ERROR![RVO]":STOP
60100 D=1:T$="W":I=0
60110 $O,D,T$,P$,I$
60120 PRINT"[CLR][CUD][CUD]WRITING BASIC TEXT TO FILE.[CUD][CUD]"
60130 FORI=1025TOA:D=PEEK(I):H$="":H$=STR$(D)
60140 PRINTH$,:$W,H$:NEXTI
60145 B=0:H$=STR$(B):$W,H$:$W,H$
60150 H$="EOF":$W,H$:$C:H$=""
60155 D=I+1
60160 IFATHENA=INT(D/16):GOTO60180
60170 GOTO60195
60180 H$=MID$("0123456789ABCDEF",1+D-A*16,1)+H$
60190 D=A:GOTO60160
60195 IFLEN(H$)<4THENH$="0"+H$:GOTO60195
60200 PRINT"[CUD][CUD][RVS]DONE.[RVO] BASIC TEXT ENDS AT $";H$
60210 POKEI,0:POKE(I+1),0:END
```

search for the next end of line.

60070 If end of line and $60 are detected then read the next byte. If this is $EA (234 decimal) then the high byte for line 60000 has been reached - (234*256)+ 96=60000

60075 Make sure by checking the next byte for $85 (decimal 143), which is the token for the Basic reserved word REM, and the first word of line 60000

60080 confirms these values and sets A equal to the counter I. Variable A now holds the decimal value of the location containing the zero byte at the end of the last line of the program to be cross-referenced. If not, then 60090 gives an ERROR message, (In many trial runs, this line has never been activated)

60100 to 60140 opens the disk file, converts into decimal strings the values of all Basic text locations from 1025 up to the end of the main program and writes them serially to the file

60145 and 60150 writes two further zero bytes at the end of the file, followed by an "EOF" and closes the file

60155 to 60200 converts the highest RAM location of the main program text into a hex string and prints this on the screen. You can then check by the $MEM command, if you wish, that the few bytes preceding this hex address do, in fact, contain the text of the last Basic line of the program to be cross-referenced

60210 restores the two bytes to the memory at the end of the main program. Type LIST and you will see that the listing terminates at the end of the program to be cross-referenced. Lines 60000 and upwards are not listed and have not been included in the disk file. (Actually, if you use the $MEM command, you can see that the text for PROGFILE is still resident in memory but it cannot be accessed from Basic because the link address for line 60000 has been removed by the zero POKEs in line 60120).

The procedure for cross-referencing a program is quite straightforward, and the steps are as follows:

1. Load a machine code BASIC LINKER program (or Petsoft's PDAS or similar)
2. Load PROGFILE and then link it to the program to be cross-referenced
3. LIST and check that the main program and PROGFILE are properly linked
4. Put a fresh formatted disk in Drive 1 to receive the file
5. Type RUN60000
6. Check the end of Basic text from the few bytes preceding the hex address given by PROGFILE at the end of its run, to ensure that this is the last line of BASIC program
7. Type NEW and load CROSS REF. (If you are cross-referencing CROSS REF itself then you don't have to reload it because it is still in the memory)
8. RUN. Input the file name and enter V for variables or L for lines (GOTOs, THENs, GOSUBs), and watch the details build up on the screen. This part is fairly slow as we are running in Basic
9. When complete, and the disk drive closes down, type Y if you require hard copy from the printer. Type N for a print-out on the screen
10. On rare occasions the printer may stop listing in the middle of a long line. This is not a fault, so do not abort by pressing the stop key unless the pause is longer than 15-20 seconds.

Using CROS REF to cross-reference itself, the following print-outs are obtained: (see over page for variables printout fig d.)

```
CROSS REFERENCE - PROGRAM: CROSS REF

  180   470   490
  210   210
  240   190
  300   308   470   500
  307   309
  310   300   302   305
  320   320
  360   326   340
  370   330
  380   310
  420   380
  460   440
  470   390   400   410   430   450
  480   470
  490   510   520
  530   180   245
  580   565
  600   580
  620   600
```

Figure c

One puzzle out of several, and the most tricky, to be solved in converting CROSS REF was the listing of DISKMON commands such as $0, $R, $C, as the variables O, R and C, and also a corruption of the next line number when the character $ was the last character on a Basic program line. This has been cured by the re-

```
CROSS REFERENCE - PROGRAM: CROSS REF

A       245  280  300  302  304  305  307  308  310  390  400
        460
A$      240  245  304  307  490  520  560  570  580
A$(     100  200  330  340  350  360
B       190  200  220  326  330  340  350  360
B$      240  245  480  500  520  580  590
B$(     100  120  480
C       280  370  390  410  420  430  440  450  540  550
C$      480  520  580  620
C(      100  140  150  160  310
C1      280  310  370  420  440
C2      130  150  280  320  370  380  450  565
C9      310  410  420  440  470  480
D       300  306  309
DI$     170
DR$     180  185  240  304  307  490
E       120  302  308
J       140  150  200  210  220  330  340  350  560  630
K       200  210  220  326  340  350  360  565  570  580
L       245  280
L$      200  206  280
M$      320  330  340  360  370  450  460
N       490  500  510
P$      170  550
Q$      120
S$      120  280  590  610  620
X       200  220  560
X$      200  206  210  220  560  580  590
X$(     100  210  220  560
Y       590  600  610
Z       540  600
Z$      130  530  540
```

Figure d

vamping of lines 300 to 309.

Very many thanks to Jim Butterfield and CPUCN for giving us a tremendously useful program, and I hope that with the aid of these notes other Computhink disk users will be able to enjoy the powerful facilities that CROSS REF provides.

Aubrey Jones.
British Telecomm

Leapfrog

The accompanying letter should give you a good idea of what this is all about, but a few words for 80 column users who want to see what's happening. Before typing the program in, and indeed before running it thereafter, type :-

printchr$(142)

Followed by hitting the return key. This puts your PET into graphics mode, and also gives you continuous graphics on the screen, rather than a slight gap between each line. To get out of this afterwards, use 14 instead of 142. A number of graphics characters are used in the listing , and to save you spending hours looking for them (since they're not actually marked on the keyboard), here are the relevant characters :-

In line 110 we encounter most of the graphics characters producing cursor movement. So line 110 would, in descriptive form, read :-

110 F$(0)=" (3c1)(cd) (cu)"

The only other one we come across occurs first of all in line 150, were the second character inside quotes is a cursor right. Now for the graphics characters. To define these, if a character in a line has a '-' over it, press SHIFT as well as the relevant character to get the desired graphical result. There are seven characters used in all, and line 130 features 5 of them. In descriptive form this would read :-

130 F$(2)="$\overline{J}(rvs)\overline{Q}(off)\overline{K}$(3cl,cd)$\overline{U}(rvs)\overline{Q}(off)\overline{I}$(cu)

These are not exactly like the characters in the listing, but they're the closest the 8032 can get, since the characters listed used to be got by shifting the numeric keys, and of course this doesn't work anymore. The other characters we encounter are in line 150 first of all, where the first character within quotes is a shifted C, and in line 260 where the line is achieved by pressing SHIFTed R (again this is as close as 8032 users can get, as the line should really be a shifted $ sign, but all we get is ... a $ sign!).

Enjoy yourself!

From P.R. Gabor,
Tel-Aviv, Israel

Dear Pete,

I sincerely believe in mixing pleasure with business - at least when the Pet is concerned.

So why not take a simple game program, and fiddle around with it a bit to emphasize the various possibilities of the PET? I think the enclosed listing might interest those readers of the "Newsletter" who are comparatively new to the Commodore Family and would like to learn more about Basic Programming.

This is a simple checker-game, called Leapfrog. The object of the game is to exchange the places of the frogs, as explained in lines 300 to 440.

Lines 460 waits for an input to start the game. The first problem was to make a graphical image of the frogs, I think the shifted "Q" characters are very suitable for this. The actual gameboard is the array A(X), a value of "1" represents a green frog, a "2" represents a black frog and a zero - an empty space. Now the main problem was to write an intelligent input sequence, where nothing but the appropriate number-keys should be pressed. This meant that I had to use "get", and not "input", and that the "PET" had to be able to differentiate, whether a "1" meant "one" or whether it was the first digit of a two-digit number. A "O" could not be accepted as the first digit. The "Delete" key should annul the entry.

The input routine starts at line 1340, the actual "get" routine at line 1820. Note the use of the flag DEL. The reason for it is, that we cannot jump directly from the subroutine to the "delete" routine. First we have to return to the calling routine (terminating the subroutine sequence) - and the calling routine must be told that it had to start the input routine all over again!

The second digit input might require some additional explanation. Line 1850 checks, if the first digit is a "1". If not, then there is no second digit. Line 1860 multiplies contents of elements 8-11 of Array A(x). Now one of these elements must be a zero (pointing to an empty space) if the leap is to or from a 2-digit. number. Therefore, if the result of this multiplication is not zero, there is no second digit.

By the way, line 1900 could reject a second digit greater than one (change "9" to "1") - but I could not resist the temptation to have a frog "scream" and fall off the CRT... (The animated sequence is between lines 1600 and 1760). Incidentally, the regular leaps are animated in lines 710-810, lines 730, 750, 770, 800 are the time-delays between the phase of the leap, these can of course be adjsted as required. By the

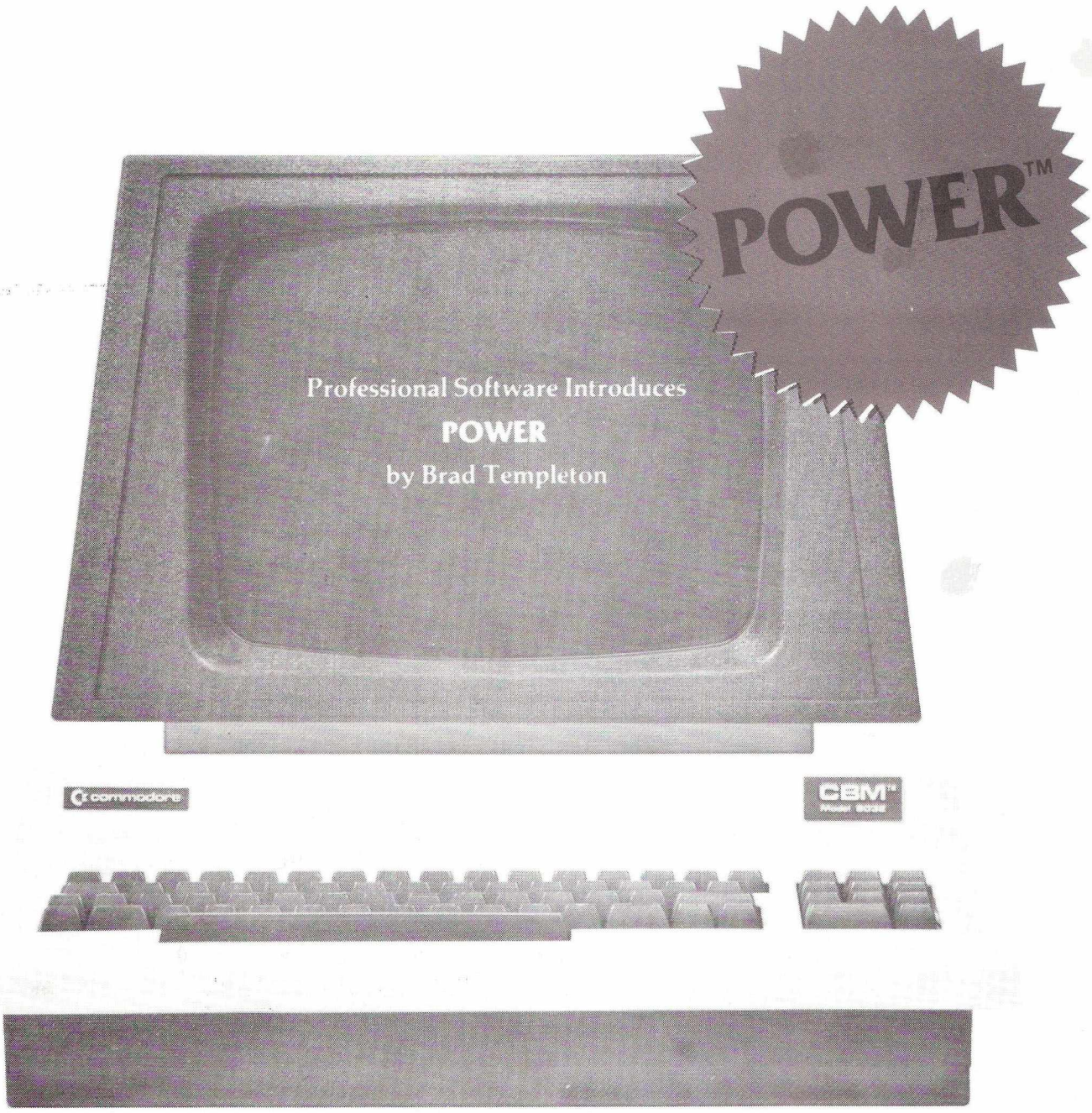
way, A = sin (3) in lines 1650 and 1660 also acts as a time delay.
I believe, the rest is properly explained by the remarks in the listing.
So type in the program, and have fun.
Incidentally - it **can** be done in 35 moves.
Kindest regards.

Yours sincerely,
P.R. Gabor

```
10 REM ********************************
20 REM *                              *
30 REM *      L E A P F R O G         *
40 REM *                              *
50 REM *         BY P. GABOR          *
60 REM *           20/08/81           *
70 REM *                              *
80 REM ********************************



90 :
100 DIM A(19):                          REM     ARRAY FOR FROGS
110 F$(0)="   ■■■■   ":                  REM     SPACE
120 F$(1)="◆■■■■◆":                     REM     STRING TO PRINT GREEN FROG
130 F$(2)="◆■■■■◆":                     REM     BLACK FROG
140 :
150 AN$(1)="─┼┼─■■■■":AN$(2)="■─┼┼─■■■■":  REM     STRETCH LEGS
160 AN$(3)="  ┼┼■■■■":AN$(4)="■  ┼┼■■■■":  REM     PREPARE FOR TAKEOFF
170 AN$(5)=F$(0):                       REM     AND OFF YOU GO
180 AN$(0)="■└┘■■■■":                   REM     FALLING!
190 :
200 PP$="■■■■■":                        REM     MOVE CURSOR TO FIFTH ROW
210 :
220 REM ********************************
230 REM *        THE RULES             *
240 REM ********************************
250 :
260 T1$="                 ──────────────"
270 T1$=T1$+CHR$(13)+"                 "
280 T1$=T1$+"■ LEAP - FROG *"
290 PRINTT1$:PRINT
300 PRINT"FIVE LITTLE GREEN FROGS AND FIVE LITTLE"
310 PRINT"BLACK FROGS ARE SITTING ON A VERY TALL"
320 PRINT"FENCE. THERE IS JUST ROOM FOR ONE FROG"
330 PRINT"BETWEEN THEM, LIKE THIS: -■"
340 FORK=1TO5:PRINTF$(1);:NEXT:PRINT"    ";
350 FORK=1TO5:PRINTF$(2);:NEXT:PRINT"■■"
360 PRINT"THE OBJECT OF THIS GAME IS TO EXCHANGE"
370 PRINT"THE GREEN AND THE BLACK FROGS BY MOVING"
380 PRINT"ONE FROG AT A TIME BY ONE SPACE (IF THE"
390 PRINT"NEXT SPACE IS FREE) OR HAVING A FROG"
400 PRINT"LEAP OVER ITS NEIGHBOUR TO A FREE SPACE"
410 PRINT"(DON'T LEAP OVER MORE THAN ONE FROG)."
420 PRINT"TO REACH THIS POSITION: -■"
430 FORK=1TO5:PRINTF$(2);:NEXT:PRINT"    ";
440 FORK=1TO5:PRINTF$(1);:NEXT:PRINT"■■"
450 PRINT"■    ■PRESS ANY KEY TO START THE GAME"
460 GET A$: IF A$="" THEN 460
470 :
480 REM ********************************
490 REM *       START GAME             *
500 REM ********************************
510 :
520 FOR K=1TO5:                         REM     INITIALIZE ARRAY
530   A(K)=1:A(K+6)=2
540 NEXT:A(6)=0
550 C=0:                                REM     SET COUNTER TO ZERO
560 PRINTT1$:PRINT:PRINT
570 FORK=1TO11:                         REM     PRINT FROGS
580   PRINTF$(A(K));
590 NEXT:PRINT"■■"
600 PRINT"                                    ."
610 PRINT"■  1  2  3  4  5  6  7  8  9  10 11"
620 PRINTPP$"■■■■■"
630 PRINT"ENTER YOUR MOVE!     - ";
640 GOSUB1340:FF=0:                     REM     RESET FLAG FOR FALL-OFF
650 PRINT"■":TD=1500:                   REM     SET CONSTANT FOR TIME DELAY
660 IF A(S)=0 THEN 1470:                REM     STARTING POS. EMPTY
670 IF ABS(S-E)>2 THEN 1500:            REM     LEAP TOO LARGE
680 IF A(E) THEN 1520:                  REM     ENDPOSITION OCCUPIED
690 IF E>11 THEN FF=1: E=12:            REM     JUMP OFF BOARD
700 X=0
710 PRINTPP$;TAB(3*(S-1));:             REM     POSITION TO JUMPING FROG
720 FOR K=1 TO 5:                       REM     ANIMATE JUMP
730   FOR L=1 TO 25: NEXT
740   PRINTAN$(K);
750   FOR L=1 TO 60:NEXT
760 NEXT
770 FOR L=1 TO 50:NEXT
780 PRINTPP$;TAB(3*(E-1));F$(A(S));
790 PRINT"■■■"AN$(1);
800 FOR K=1 TO 150:NEXT
810 PRINTF$(A(S));
820 C=C+1
830 A(E)=A(S): A(S)=0
840 FOR I=1TO6
850   X=X+A(I)*10↑I
860 NEXT
870 IF FF THEN 1600
880 IF INT(X)<>222220 THEN 620:         REM COMPUTED X IS NOT EXACT INTEGER
890 :
900 REM ********************************
910 REM *          FINISH              *
920 REM ********************************
930 :
940 PRINT"■■■■■■":IF C<60 THEN990
950 PRINT"                              "
960 PRINT"■YOU FINISHED, AT LAST!!!"
970 GOTO1020
980 :
990 IF C<50 THEN1060
1000 PRINT"                              "
1010 PRINT"■NOT A BAD RESULT!!!"
1020 PRINT"■YOU NEEDED"C"MOVES TO SOLVE THE PROBLEM"
1030 PRINT"YOU SHOULD REALLY DO BETTER!!"
1040 GOTO1240
1050 :
1060 IF C<40 THEN1120
1070 PRINT"                              "
1080 PRINT"■VERY WELL DONE!!!"
1090 PRINT"■YOU SUCCEEDED TO COMPLETE THE GAME IN"
1100 PRINT"ONLY"C"MOVES - AN ABOVE AVERAGE RESULT!":GOTO1240
1110 :
1120 IF C=35 THEN1190
1130 PRINT"                                          "
1140 PRINT"■EXCELLENT!! YOU ARE A REAL EXPERT!!!"
1150 PRINT"■YOU HAVE DONE IT IN ONLY"C"MOVES, THIS"
1160 PRINT"IS ALMOST THE BEST POSSIBLE RESULT."
1170 GOTO1240
1180 :
1190 PRINT"                                          "
1200 PRINT"■■CONGRATULATIONS! NOBODY CAN DO IT BETTER"
1210 PRINT"■YOU COMPLETED THE GAME IN 35 STEPS."
1220 PRINT"THIS IS THE ABSOLUTE MINIMUM."
1230 :
1240 PRINT"■■WOULD YOU LIKE TO PLAY AGAIN? (Y/N)"
1250 GET A$: IF A$<>"Y" AND A$<>"N" THEN1250
1260 IF A$="Y" THEN520
1270 PRINT"■THANKS FOR PLAYING 'LEAP-FROG' -"
1280 END
1290 :
1300 REM ********************************
1310 REM *      INPUT ROUTINE           *
1320 REM ********************************
1330 :
1340 POKE158,0:                                 REM     NO CHAR IN KEYBOARD BUFFER
1350 DEL=0:PRINT"■":                            REM     FLAG, SET IF DELETE
1360 PRINTTAB(23);"FROM          ■"
1370 PRINTTAB(28);"■ ■■";
1380 GOSUB1820: IF DEL THEN 1350
1390 S=VAL(C$): PRINT" TO ■ ■■";
1400 GOSUB1820: IF DEL THEN 1350
1410 E=VAL(C$): RETURN
1420 :
1430 REM ********************************
1440 REM * MESSAGES FOR BAD INPUTS  *
1450 REM ********************************
1460 :
1470 PRINT"THERE IS NOBODY AT POSITION #"S"■■.■"
1480 PRINT"PLEASE TRY AGAIN!":GOTO1570
1490 :
1500 PRINT"HEY, I CANNOT JUMP THAT FAR!!":GOTO1570
1510 :
1520 PRINT"HEY, WHAT DO YOU THINK YOU ARE DOING??"
1530 PRINT"■THAT IS MY BEST FRIEND, SITTING ON"
1540 PRINT"■SEAT NR."E"■■! YOU CAN'T EXPECT ME TO"
1550 PRINT"■LAND ON HIM!?!  PLEASE TRY AGAIN"
1560 TD=3000:                                   REM       INCREASE TIME DELAY
1570 FOR K=1 TO TD: NEXT:                       REM       TIME DELAY
1580 GOTO560
1590 :
1600 PRINTPP$;TAB(33);F$(0);"■■■";
1610 PRINTF$(A(E));"■■■";AN$(1);AN$(2);
1620 A$="%##%!!■■■■■■■"
1630 PRINT"■■■■";TAB(34);
1640 FOR K=1 TO 20
1650   PRINT"■"A$;: A=SIN(3)
1660   PRINT"■"A$;: A=SIN(3)
1670 NEXT: PRINT"        ■■■■"
1680 PRINT"                              ──────────────"
1690 PRINT"■NOW LOOK WHAT YOU HAVE DONE!!"
1700 PRINTPP$;"■";TAB(34);AN$(0);
1710 FOR K=72 TO 0 STEP -4
1720   PRINTF$(A(E));"■■■";AN$(0);
1730   FOR L=1 TO K: NEXT
1740   PRINTF$(0);"■■■■";
1750 NEXT:A(12)=0
1760 PRINTF$(0);"■■■■■■":GOTO 1240
1770 :
1780 REM ********************************
1790 REM *GET A ONE- OR TWO-DIGIT NR *
1800 REM ********************************
1810 :
1820 GETC$: IF C$="" THEN 1820
1830 IF C$=CHR$(20) THEN DEL=1:RETURN:  REM       DELETE
1840 IF C$<"1" OR C$>"9" THEN 1820:     REM       REJECT ENTRY
1850 PRINTC$;: IF C$<>"1" THEN RETURN:  REM       SINGLE DIGIT ENTRY
1860 IFA(8)*A(9)*A(10)*A(11)THENRETURN:REM        POS 8-11 OCCUPIED
1870 PRINT"■ ■■";
1880 GETCC$:IF CC$="" THEN 1880
1890 IF CC$=CHR$(20) THEN DEL=1:RETURN:REM        DELETE
1900 IF CC$<"0" OR CC$>"9" THEN 1880:   REM       REJECT ENTRY
1910 C$=C$+CC$:PRINTCC$;:RETURN
READY.
```



ADD POWER TO YOUR COMMODORE COMPUTER

£49 + VAT

POWER produces a dramatic improvement in the ease of programming BASIC on Commodore computers. POWER is a programmer's utility package (in a 4K ROM) that contains a series of new commands and utilities which are added to the Screen Editor and the BASIC Interpreter. Designed for the CBM BASIC user, POWER contains special editing, programming, and software debugging tools not found in any other microcomputer BASIC. POWER is easy to use and is sold complete with a full operator's manual written by Jim Butterfield.

POWER's special keyboard 'instant action' features and additional commands make up for, and go beyond the limitations of CBM BASIC. The added features include auto line numbering, complete tracing functions, single stepping through programs, line renumbering, and definition of keys as BASIC keywords. POWER's 'WHY' command enhances debugging by listing the appropriate program line and highlighting where BASIC stopped executing. The cursor movement keys are enhanced by the addition of auto-repeat, and text search and replace functions are added to help ease program modification. POWER can even execute a sequential tape or disk file as though it were typed on the keyboard, allowing the user to merge two BASIC programs together. Cursor UP and Cursor DOWN produce **previous** and next lines of source code. COMPLETE BASIC PROGRAM listings in memory can be displayed on the screen and scrolled in either direction. You can even add your own commands to BASIC. Like our very successful Word Processing Programs (the "WordPro" series), POWER even includes convenient "stick-on" keycap labels which define new functions on the keyboard. POWER is a must for every dedicated CBM user.

Call us today, for the name of the Professional Software dealer nearest you.

Professional Software, Ltd.
153 High Street
Potters Bar
Hertfordshire EN6 5BB
Tel: (STD 0707) 42184 / (STD London 77)

The BASIC Compiler

John Stout

In this issue the overall structure of the compiler is given, together with examples of the assembly language that will be generated by the compiler for some of the statement types encountered in BASIC programs. Enough information should be presented here, in a 'bare bones' form, for anyone with sufficient time and energy to work ahead of the series of articles. Should anyone do so I would be most interested to receive any thoughts about the compiler and its construction and to include them in future issues.

The compiler will consist of an initialisation subroutine, which will be added to as the series progresses, a file handling section which asks the user for the filenames to be used by the compiler and opens them, the two main program loops, the first of which compiles a line of BASIC and the second, nested inside the first, which compiles BASIC statements within the line. Finally there will be one subroutine for each possible statement type, e.g. POKE, IF, PRINT.

BASIC 2 and BASIC 4 differences.

In writing the compiler there are a number of areas where the differences between BASIC 2 and BASIC 4 could cause some difficulty. For the compiler's side (as opposed to the assembler's) these are minor. Testing for the status of the disk is easier in BASIC 4 (using DS and DS$) so we shall use a subroutine which returns the two variables DS and DS$ as a disk error checker. For BASIC 4 this subroutine will consist of a RETURN statement only, while the BASIC 2 version will input from the error channel the necessary variables and convert them to the DS and DS$ form. BASIC 4 of course has a number of extra statement types concerned with handling the disk more easily, e.g. APPEND and RECORD, but the compiler will generate the necessary assembly language to handle these statement types in the way publicised by Jim Butterfield (Compute Issue 9, February 1981) and others. This involves altering the filename for such things as APPEND (add ",A" to the filename when opening) and printing strange combinations of secondary addresses and record numbers to the error channel for RECORD. Since both sets of ROMs can handle this sort of low level programming (done by BASIC 4 anyway) there should be no problem is using exactly the same procedure to handle APPEND etc on both ROM sets.

Since we will use the assembler to handle any absolute or page zero addresses the compiler need only use symbols for referencing such addresses, e.g. INTVEC for the interrupt vector, CLSFLS for the $FFCF routine to close all files and so on. Thus the code generated by the compiler will be the same for BASIC 2 and BASIC 4 and we need only change the equates which set the symbols to be some value, e.g. INTVEC=$90, CLSFLS=$FFCF, in the file BASICLIB.SRC which will be included in the assembly language output file using the .LIB assembler directive. In fact what we shall probably do is to have two versions of BASICLIB.SRC, called BASIC2LIB.SRC and BASIC4LIB.SRC and ask the user what machine the compiled program is to run on, including only the file that is appropriate for the ROM set to be used.

Initialisation.

Where possible each subroutine handling a statement type will do its own initialisation but there are some global variables that will need to be set up before any compiling can be done. For example we will use variables IN and OUT to denote the channel numbers used for inputting from the BASIC program file and outputting to the assembly language file. Counters used in generating labels for such things as IF statements will need to be set to zero, the error/command channel will need to be opened to the disk and so on. A 128 element integer array will be used to hold zeroes in element I if the token value (128+I) is not a valid first token in a statement and holding a value J if it is. This value J will be a number which is to be used as an index in a series of ON...GOTO and ON...GOSUB statements which will direct control within the compiler to the routine appropriate for the statement type being compiled. In fact by using the absolute value of the value J we can use the fact of an element in this array (which we shall call VT% ()) being negative to show that a statement type with token (128+I) has been met in the program. When compilation has finished the compiler can scan through TV% () and perhaps only include routines from a set of library routines on disk which are needed in this particular program. Thus instead of loading a fixed size block of routines, some of which may never be called because there are no instructions in the compiled program which use them, routines in the compiled vesion for this compiler will be unused only if the program flow avoids instructions which use them.

Since we propose to list each compiled line to the assembly language file as a comment (probably broken into statement comments), and since the BASIC language input will be in token form we will need a string array T$() say, which will contain in element (I-128) the string representation on the token whose value is I, e.g. T$(0)="END" since the token for END is 128.

This is not of course a complete list of the initialisation needed, but rather than attempt to forecast the compiler's future requirements we will leave the initialisation in a subroutine and add to it as the compiler progresses.

File handling

The file handling section of the compiler will use the concept of a filename extension to operate properly. A file extension is a short suffix, added to a filename proper, which conveys some information about what sort of information the file contains. We adopt the convention that a filename ending with .BAS contains a **BAS**IC SAVEd file, while one ending in .SRC contains ASCII **SouRC**e, suitable for input to the assembler with each line terminated by a carriage return character. Note that the extensions in these two cases are BAS and SRC, i.e. the . is simply used to separate the filename from its extension.

When the user is asked for the name of the BASIC file to be compil-

ed the compiler will search for a file of the name given first of all. If this search fails the compiler will then look for a file whose name has the BAS extension and if this search fails report an error. Of course if the user enters a filename already with the BAS extension only one search will be performed.

When the user is asked for the assembly language file name things are slightly different. Since the compiler creates this file it will always use the SRC extension and if a file of that name already exists the user will be asked if it is to be deleted. If it is not then the user is asked for an alternative file name otherwise it will be deleted and a new file with the same name created.

Line handling

The line handling loop has at its start a subroutine call to read the next line of BASIC in, ending the compiler run if the high byte of the line pointer at the start of the line is zero, and producing two versions of the line. The first contains the token form in a string LN$ while the second, in LI$, contains the human readable form for listing to the SRC file as a comment and for printing to the screen in the case of an error. A variable LN will be set to 1 by this routine and will always point to the next byte of the source line which is to be compiled. After reading the first byte of the source line the routine will return to the main program.

The next task of the line handling loop is to generate any necessary labels. These will take the form of Lnnnn and Innnn as explained in the previous article. Labels in the form Lnnnn will act as the targets for GOTO and GOSUBs while Innnn labels will be used if there was an IF statement on the previous line which needs a target to jump to if the IF condition fails. Also generated here will be code to call the subroutine LINNUM, again discussed in the first article. Control then passes to the statement handling loop.

Statement handling

At the start of each statement (which may also be the start of a line) we must check to see if the previous statement was an ON...GOTO/GOSUB. If it was then we need to generate a label of the form Onnnn which is used as a target if the index in the ON...GOTO/GOSUB falls outside the range of possibilities given or if the ON...GOSUB form was used. In both these cases we need to be able to reach the start of the next statement (as opposed to line for IF statements).

The first byte of the statement is then examined. If it lies in the range 65..90 (ASC("A")..ASC("Z")) then we must have an assignment statement, e.g. A=O, Z(1,1)=25.6. We then pass control to the relevant routine which will check to see if the variables have been defined (see the previous article for details of how variables will be defined) and then generate the necessary assembly language to cause evaluation of the assignment expression (as remarked in the previous article the expression evaluator will be the heart of the compiler since it will have to deal with defining variables, array indexing, type matching and so on). Should the first byte of the statement fall in either of the ranges 0..64 or 91..127 a syntax error is reported, with two possible exceptions. The first is where more than one colon is used as separator between two statements, e.g. PRINT A::GOTO 10. This is equivalent to allowing one of the possible statement types to be the null statement. The second possiblity is where someone uses an IF...THEN statement followed by a number, e.g. IF A=O THEN 100. In the first case we simply read the next byte and start the statement loop again while in the second we back the pointer LN by one and goto the routine for handling a GOTO statement.

The only other possibilities left for a byte are in the range 128..255 so that we check the VT% () array for the token we have encountered being a valid first token (if not we print the line in error with an error message), set the value of the VT% () element being examined to be negative and then using a number of ON...GOTO/GOSUB statements end up by executing the correct routine for the statement type met. Each of the statement type subroutines will, when returning to the main program, have read the terminating byte of the statement they have just compiled (usually a colon or a null byte).

Errors

For the moment we adopt a fairly simple error handling technique. When an error is met the line in error is printed to the screen (or printer if the user prefers and has a printer) and the location of the error pointed out as far as possible. An error message is then printed and the scan of the line proceeds until the offending statement or line has been passed over. Compilation then proceeds with the next line or statement.

Samples of the assembly language generated by the compiler

To finish off this discussion of the compiler we will show the type of assembly language statements produced by the compiler for certain types of BASIC statements.

A REM statement is perhaps the simplest of all the statements (save the null statement introduced before) since it generates no assembly language at all. All the compiler does is to ignore the rest of the current line and advance straight to the next.

LET is almost as simple since we ignore the LET and read in the next byte (note that the read next byte routine will be like the BASIC interpreter's CHRGET routine in that it will ignore spaces) and returns to the head of the statement handling loop. This does mean that a statement such as LET GOTO 100 will be compiled successfully but if anyone wishes this can be avoided fairly simply.

GOTO is another simple statement to compile since all the compiler need do is to convert the target line number to hexadecimal, prefix the hexadecimal representation by L and output the instruction JMP Lnnnn to the assembly language file. Undefined line numbers will be caught by the assembler as undefined symbols but if need be we can let the compiler do some elementary housekeeping and list undefined line numbers at the end of compilation.

GOSUB is much like GOTO except that we produce a JSR Lnnnn rather than a JMP Lnnnn. The only complication might be that we should check for stack underflow before we perform the JSR, otherwise we might well crash the compiled program. Possibly we could build a stack check into a frequently executed routine in the BASIC*LIB.SRC file, e.g. LINNUM, and halt execution if only a few bytes are left in the stack.

Similarly RETURN should be easy to compile since it could compile to just an RTS instruction. However when BASIC executes a RETURN it

checks for any active FOR...NEXT loops that have been defined since the corresponding GOSUB and removes them from the stack. Since we will be using a different arrangement for the FOR...NEXT loops this will not be strictly necessary, but it will be as well to include such a check for completeness and compatibility. Each record of a FOR...NEXT loop will then need to contain the value of the stack pointer when the loop came into existence, but the subject of FOR...NEXT loops is best left for a future issue.

ON...GOTO/GOSUB statements are the most complicated statements that we consider here. The generated code must evaluate the value of the index (i.e. the ...), check that the value, when converted from floating to fixed point, of the index does not exceed the number of possible line alternatives and is not less than 1, then load the address of the start of the next **statement** less 1 onto the stack for an ON...GOSUB, use the value of the index to index a table of addresses of the relevant line numbers and use the PHA/PHA/RTS technique to transfer control to the correct line number. The code generated will be something like figure a.

IF statements are the final statement type we consider here. They are fairly simple but raise a point which cannot be answered until the compiler has been used on a number of programs. The general idea is that we generate assembly language which leaves a Boolean value (true of false) in the primary floating point accumulator, using the standard PET convention- that 0 represents false and non-zero true. We then test the value of the exponent of the result and if it is zero we want to jump or branch to the next line, i.e. to the label Innnn. However, the question that needs to be settled is whether we can branch, in which case we assume that the assembly language generated by the compiler for the rest of the line won't occupy more than 127 bytes, or whether we need to assume the worst and always jump to the next line. The two cases are shown in figure b.

The second method is guranteed never to fail but needs an extra 3 bytes and is slower. Perhaps the simplest solution is to adopt the first method and if it consistently fails to change the compiler over to using the second method. If only occasionally does the first fail then it might be worth while leaving the first method alone for the cases where it works and manually altering the few that fail. Since the compiler is generating assembly language that can be read by the assembly language development system, including the editor, this is a simple thing to do, and could be the way to produce any optimisation without expanding the size of the compiler.

Summary

We have looked at the overall structure of the compiler and established a number of general points about its operation. Some questions still remain to be answered, e.g. DIM options, FOR...NEXT structure and so on, but most of these are sufficiently complex to warrant a complete article.

In the next issue the first section of actual programming will be given, which should be able to compile most of the statement types looked at in some detail in this article.

Figure a.

```
;ON VT-10 GOSUB 10,20,30,40,50
          ....
          ....
          ....
          ....      ;CODE TO PRODUCE THE VALUE OF VT-10 IN FIXED POINT IN LOCATIONS
                    ;FIXPTL, FIXPTH.
          LDA FIXPTH
          BNE Onnnn
                    ;IF THE HIGH BYTE IS NON-ZERO THEN THE ON...GOTO/GOSUB MUST FAIL
          LDX FIXPTL
          DEX       ;INSTEAD OF RANGE 1..aa (NUMBER OF ALTERNATIVES) USE RANGE
                    ;0..(aa-1).
          CPX #$aa
          BCS Onnnn
                    ;CARRY SET IMPLIES THAT X GREATER THAN OR EQUAL TO aa


          LDA #>Onnnn-1
          PHA
          LDA #<Onnnn-1
          PHA       ;THE STACK NOW CONTAINS THE CORRECT ADDRESS FOR AN RTS TO
                    ;RETURN TO Onnnn, THE ADDRESS OF THE NEXT STATEMENT. THESE
                    ;4 INSTRUCTIONS WOULD BE OMMITTED IF ON...GOTO WAS USED
                    ;RATHER THAN ON...GOSUB.
          LDA THnnnn,X
          PHA
          LDA TLnnnn,X
          PHA
          RTS       ;THIS TRANSFERS CONTROL TO THE CORRECT LINE NUMBER. nnnn IS
                    ;THE HEXADECIMAL REPRESENTATION OF A COUNTER INCREMENTED BY
                    ;1 FOR EVERY ON...GOTO/GOSUB STATEMENT COMPILED.
THnnnn    .BYTE >L000A
          .BYTE >L0014
          .BYTE >L001E
          .BYTE >L0028
          .BYTE >L0032
TLnnnn    .BYTE <L000A
          .BYTE <L0014
          .BYTE <L001E
          .BYTE <L0028
          .BYTE <L0032

Onnnn
```

Figure b.

```
          ....
          ....
          ....
          ....      ;CODE TO EVALUATE THE BOOLEAN VALUE
          LDA FP1EXP
          BEQ Innnn
                    ;ASSUME WE CAN BRANCH AROUND THE REST OF THE LINE WHICH
                    ;MUST THEN TAKE LESS THAN 128 BYTES.
          ....
          ....      ;REST OF LINE CODE
Innnn

or        ....
          ....
          ....
          ....      ;CODE TO EVALUATE BOOLEAN VALUE
          LDA FP1EXP
          BNE +$03
          JMP Innnn
          ....
          ....      ;REST OF LINE CODE
Innnn
```

Machine Code Programming

PASSING VALUES BETWEEN BASIC AND M/C ROUTINES.

BASIC is a good language in which to develop programs but it runs compartively slowly. Machine code programs run quickly but they take far more time than BASIC to develop. One approach to resolving this dilemma is to write a hybrid program, most of which is written in BASIC but which uses machine code subroutines to speed up potential bottle necks.

The question then arises, "How does one pass values from BASIC variables to the machine code routine, and machine code values back to BASIC?". BASIC has a specially designed function called USR. However, USR passes a single numeric value in floating point format. Suppose you want to pass more than one item, or are concerned with strings or small integer values?

I use two alternative methods to USR. One needs POKE and PEEK ( ), the other uses a BASIC string as a machine code buffer area.

POKE and PEEK ( )

With POKE and PEEK ( ), bytes outside the normal BASIC area are used. The values to be passed are placed in them by the BASIC program, using POKE (location), (value) and the machine code routine called using SYS. The machine code routine the proceses the value as it would any other bytes of memory. If the machine code routine has information to pass back to BASIC, it places them in the dedicated locations, and returns to BASIC using 'RTS'. BASIC then accesses the values and puts them into variables, using PEEK (location).

Quite a good area to use is our old friend the second cassette buffer. I tend to use it for passed values even when the machine code routine is located at the top of user memory. It's best to keep to the bottom end of the buffer, 826 onwards, since one or two operating system routines in later PETs make temporary use of the upper bytes. Here's an example of how the method works. Suppose you have two integer values in the range 0 to 255 in variables A and B, and the requirement is to order them so that the higher of the two is in variable A and the lower in variable B. The Basic code is something like fig a

If you decided that the swop should be in machine code, then the approach would be along the lines of fig b

Where SYS 28000 calls a machine code routine derived from the source code, fig c

STRING BUFFER

Using a BASIC as a buffer is more involved, put allows you to pass up to 255 characters very conveniently. To understand the method you need to recall how MICROSOFT BASIC stores string variables.

Strings are held in two portions. One part, the string parameters, are included in the variables table which immediately follows the BASIC text. The parameters define the variable name and type, and for a string, where in high memory the actual characters are stored and how many of them there are. The other part is of course the actual string of characters stored in high memory.

The string parameter format is as in fig d.

BASIC sets up variable parameters in the table in the order that the variables are declared or met in the program. So if you set up A$ as the first variable in the program, the parameters for A$ will be the first ones in the variable table. BASIC maintains a pointer to the start of this table, and it's quite straight forward to use this pointer to find the parameters of the first string.

The machine code routine can examinine the string pointer to find where in memory the characters are, and at the string length to demonstrate how many characters it's dealing with.

As an example, suppose you wanted to change all upper case-characters in A$ to lower case, to allow two words to be alphabetically compared. The BASIC code is something like fig e.

To use a machine code routine, a string variable, say A$, must be declared as the first variable in the program and later filled with the character string to be converted, prior to calling

Figure a

```
IF B<A THEN C=A : A=B : B=C
```

Figure b.

```
POKE 826,A : POKE 827,B : SYS 28000 : A=PEEK(826)
P=PEEK(827)
```

Figure c.

```
COMP  LDA 826     ;Compare value in 826
      CMP 827     ;with the value in 827
      BMI SWOP    ;Swop if (827) is higher
      RTS         ;Otherwise return to BASIC
SWOP  LDX 827     ;Use Reg.X as transfer area
      STA 827     ;Put (826) into 827
      STX 826     ;Put (827) into 826
      RTS         ;Now exit to BASIC
```

Figure d.

```
Byte 1   Byte 2   Byte 3   Byte 4   Byte 5   Byte6   Byte 7
Variable name     length   pointer to str.    0       0
```

the machine code routine with SYS.

Typical coding for the machine code routine is as in fig f

MACHINE CODE TO A STRING

The same string parameters can also help with passing a string of characters resulting from a machine code program back into a BASIC string. A typical application would be a machine code routine which is receiving information via a telephone modem, and needs to feed the characters back to the main BASIC program.

Once again you need to declare a string variable as the first variable in the table. Suppose the characters are being collected into the second cassette buffer. Once the machine code routine has changed the pointer portion of the string parameter to point at 826, and the length portion to reflect how many characters are to be passed, the transfer is complete!

For example, suppose the machine code routine has received the message "TRANSMISSION COMPLETED." via the modem, and stored the ASCII characters in location 825.

See figure g over page

Figure e.

```
Z$=""
FOR Z= 1 TO LEN(A$)
Z$=Z$+CHR$(ASC(MID$(A$,Z,1))AND 127)
NEXT Z
A$=Z$
```

Figure f.

```
POINT  =  42
STRPTR =   0
LEN    = 825
;
*=826
;
SETUP  LDY #2          ; Start at third byte of table
       LDA (POINT),Y   ; Remember string length
       STA LEN         ; for later
       INY             ; Move on to next byte of table
       LDA (POINT),Y   ; Set up pointer to string
       STA STRPTR      ; low byte
       INY
       LDA (POINT),Y   ; and high
       STA STRPTR+1
       LDY LEN         ; Recover string length
       DEY             ; Adjust to deal with chr zero
CONVRT LDA (STRPTR),Y  ; Move backwards along string
       AND #%01111111  ; making bit 7 zero (lwr case)
       STA (STRPTR),Y  ; Put it back
       DEY             ; Move on to next character
       CPY #$FF        ; unless chr zero
       BNE CONVRT      ; has been processed,
       RTS             ; in which case exit to BASIC

.END
```

Mike Niklaus
Training Manager
Commodore U.K

Drop Out Proof Input

You may be interested in the enclosed program which is a drop-out proof input routine for 'NEW-ROM' PETs (BASIC 2). Written in machine code it is completely relocatable and is called by the USR function, eg to input a value to variable V the Basic program would include the line:- 100 V=USR 10). The variable may also be of Array type. eg A(I)=USR(0) is acceptable.

Only the numerical keys plus .+-g and E are accepted all others including the stop key are ignored, apart of course from carriage return and delete. A shifted carriage return will delete the whole number currently being entered.

As I have no printer I am unable to provide a dissassembled listing of the machine code but the Basic operation is as follows. The character count pointer in $SI is initialised to zero and the cursor enabled. A character is obtained by the GET routing $FFE4 and checked for type. If acceptable it is added to the buffer at $0200 - and also printed to the screen using $FFD2.

On receipt of a c.return a 00 is added to the buffer, the contents of pointer $77,$78 are saved and the cursor is turned off. A jump is made to $CB3C in the BASIC input routine which converts the string in $0200 - to floating point and leaves it in the variable in the calling USR function.

Below are 2 more short machine

Figure g.

```
POINT  =  42
BUFFER = 826
CHRCNT = 825
;
SETUP  LDY #2          ;Point at length byte of parameter
                              LDA    CHRCNT         ;Copy
character count to length
       STA (POINT),Y
       INY             ;Point at location bytes.
       LDA #<BUFFER    ;Adjust location to point to
       STA (POINT),Y   ;buffer (low byte)
       INY
       LDA #>BUFFER    ;high byte
       STA (POINT),Y
       RTS             ;That's all there is to it!
```

Figure 1.

```
.: 033A 20 F8 CD 20 78 D6 86 D8
.: 0342 20 F8 CD 20 78 D6 86 C6
.: 034A 20 5D E2 20 F8 CD 4C AB
.: 0352 C9 xx xx xx xx xx xx xx
```

Figure 2.

```
10 P=826 : PRINT"↓CLR↓"
100 FOR C = 15 TO 25
110 FOR R = 10 TO 15
120 SYS P,R,C,"↓RVS↓ ↓OFF↓"
130 NEXT R,C
```

Figure 3.

```
.: 033A 20 29 E2 20 A6 F5 20 3C
.: 0342 F6 20 CA FD 20 6A E7 A5
.: 034A C9 85 FB A5 CA 85 FC 20
.: 0352 CD FD 20 6A E7 20 D0 FD
.: 035A D0 E1 xx xx xx xx xx xx
```

code programs I have found useful and which your readers may find useful also.

They are both for 'NEW-ROM' PETs (BASIC 2.0). The first programs will print an expression (numerical or string) at any point on the screen without the need for expressions such as PRINT LEFT$ (DN$,10)TAB(20)M$. The syntax is SYSP,R,C,X where P is the short address of the code (eg 826) R is the row in which to print (R=0 -24) C is the column where the first character is printed X is the expression to be printed. It can be of any type e.g. SIN(N), 27,N"123", A$, A1, A$(N) are all valid. The commas between P R C and X are essential if the expression X is followed by a semi colon the cursor will be left after the expression, otherwise it will be at the start of the next line.

The program is fully relocatable and is given if figure 1 in the form of a hex dump. It occupies 25 bytes.
Also by including the routine in For-Next loops it is easy to draw borders or to fill in areas of the screen with any graphics character. (fig 2)
will print a block of reversed spaces in the middle of the screen.

The second program will search through a tape and print out for each program found the following information
1) Program name (if any)
2) Start address of program
3) End address of program
It is relocatable and is called simply by SYS S where S is the start address of the program (probably 826). If no previous tape has been loaded or saved then it is necessary to press LOAD then (STOP) to set up the cassette load routines for TAPE // 1 (otherwise a 'PRESS PLAY ON TAPE **//0'** message is printed). The code is as figure 3.

BASIC PROGRAM TO POKE DATA

```
100 REM INPUT ROUTINE LOADER
110 INPUT"[CLR] START ADDR (DECIMAL)";SA:A=SA
120 READ D:IF D=-1 THEN 140
130 POKE A,D:A=A+1:GOTO 120
140 PRINT "PROGRAM LOADED"
150 HI= INT(SA/256)
160 POKE 2,HI
170 POKE 1, SA-256*HI
180 FOR I=1 TO 1000:NEXT
190 :
200 REM INPUT MACHINE DEMONSTRATION
210 ?"[CLR]"
220 PRINT "ENTER VALUE";:V=USR(0)
230 PRINT "VALUE WAS" V
240 PRINT: PRINT: GOTO 220
```

DATA FOR INPUT ROUTINE

```
1000 DATA 32,128,210,162,0,134, 161,134,
     11,162,0,134,167,32,228,255,201,58,176.
1010 DATA 24,56,233,43,56,233,213,176,16,201,
     47,240,233,32,210,255,166,161.
1020 DATA157,0,2,230,161,208,221,166,161,
     240,211,201,69,240,236,201,20,208.
1030 DATA 5,198,161,32,210,255,201,141,
     240,29,201,13,208,196,169,32,133,170.
1040 DATA 32,210,255,32,213,201,165,119,
     164,120,133,72,132,73,169,1,133,167.
1050 DATA 76,60,203,169,20,32,210,255,
     198,161,208,249,240,154,-1.
```

Peripheral Spot

```
    DISK MEMORY DISPLAY - DOS 2.0

100 PRINT"■■DISK MEMORY DISPLAY     JIM BUTTERFIELD"
105 L=8:IFPEEK(32848)=4THENL=16
110 DATA77,45,87,0,18,16,162,0,189
120 DATA157,64,06,232,224,16,208,245,108,2,252
130 FORJ=1TO9:READX:C$=C$+CHR$(X):NEXTJ
140 FORJ=1TO11:READX:D$=D$+CHR$(X):NEXTJ
150 PRINT"■THERE ARE TWO PROCESSORS:"
160 PRINT"  1) THE IEEE PROCESSOR;"
170 PRINT"  2) THE DISK PROCESSOR;"
180 INPUT"WHICH DO YOU WANT TO PEEK (1 OR 2)";D
190 PRINT"INPUT MEMORY ADDRESS"
200 PRINT"IN HEXADECIMAL:":OPEN1,8,15
210 P$=CHR$(4)+CHR$(16):R$=CHR$(224)
220 PRINT"  ■■■■                                    ■"
230 INPUTZ$
240 PRINT"■";:IFLEN(Z$)<>4THENGOTO220
250 FORJ=1TO4:Y=ASC(MID$(Z$,J))
260 IFY<58THENY=Y-48
270 IFY>64THENY=Y-55
280 IFY<0ORY>16GOTO220
290 Y(J)=Y:NEXTJ:K=0:PRINT"■■■■■■■";
300 U=Y(3)*16+Y(4):V=Y(1)*16+Y(2)
310 IFD<>2GOTO360
320 PRINT#1,C$;CHR$(U);CHR$(V);D$
330 PRINT#1,"M-W";P$;CHR$(1);R$
340 PRINT#1,"M-R";P$:GET#1,X$:IFX$=R$GOTO340
350 U=64:V=18
360 PRINT#1,"M-R";CHR$(U);CHR$(V)
370 GET#1,X$:IFX$=""THENX$=CHR$(0)
380 PRINT" ";:X=ASC(X$)/16
390 FORJ=1TO2:X%=X:X=(X-X%)*16:IFX%>9THENX%=X%+7
400 PRINTCHR$(X%+48);:NEXTJ
410 U=U+1:IFU=256THENU=0:V=V+1
420 K=K+1:IFK<LGOTO360
430 Y(0)=0:Y(4)=Y(4)+L:J=4
440 IFY(J)>15THENY(J)=Y(J)-16:J=J-1:Y(J)=Y(J)+1:GOTO440
450 PRINT:PRINT"  ";:FORJ=1TO4:Y=Y(J):IFY>9THENY=Y+7
460 PRINTCHR$(Y+48);:NEXTJ:PRINT"■":GOTO220
READY.
```

COPY/ALL will copy relative files correctly. It replaces COPY.ALL which contains a bug; it does not replace COPY ALL which is much faster if relative files are not involved.

The Basic listing mixes upper and lower case for readability; type it in without shifting. There should be lots of space for the following Machine Language program.

```
100 print"[clr] DISK.COPY.ALL       JIM BUTTERFIELD"
110 dim L2(232),L1%(232),N$(232),T%(232),T$(4)
120 data XXX,SEQ,PRG,USR,REL
130 forJ=0to4:readT$(J):nextJ
140 input"FROM UNIT  8[3left]";F
150 gosub800
160 F$=D$
170 input"TO UNIT  9[3left]";T
180 gosub800
190 T$=D$
200 ifF=T andF$=T$thenrun
210 gosub860
220 close1:close15:open 15,F,15:print#15,"i"+F$
230 gosub830:if E then stop:goto220
240 input"PATTERN  *[3left]";P$
250 forJ=1tolen(P$):ifmid$(P$,J,1)<>"*"thennextJ
260 P1=J-1:ifP1>0thenP$=left$(P$,P1)
270 print"HOLD DOWN 'Y' OR 'N' KEY TO SELECT"
280 print"PROGRAMS TO BE COPIED..."
290 open 1,F,3,"$"+F$
300 gosub830:ifEthenstop:goto220
```

Continued

```
310 get#1,A$:A=asc(A$+" ")
320 ifA=1orA=65thenL1=253:goto350
330 ifA=67thenL1=761:goto350
340 close1:print"?????":stop
350 forJ=1toL1:get#1,X$:nextJ
360 N=0:N1=0:R=255:Z=89
370 N$="":gosub1020:T9=Y-128:gosub1000
380 J1=1:Z$=chr$(160):forJ=1to16:get#1,X$:N$=N$+X$
390 ifX$<>Z$thenJ1=J
400 nextJ
410 gosub990:L1%=Y
420 gosub980:gosub980
430 L2=X+256*Y
440 ifT9<1orT9>4goto540
450 ifP1>0thenifP$<>left$(N$,P1)goto540
460 printN$;" ";T$(T9)
470 p=peek(151)andR
480 getZ$:ifZ$=""andP<255goto520
490 ifZ$="Y"orZ$="N"thenZ=asc(Z$):R=255:goto520
500 ifZ$=chr$(13)thenR=0:goto520
510 goto480
520 ifZ<80thenprint"[down]                                    [down]":goto540
530 N=N+1:L2(N)=L2:N$(N)=left$(N$,J1):T%(N)=T9:L1%(N)=L1%
540 ifST>0goto570
550 N1=N1+1:ifN1=8thenN1=0:goto370
560 gosub1000:goto370
570 close1:close15:print" * * * * *"
580 forJ=1toN
590 L2=L2(J):T%=T%(J):ifL>L2goto640
600 print"*** output disk full"
610 input"do you have a new one";Z$
620 ifasc(Z$)<>89thenend
630 gosub860:goto590
640 open14,F,15:open15,T,15
650 printN$(J);left$("                    ",17-len(N$(J)));
660 open3,F,3,F$+":"+N$(J)+","+T$(T%)
670 input#14,E,E$,E1,E2:gosub840:ifEthenprint"** ";E$;E:goto750
680 ifT%=4thenopen4,T,4,T$+":"+N$(J)+",L,"+chr$(L1%(J)):goto700
690 open4,T,4,T$+":"+N$(J)+","+T$(T%)+",W"
700 L=L-L2:gosub830:ifEthenprint"*** ";E$;E:goto750
710 ifT%=4thensys3312:goto730
720 sys3272
730 N$(J)="":gosub830:ifEthenprint"**** ";E$;E:goto750
740 print"[down]"
750 close4:close3:close15:close14
760 nextJ
770 X=fre(0):input"another input disk ready";Z$
780 ifasc(Z$)=89goto220
790 end
800 input"drive  0[3left]";D
810 ifD*D<>Dgoto800
820 D$=chr$(D+48):return
830 input#15,E,E$,E1,E2
840 ifE=0thenE=(ST and 191):E$="*st*"
850 return
860 open15,T,15:input"want to new the output disk  n[3left]";Z$
870 ifasc(Z$)<>89goto910
880 input"disk name,id";X$,Y$
890 print#15,"n"+T$+":"+X$+","+Y$
900 gosub830:ifEthenstop:goto860
910 print#15,"I"+T$:open1,T,0,"$"+T$+":!#$%&"
920 gosub830:ifEthenstop:goto860
930 gosub980:gosub980
940 get#1,X$:ifX$<>""goto940
950 gosub980
960 L=X+Y*256:print"(";L;"blocks free )"
970 close1:close15:return
980 get#1,X$
990 get#1,X$
1000 get#1,X$
1010 X=len(X$):ifXthenX=asc(X$)
1020 get#1,X$:Y=len(X$):ifYthenY=asc(X$)
1030 return
```

Continued

Machine Language portion of program:

```
.
.:  0cc8 a2 03 20 c6 ff 20 73 0d
.:  0cd0 8e 40 02 a5 96 8d 41 02
.:  0cd8 20 cc ff a2 04 20 c9 ff
.:  0ce0 ae 40 02 20 61 0d ad 41
.:  0ce8 02 f0 da 85 96 4c cc ff
.:  0cf0 a9 00 85 5f 85 60 20 cc
.:  0cf8 ff e6 5f d0 02 e6 60 a2
.:  0d00 0e 20 c9 ff a2 50 20 61
.:  0d08 0d a2 03 20 61 0d a6 5f
.:  0d10 20 61 0d a6 60 20 61 0d
.:  0d18 a2 01 20 61 0d 20 cc ff
.:  0d20 a2 0e 20 c6 ff 20 73 0d
.:  0d28 20 cc ff e0 30 f0 01 60
.:  0d30 a2 03 20 c6 ff a0 ff c8
.:  0d38 20 73 0d a4 61 8a 99 7a
.:  0d40 02 a6 96 f0 f2 84 62 20
.:  0d48 cc ff a2 04 20 c9 ff a0
.:  0d50 ff c8 b9 7a 02 aa 20 61
.:  0d58 0d a4 61 c4 62 d0 f2 f0
.:  0d60 95 84 61 a0 10 8a 20 d2
.:  0d68 ff a5 96 29 03 f0 03 88
.:  0d70 d0 f3 60 84 61 a0 10 20
.:  0d78 e4 ff aa a5 96 29 03 f0
.:  0d80 03 88 d0 f3 60 aa aa aa
.
```

Save with Machine Language Monitor:

```
     .S "0:COPY/ALL",08,0401,0d85
```

After program has been saved, it may be loaded and saved normally with Basic commands.

PRINT ASCII FILES

```
100 PRINT"■PRINT ASCII FILES    JIM BUTTERFIELD"
110 INPUT "DISK FILE NAME";F$
120 OPEN8,8,8,F$
125 INPUT"SCREEN OR PRINTER";D$
126 D=ASC(D$):IFD=80GOTO130
127 IFD<>83GOTO125
128 POKE59468,14:PRINT"■";:OPEN4,3:POKE136,0:GOTO140
130 OPEN4,4:POKE136,17:PRINT#4,"LISTING, FILE:  ";F$:PRINT#4,"■";
140 SYS1536
150 PRINT#4
160 CLOSE4:CLOSE8
READY.
```

THE MACHINE CODE PART

```
?
.
.:  0600 A2 08 20 C6 FF 20 E4 FF
.:  0608 A6 96 08 AA 20 CC FF 8A
.:  0610 29 7F C9 7F F0 3A C9 1F
.:  0618 B0 06 C9 0D F0 14 D0 1A
.:  0620 C9 61 90 04 29 5F D0 0A
.:  0628 C9 41 90 06 C9 60 B0 02
.:  0630 09 80 A2 04 20 C9 FF 20
.:  0638 D2 FF C9 0D D0 0F A5 9E
.:  0640 29 01 D0 FA 85 9E A5 88
.:  0648 F0 03 20 D2 FF 20 CC FF
.:  0650 20 E1 FF 28 F0 AA 4C CC
.:  0658 FF 46 80 0B F3 7F 00 00
.:  0660 44 80 01 F0 7F 00 00 44
.:  0668 00 87 26 00 00 00 AA AA
.:  0670 AA AA AA AA AA AA AA AA
.
.?
```

Using the User Port

A.H. Potten

What is the "User Port" and what is it for!
Let's take a look at the back of your PET ..... you should see something like Figure 1. The slot marked "Parallel User Port" - J2 is what you need and an edge-on view, somewhat enlarged is shown at the bottom of Figure 1. If you need a technical description of the Port then this article is NOT for you. Consult the CBM Manual or "The PET Revealed" or one of the many other publications which give as much detail as you may need. THIS article is strictly for beginners.

The user port enables you to have two-way communication with the PET without using the keyboard, cassette, screen, disk unit or printer (apart of course from writing or entering a program). Since it can be a two-way procedure it is often referred to as Input/Output or I/O for short. The IEE interface also allows I/O to take place and is much more powerful but at the same time much more difficult to get to grips with ! Nevertheless the user port enables you to undertake a wide variety of tasks such as switching and/or monitoring or eight channels, sound generation, data collection, counting, process control and similiar activities.

In order to use the port a connector is needed, and it must be the correct type. I was sold one by a radio-spares dealer as being suitable for a PET only to find that it was double-sided. The top and bottom contacts were connected. So you have been warned...make quite sure that you know what you are doing since connections to the top row of contacts can do nasty things to the insides of your computer. And while on the subject, make sure that you never input a voltage to the PET greater than 5 volts and remember that the maximum output is of the order of 200 milliamps. You may also be a little confused by the "identification" system. Figure 1. shows that the top row of pins (contacts) are numbered 1 to 12 which is clear, however the bottom row is A to N omitting G and I (!). Not so clear is the fact that these contacts are also referred to as PA 0 to 7 CAl, CB2 and there two digital grounds. For full explanation see CBM Manual p.79 but the following tells you the relevant bits :-

A - Digital ground
B - C A 1
C - P A O
D - P A 1
E - P A 2
F - P A 3
H - P A 4
J - P A 5
K - P A 6
L - P A 7
M - C B 2
N - Digital ground

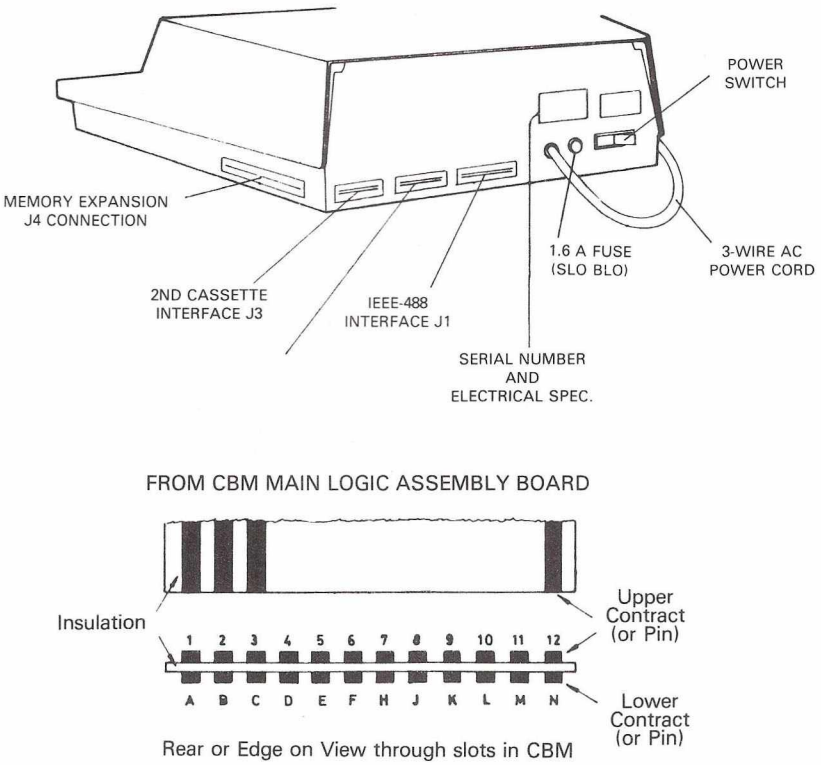
You may well recognise CB 2 since sound output is often taken from contacts M and N i.e. from CB 2 and ground.

My first attempt to "come to grips" with the subject was to buy "The User Port Cookbook" from CBM. Apart from, the expected difficulties due to it's American orientation, the misprints (Versatile Inter**free** Adaptor), plan of the pin-out (viewd for the top with the explanation that "The user port ins are on the bottom "...yes" user port ins", I was finally discouraged by the statement that the suggested circuit (not easy to follow) draws about 200 MA which is close to the maximum available from the PET. The accompanying cassette I found equally unhelpful since it is simply a tape version of the printed program in the "Cookbook" which is not explained ! The final straw is the poorly reproduced technical description of the VIA (Versatile Interface Adaptor) which I will not attempt to explain !

So when I saw Mektronic Consultant's advertisement for the COMMUNIKIT which plugs in to the PET and "accepts inputs through the user port and allows outputs "and by means of simple BASIC PEEKS and POKES allows it to perform as a versatile controller I promptly bought a kit.

The Communikit has been designed to be extremely simple to construct and will only take around 90 minutes for a beginner equipped with only a soldering iron and a few tools to

Figure 1



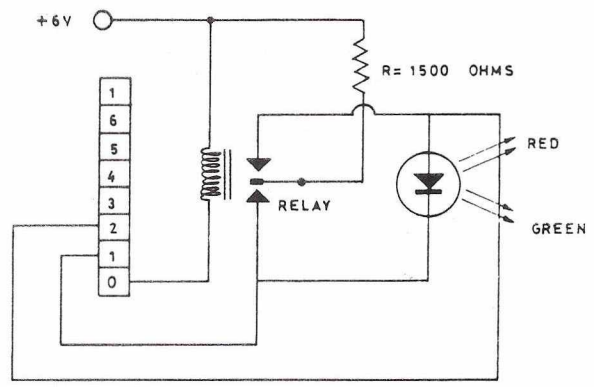
assemble.

It will handle up to 8 inputs or outputs directly, without the need to use the special control lines CA 1 and CB 2. With the use of simple handshaking procedures the effective capacity can be expanded virtually indefinitely.

The unit is powered by an external battery of anything in the range 8 to 24 volts and is capable of driving up to 500 milliamps per channel (subject to a total maximum of 2.5 amps).

L.E.D.'s indicate the status of each channel.

I am pleased to say that I found the Kit as easy to construct as the makers claimed, and the instructions seemed clear and comprehensive. Since the connector came ready-wired mistakes are unlikely at this stage and in fact I only experienced one problem - a faulty L.E.D. Almost unheard of I have been informed. I did of course take the precaution of testing all components as far as possible before asembly and the L.E.D. was OK at this stage but if failed when the circuit was completed and gave me some anxious moments since I could not at first believe that this could be the trouble. A minor problem was that during testing the instructions state




```
10 A=59459:B=255:C=254:D=252
20 E=251
30 POKE A,B:PRINT "PRESS SPACE BAR"
40 POKE A,C:FOR I=1 TO 250:NEXT
50 POKE A,D:FOR I=1 TO 250:NEXT
60 POKE A,B:FOR I=1 TO 500:NEXT
70 POKE A,E:FOR I=1 TO 500:NEXT
80 N=N+1:IF N<5 GOTO 290
90 WAIT 59410,4,4
100 N=0:GOTO 30
```

Figure 2

that at one stage ALL the I.E.D.'s should be out, this should read ALL but CB 2.

The circuit diagram is given in figure 3

You do need to know a little about the DDR (no not a German abbreviation but the Data Direction Register). This register is held at location 59459 and the decimal value can be anything from 0 to 255. As you will have gathered when it is 0 all the L.E.D.'s are lit (which means that the contacts are at logic level 1 or 5 volts) and when the value is 255 they are all out (logic level 0 or 0 volts).

This can best be explained in a table (fig a) ***over page***

If you want every other channel to be an input (which is indicated by alternate L.E.D.'s lit) then it should be obvious that one of the two patterns which will do this is (fig b) ***over page***

Figure 3

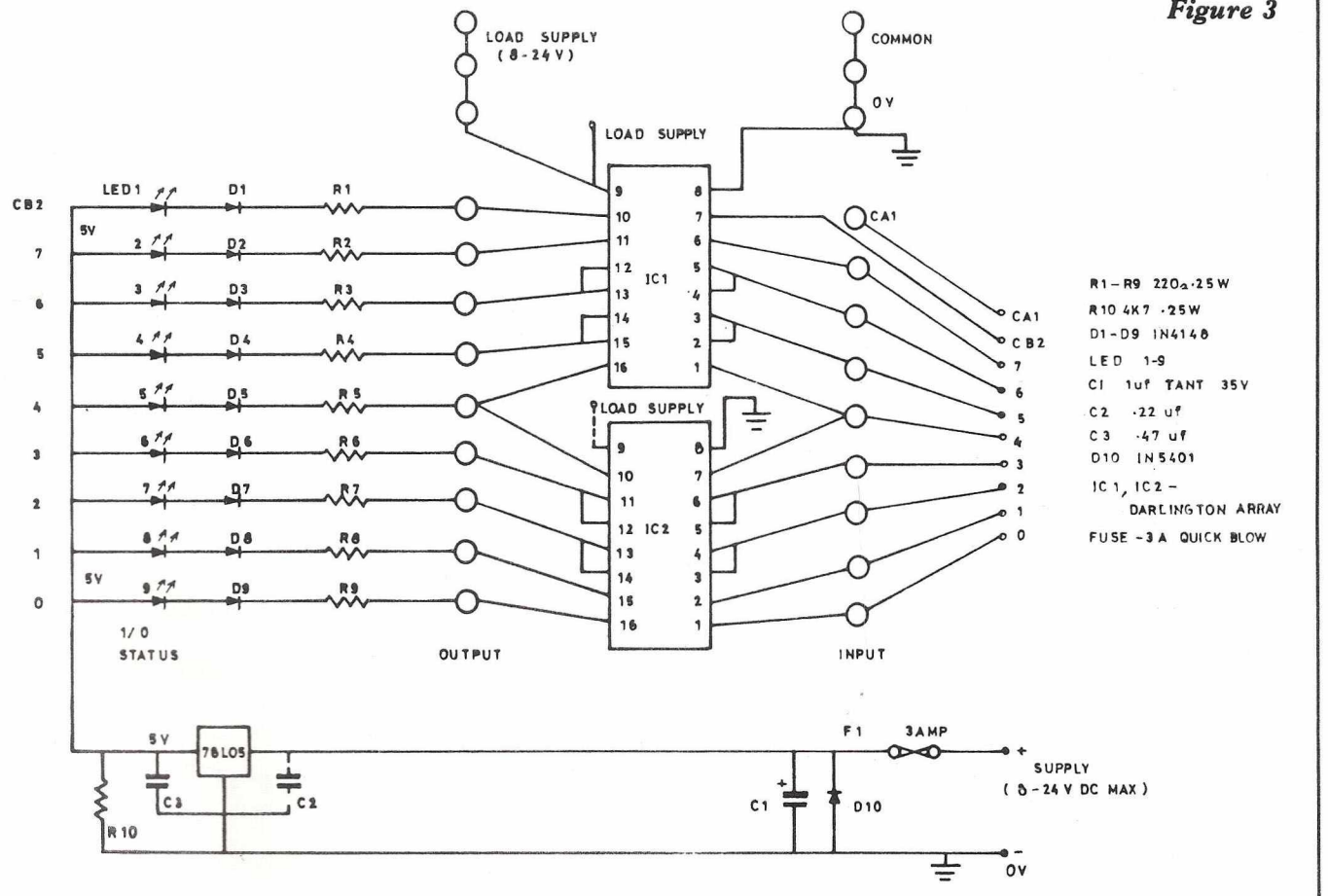


fig a.

Channel numbers	7	6	5	4	3	2	1	0	(i.e. PA 0-7)
Decimal 0	0	0	0	0	0	0	0	0	(i.e. binary 0)
Decimal 255	1	1	1	1	1	1	1	1	(i.e. binary 255)

fig b.

```
0 1 0 1 0 1 0 1  ( i.e. binary 85 )
( 64 + 16 + 4 + 1  = 85 )
```

fig c.

```
5 REM:...BINARY COUNT...
10 POKE59459,255    (INITIALISES)
20 J= 59471  (A CONSTANT AVOIDS ERRORS)
30 FORK=0TO255
40 POKEJ,K
50 FORL=1TO100:NEXT  (DELAY LOOP)
60 NEXTK
70 GOTO30
```

The first check I made was for current consumption using a robust milliameter (120 ma. f.s.d.). This was connected together with a switch as shown in Figure 4 and proved to be very useful. I found that my unit used 4.5 ma. with a 12 volt supply, appreciably less than the maximum of 50 ma. given in the instructions. For most of my subsequent experiments I used a 6 volt supply which gave me satisfactory results. With all L.E.D.'s lit, two small 6 volt relays a P.O. relay and a solid state alarm operating I was still only using 120 ma.

And so when all seemed to be in order I gave the first command POKE 59459,255 and all the L.E.D.'s went out (except CB2) just as they were supposed to do!

Thus the command POKE 59459,85 wil do just what you need. If you refer back to Figure 4 it should now be clear that the line of L.E.D.'s indicate the status of each channel, be it input or output.

Despite the makers instructions that the voltage applied to the Kit should be in the range 8 to 24 volts I found that I got quite satisfactory results using a 6 volt battery, the L.E.D.'s being sufficiently bright. I felt that while experimenting an accidental mis-connection at 6 volts was much less likely to do damage than at 24 volts. I later found that even using 4.5 volts (an ordinary dry battery) the unit still operated in all respects except that now the L.E.D.'s were distinctly dim.

Having got this far it is easy to write programs which will flash the L.E.D.'s in any patterns you can think of, or you can arrange for them to "count" in binary (fig c)

The next stop is to switch a load of some sort. For example a lamp of the same voltage as your external supply which will not take more than 500 ma. is connected across from one of the PA terminals to the top terminal(s) which should be at your voltage.

My first project was to operate a programmed Xmas tree which had two sets of fairy lights, several individual lamps, and some stars which could be rotated by means of an air pump! I used two relays to switch the fairy lights on and off (240 volts), a P.O. relay to switch the air pump (also 240 volts), and the 6 volt lamps were directly switched. The program displayed a Xmas tree and the appropriate greeting and played an excruciating version of the "First Nöel"

Naturally you should take special care when switching mains voltages. There is not only the danger of electric shock, you may get induced voltages in the wiring, mains spikes or back EMF's from relays. These may only corrupt your program if you are lucky ...if not you may have a repair bill. So if in doubt seek advice FIRST.

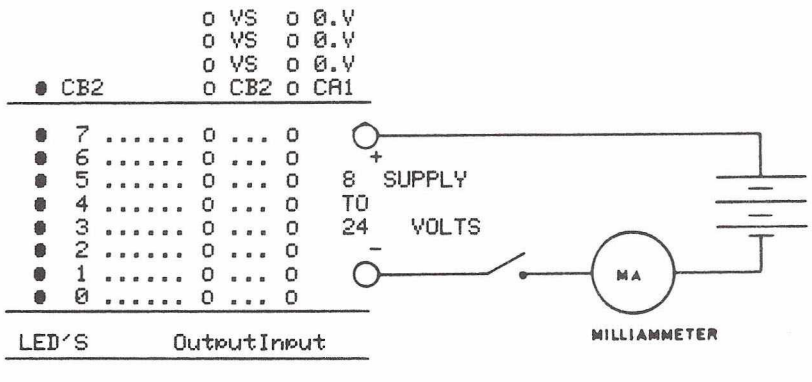
I then devised a circuit, see Figure 2 which reversed the polarity of the supply to a bi-colour L.E.D.'s. A slightly risky experiment since failure would have resulted in a dead short. However all was well and the L.E.D. obediently flashed red and green. Magic ? Well no, but quite a boost to my confidence!

When first initialised all the output channels are at 0 volts and all the input channels are at 5 volts. The status of the input channels can be read from the request PEEK(59471) and the decimal answer is 255. If you short one or more of the channels to ground (common or earth) then PEEK(59471) becomes the corresponding decimal number of the binary "pattern" which you have created. I connected a set of eight switches to the input terminals and thus could enter any desired combination or number. Connections are conveniently made using "ribbed cable". This way my most expensive component (most of my components are ex-T.V. or surplus) until I became aware of surplus computer dealers and cheap second-hand cable.

Continued next month

Figure 4

Here is a diagram of your Communikit :-



INVADERS IN ROM!

They said it couldn't be done! We've managed to put INVADERS into ROM for the 8032 - and it uses the full width of the screen. All you need to do is unplug the chip in UD7 - it's not soldered in - and replace it with our new ROM INVADERS chip. Normal operation of the 8032 is entirely unaffected, but type SYS 59648 and you'll find yourself struggling to survive against a massive alien fleet. When you hear the boss coming just press the escape key and then - well, back to work!

ROM INVADERS makes full use of the built-in 'bleeper' to add appropriate sound effects. It's the most self-contained game ever, and at £19.95 plus VAT really good value. Other SUPERSOFT arcade games are ASTEROIDS, SUPER GLOOPER, METEORITES, and GIDDY GHOULS - they cost £8 plus VAT each and run on all 40 and 80 column machines except Old Rom.

HIGH RESOLUTION GRAPHICS

The HR-40 High Resolution Graphics Board is easy to fit - no soldering, no tracks to cut, no links to make or break. The resolution of 320 by 200 is as good as you'll get, and the picture is rock steady. It has 8k of its own RAM, and utility software in EPROM, so you'll have a full 8, 16 or 32k of memory for your own programs. Don't worry about those plug-in chips you've bought - sockets UD3 and UD4 are duplicated on the board!

Fully assembled, the HR-40 board fits any large keyboard PET/CBM computer other than the new 12in models. If you have a 'FAT-40' machine then you should order the HR-40B version, but they're both the same incredible price, just £149 plus VAT. The HR-80 board for the 8032 should be available during December - write or phone for up to the minute information.

DON'T BUY A WORD PROCESSOR!

That's right - don't buy a word processor until you've seen MicroScript, the new ROM-based word processor for the 8032. Because it's in ROM up to 30000 bytes are available for text - that's nearly three times as much as with other systems - and it's not only easier to learn, but faster in operation!

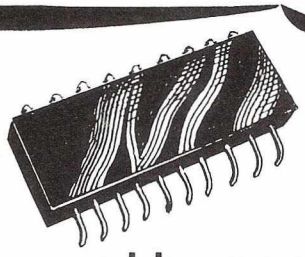
Ask your dealer to demonstrate how easily MicroScript handles standard letters - and how the exact format of any document can be displayed on the screen. You'll notice too that there's no need to wait whilst documents are being printed - once you've started a print run you can continue to use the computer and disk regardless!

..... BUT ON THE OTHER HAND

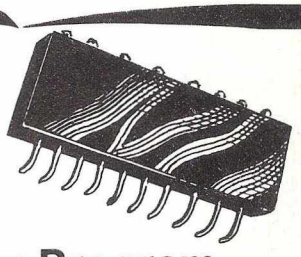
If word processing is a relatively minor task for your PET/CBM system you're probably not prepared to spend hundreds on a word processor, however sophisticated. PAPERMATE is a word processor with most of the features you'll find on more expensive programs - yet it costs just £35 on tape or £36.50 on disk. All you need to use PAPERMATE is 16k of memory - whether you've got 40 or 80 columns, disk or tape just doesn't matter.

SUPERSOFT

First Floor, 10—14 Canning Road, Wealdstone,
Harrow, Middlesex, HA3 7SJ, England
Telephone: 01-861 1166



audiogenic LTD
CHIP SHOP



EDEX 2.0 & 4.1

adds commands to BASIC for use within your Program

IF THEN ELSE ● PLOT ● BEEP ● PRINTUSING ● SWAP
MERGE ● HARD COPY ● PLUS A RANGE OF TOOLKIT
TYPE FUNCTIONS AND A FAST EDITING SYSTEM

EDEX is an extension to BASIC which considerably enhances the potentialities of the Commodore PET/CBM
It consists in a 4K-BYTE ROM which installs inside the PET/CBM.

EDEX is compatible with Commodore disk devices as well as with the DOS Support Program.
EDEX operation is fully transparent towards the Microsoft Basic Interpreter
EDEX is fully compatible with prior programs written without EDEX.

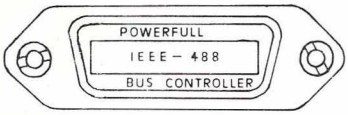
AUTO
Activates automatic line numbering.
APPEND *
Allows the creation of a program with a subroutine library
BEEP
Gives a sound of programable pitch and duration
CALL
Calls a machine language subroutine with transmission of up to 16 arguments
DELETE
Allows multiple line suppression
DUMP
Lists all variables in a program, together with their values
EDITING *
e.g. @ M prints MID$
ERROR
Shows where an error has occurred
FIND
Lists all lines where a given character string is present
HARD COPY □
Dumps screen to printer
IF THEN ELSE
With up to 16 nested conditions
MERGE □
Merge two programs files
PLOT
Plots curves of 50 x 80 or 160 resolution
PRINTUSING
Formats printing on screen or any printer
RENU
Program renumbering
RESET
Suppresses a dot (contrary of PLOT)
SWAP □
Swap one program for another keeping variables
TRACE □
Single line execution (displayed at top of PET)

*** EDEX 2.0 only □ EDEX 4.1 only**

EDEX 2.0 for use with BASIC 2 40 Column Pets **£39.50**
EDEX 4.1 for use with 80 Column Pets **£49.50**

Available shortly for BASIC 4 40 Column PETs

IEEE-488 PACK



The end of instrumentation's problems. It resolves all kind of troubles:
- Time-out
- Special characters ("null", and so on...)

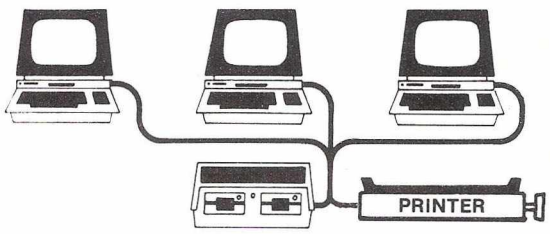
IEEE-PACK allows the use of IEEE-488 Universal Commands:

- DCL	(Device clear)	- SDC	(Selective device clear)
- SPE	(Serial poll enable)	- SPD	(Serial poll disable)
- LLO	(Local lockout)	- GTL	(Goto local
- PPL	(Parallel poll configure)	- PPU	(Parallel poll unconfigured)

IEEE-PACK also allows BASIC interrupt with functions:
- ONKEY "x", line number
- ONSRQ line number (On Service Request)

IEEE-PACK comes complete with two ROMs. **£89.50**

MULTEX



MULTEX allows several CBM 8032 to work together on the same peripherals.

MULTEX is a ROM which replaces a ROM of the CBM 8032.

Except the substitution of this ROM no other modification is required on the CBM 8032.

MULTEX is much cheaper than any other system.

MULTEX £69.50

Access DINERS

ALL PRICES INCLUDE V.A.T. & P.P.
AVAILABLE FROM ALL GOOD DEALERS OR DIRECT FROM

AMERICAN EXPRESS BARCLAYCARD VISA

AUDIOGENIC, P.O. Box 88, 34-36 Crown Street, Reading, Berks. Tel: Reading (0734) 595269