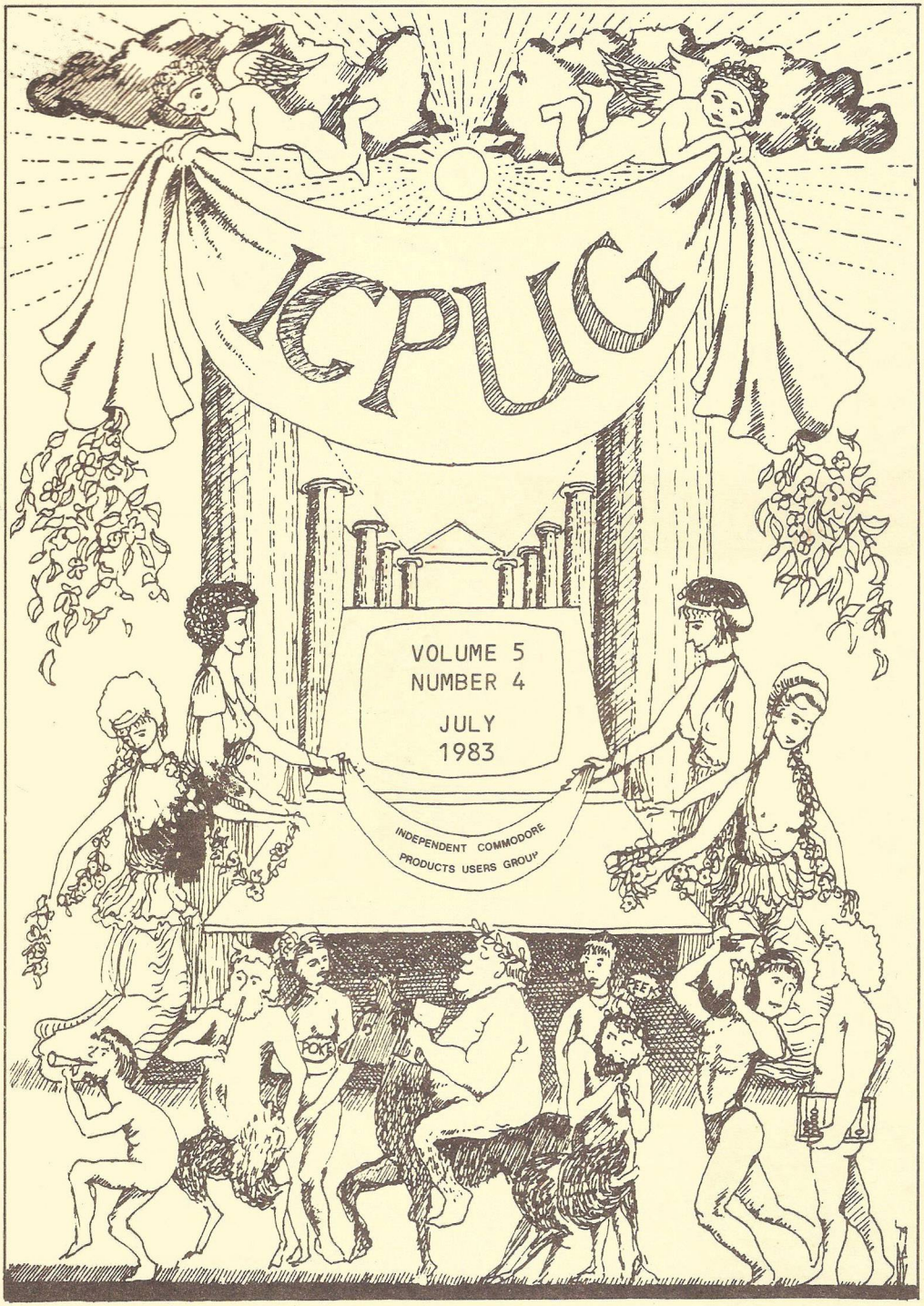

ICPUG
VOLUME 5
NUMBER 4
JULY
1983
INDEPENDENT COMMODORE
PRODUCTS USERS GROUP
POKE
PEEK

HONORARY NATIONAL OFFICIALS

CHAIRMAN	Wing Cdr. Mick Ryan 164 Chesterfield Drive Riverhead Sevenoaks, Kent TN13 2EH Telephone: Sevenoaks (0732) 453530
TECHNICAL QUERIES SECRETARY	Jim Tierney 11 Collison Place Tenterden Kent TN30 7BU Telephone: 058 06 2711
REGIONAL CO-ORDINATOR	Terry Devereux 32 Windmill Lane Southall Middlesex UB2 4ND
TREASURER	Joseph Gabbott
SOFTWARE LIBRARIAN	Bob Wood 13 Bowland Crescent Ward Green Barnsley, South Yorks S70 5JP Telephone: (0246) 811585 (work) (0226) 85084 (home)
MEMBERSHIP SECRETARY	Jack Cohen 30 Brancaster Road Newbury Park Ilford, Essex IG2 7EP Telephone: 01 597 1229
EDITOR	Ron Geere 109 York Road Farnborough Hants GU14 6NQ
VIC CO-ORDINATOR	Mike Todd 27 Nursery Gardens Lodgefield Welwyn Garden City Herts AL7 1SF
DISCOUNTS OFFICER	John Bickerstaff 45 Brookscroft, Linton Glade Croydon CR0 9NA Telephone: 01 651 5436
ASSISTANT EDITOR	Tom Cranstoun

INDEPENDENT COMMODORE PRODUCTS USERS GROUP

VOL 5
No 4

Newsletter

JULY
1983

Europe's first independent magazine for PET users

EDITOR'S NOTEBOOK

September may seem a long way off, but I feel bound to re-iterate the substance of my editorial of last September's issue. There were no takers for the post of editor and it was perhaps fortunate that I was able to continue. This year we are not so fortunate and I shall not be in a position to be Editor after the September issue.

Our membership is higher than it has ever been, sc there should be someone out there who can collate contributor's text onto disk, arrange it in the format of the magazine and print out one copy from which the printers can reproduce and distribute the magazine. Contact me soon if you feel that you can undertake the commitment and I will pass on the details of the technique that I adopt. In the event of more than one member coming forward, we shall have the pleasure of seeing someone democratically elected at the Annual General Meeting.

Many new products made their debut at the Commodore Show and it takes quite some time to digest the information. Rather than overload this issue, I have deferred mention of many of the items until next issue. One such product is BC BASIC which eases the problem of producing sound and pictures on the C-64, and it will be reviewed next issue.

R.D.G.

--oOo--

2031 DISK DRIVE NOTES

By D.Viner.

I have now been the owner of a 2031 disk drive for around three months and, after reading about all the trouble some owners have had, I would like to make a few points of my own. As my PET has been upgraded to BASIC4.0 (it started out life as old ROM but with the aid of a new home-built ROM board, some EPROMs and a lot of track cutting I am now among the happy ranks of the 4.0s) and I have had none of the trouble that BASIC2.0 owners have come across. The only reported bug, the occasional corrupted directory listing, occurs very seldom and doesn't appear to cause any other trouble.

Obviously, as the 2031 is only a single drive, some of the disk commands do not work. BACKUP just returns a syntax error (disk error 31) and any attempt to access the non-existent drive 1 via any other command (COPY, CATALOG, etc.) either gives error 31 again or Drive not ready (error 74).

Internally the 2031 has been compared to a 4040 sawn in half because only 2K of RAM is present. Also, and I have not seen this mentioned anywhere, the number of microprocessors have been halved ! The 4040 has a 6502 to take care of the IEEE processing and a 6504 for controlling the drives themselves. In the 2031 a 6502 runs everything. This probably accounts for the slightly slower speed of this device compared to the dual drives.

The memory map is quite uncomplicated. The RAM is from $0000 to $07FF with the BAM area at $0700-$07FF; the work area, stack and zero page at $0000-$02FF; and the rest (what's left of it) for the buffers. Two VIAs (6522) sit at $1800 and $1C00 and the ROMs (two 8K chips) occupy $C000 to $FFFF. Due to the address lines not being fully decoded the ROMs are also duplicated at $8000 to $BFFF; and $0000 to $1FFF also appears at $2000, $4000 and $6000.

As the memory layout has been totally changed around from the 4040 (Commodore running true to form again) it hardly needs saying that many of the utility programs that run on the other drives will need updating before they will work. Some I have managed to get working but others will require quite a bit of detective work before they are happy with DOS 2.6.

The utility program following shows which tracks and sectors have been used by each file (PRG, SEQ or REL) on a disk. It lists either to the screen or a printer (I hope, as I don't possess one as yet) and should run on all drives including DOS 1.

```
320
100 REM **  TRACKS & SECTORS USED ON DISK FILES (PRG, SEQ
     OR REL)
110 REM **  D J VINER  15.4.83
120 S$="                    ":REM    40 SPACES
130 PRINT"<clr>TRACKS & SECTORS USED"
140 INPUT"<3dn>PRINTER   N<3lft>";Z$
150 IFZ$="N"THEND=3:GOTO180
160 IFZ$="Y"THEND=4:GOTO180
170 GOTO130
180 PRINT"<dn>INSERT DISK INTO DRIVE 0 & PRESS SPACE
    <2dn>"
190 GETZ$:IFZ$<>" "THEN190
200 :
210 REM ** OPEN PRINT CHANNEL
220 OPEN1,D
230 :
240 REM ** OPEN DISK CHANNELS
250 OPEN15,8,15,"IO":Q$=CHR$(0):OPEN2,8,2,"#":
    T=18:S=1:C=3
260 :
270 REM ** GET A BLOCK FROM DISK
280 PRINT#15,"U1:2,";0;T;S:GOSUB790
290 :
300 REM ** GET NEXT TRACK & SECTOR OF DIRECTORY
310 PRINT#15,"B-P"2;0
320 GET#2,A$,B$:GOSUB790:T1=ASC(A$+Q$):S1=ASC(B$+Q$)
330 :
340 REM ** GET TRACK & SECTOR OF FILE
350 PRINT#15,"B-P"2;C
360 GET#2,A$,B$:GOSUB790:FT=ASC(A$+Q$):FS=ASC(B$+Q$)
370 :
380 REM ** CHECK FOR LAST FILE
390 IFFT=0THENCLOSE1:CLOSE15:CLOSE2:END
400 :
410 REM ** GET FILE NAME
420 AA$="":FORJ=0TO15:GET#2,A$:GOSUB790:AA$=AA$+A$:NEXT
430 :
440 REM ** GET BLOCKS USED BY FILE
450 PRINT#15,"B-P"2;(C+27)
460 GET#2,A$,B$:GOSUB790:B=ASC(A$+Q$)+(ASC(B$+Q$)*256):
    C=C+32
470 :
```

```
480 REM ** PRINT FILE NAME & BLOCKS USED
490 B$=STR$(B):AA$=AA$+"     "+B$+" BLOCK(S)":N=1:
    PRINT#1,AA$:PRINT#1
500 :
510 REM ** CHECK FOR LAST DIRECTORY ENTRY IN CURRENT BLOCK
520 IFC>255THENT=T1:S=S1:C=3
530 :
540 REM ** PRINT FILE TRACK & SECTOR
550 N$=RIGHT$(S$+STR$(N),4)
560 N$=LEFT$(N$+"  TRACK :"+STR$(FT)+S$,22)+"SECTOR :"+
    STR$(FS):N=N+1
570 PRINT#1,N$
580 :
590 REM ** GET NEXT FILE TRACK & SECTOR
600 PRINT#15,"U1:2,";0;FT;FS:GOSUB790
610 PRINT#15,"B-P"2;0
620 GET#2,A$,B$:GOSUB790:FT=ASC(A$+Q$):FS=ASC(B$+Q$)
630 :
640 REM ** CHECK FOR END OF FILE
650 IFFT>0THEN550
660 :
670 REM ** WAIT FOR KEYPRESS IF PRINTING TO SCREEN ONLY
680 IFD=4THEN280
690 PRINT"<2dn>PRESS SPACE OR X TO EXIT"
700 GETZ$:IFZ$=""THEN700
710 IFZ$="X"THENDCLOSE:END
720 IFZ$<>" "THEN700
730 PRINT"<up>                                    <up>"
740 :
750 REM ** GO AND GET NEXT FILE
760 GOTO280
770 :
780 REM ** DISK ERROR CHECK
790 INPUT#15,E,E$,E1,E2:IFE<20THENRETURN
800 PRINTE;E$;E1;E2:CLOSE15:CLOSE1:CLOSE2:END
810 :
820 REM ** SAVE PROGRAM
830 A$="USED T & S":SCRATCH(A$):DSAVE(A$):VERIFY(A$),8
```

--o0o--

PET SERIAL INTERFACE

By Barry West.

Though well-endowed with parallel ports, the PET has no built-in serial interfaces. Serial communication means sending one bit at a time over a single line; uses range from Telex-type communication, to driving certain printers and other peripherals.

A common application of serial data is in sending information over a telephone line, between a computer and a terminal (or another computer). For example, there are amateur-run 'bureaux' such as FORUM-80, using which, it is possible to download software and exchange messages with other users. The interface to the phone system is via a 'modem' (MOdulator/DEModulator) - these are available commercially, alternatively kits for 300 baud (V.21) modems are marketed by Messrs Maplin (0702-552911) and Ambit (0277-230909). The Maplin modem requires direct connection to the phone line, while the Ambit one is acoustically-coupled. Incidentally, both of these firms allow people to order components by telephone connection to their computers, using a terminal and modem.

As to the problem of making the PET communicate serially, again there are ready-made solutions on the market, such as the Yorkshire Microcomputers 'NETKIT', which is a hardware interface complete with driving software in ROM, providing two-way data transfer at several bit-rates under software control.

For the 'hackers' amongst us, the following notes may be of help. At low bit-rates, an entirely software solution may be considered; however, the processing needed would be hard to merge into whatever else the PET was doing in parallel. Therefore, I have chosen a hardware method, using a chip known as a UART or ACIA [Universal Asynchronous Receiver/Transmitter or Asynchronous Communications Interface Adapter - Ed], type-number SY-6551, which provides asynchronous (start/stop) serial input/output, and looks after framing, parity bits and like matters. The circuit shown requires connections to the memory connector

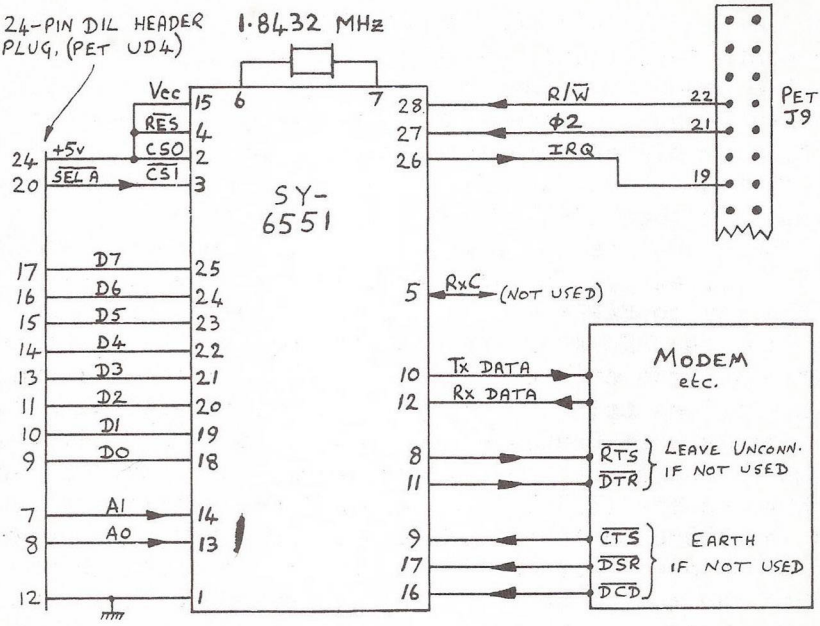
J9 and also a spare ROM socket (UD4 is assumed). The only components needed are the chip and a 1.8432 MHz crystal, plus the connectors and a piece of Veroboard, etc; the assembly is best mounted inside the PET, to keep the leads short. The serial data input and output and the five modem control lines are at TTL levels - more electronics is needed if a true RS-232C interface is wanted.

The chip is memory-mapped, to addresses $A000 to $A003. Actually, in the simple scheme shown it re-appears all the way to $AFFF - with better decoding, room could be made for a 2K ROM. By addressing the relevant memory locations, the chip can be made to talk (and listen) to serial devices using 5, 6, 7 or 8-bit codes, with odd or even parity, at rates between 50 and 19,200 bits/second ('baud'). A simple assembly-language program is shown, which turns the (3032) PET into a 'dumb' terminal, for use with a 300-baud modem. More ambitious use of the serial port would preferably be interrupt-driven, and should take care of the modem control signals provided by the chip.

Successful uses of the interface so far include communication with remote computers using a modem; reception of 'ham' 5-bit RTTY (teleprinter) signals; and passing PET programs to a BBC computer (boo !).

Offers to write some decent machine-code software to exploit the interface would be most welcome....

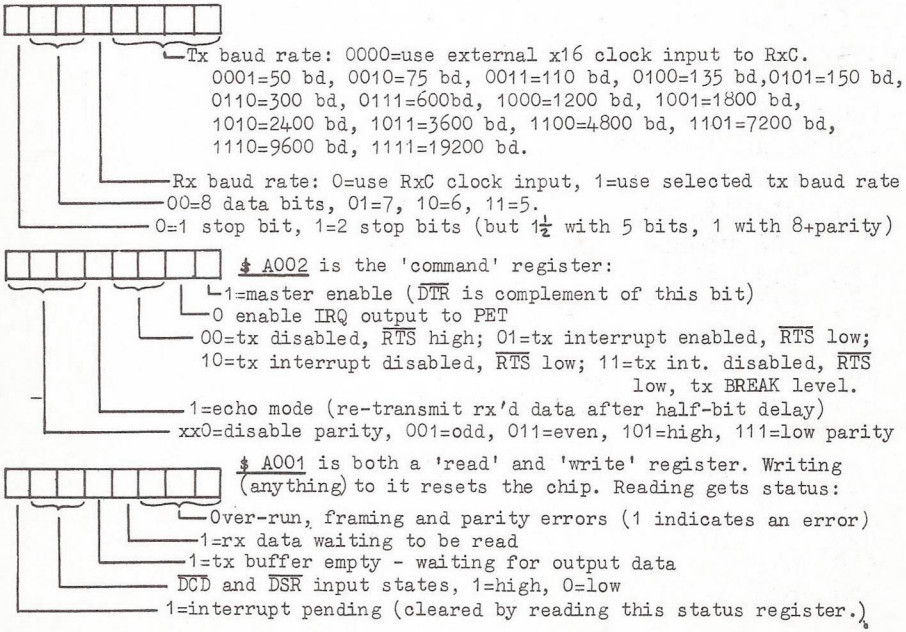
```
100 "*=$7000
130 "SCREEN=$E3D8
140 "KEYBD=$E285
150 "UART=$A000
210 "LDA #$0036  ;INIALISE UART
220 "STA UART+1  ; RESET CHIP
225 "STA UART+3  ; 300 BD
230 "LDA #$006B  ;IRQ OFF
240 "STA UART+2  ;EVEN PARITY
250 "LDA #112    ;LOWER TOP OF
260 "STA 53      ;    RAM
270 "LDA #14     ;UPPER/LOWER
280 "STA 59468   ;  CASE
300 "LOOP:LDA UART+1 ;STATUS REG
310 "AND #8      ;RX CHAR ?
320 "BEQ TX      ;...NO
```



The chip is memory-mapped to 4 locations at $A000 - A003.

$ A000 is the data being transferred to or from the modem etc.

$ A003 is a control register, used to set baud-rate etc. under software control:



```
330 "LDA UART+1  ;CHAR TO BE RX'D
340 "AND #3      ;FRAMING,PARITY
350 "BNE DUFF    ;  ERRORS?
360 "LDA UART    ;GET THE CHAR.
365 "AND #$007F
370 "TAX         ; CONVERT IN-
380 "AND #$0060  ; COMING ASCII
390 "CMP #$0060  ; CHAR TO PET
400 "BNE NOT11   ; 'ASCII' CODE
410 "TXA
420 "AND #$00DF
430 "BNE PETASC
440 "NOT11:CMP #$0040
450 "BNE NOT1X
460 "TXA
470 "ORA #$0080
480 "BNE PETASC
490 "NOT1X:TXA
500 "PETASC:BNE DISPLA
505 "DUFF:LDA UART ;CLEAR DUFF CHAR
510 "LDA #$00A6
520 "DISPLA:JSR SCREEN ;WRITE CHAR
530 "TX:LDA UART+1  ;STATUS REG.
540 "AND #16     ;UART READY FOR
550 "BEQ LOOP    ; TX CHAR ?
555 "SEI
560 "JSR KEYBD   ; 'GET' CHAR
570 "BEQ LOOP
580 "CMP #18     ;RVS KEY ?
590 "BEQ EXIT    ; YES-EXIT
600 "TAX
610 "AND #$00E0  ;PET 'ASCII'
620 "CMP #$0040  ; TO ASCII...
630 "BNE TRANS
640 "TXA
650 "ORA #$0020
660 "BNE ASC
670 "TRANS:TXA
680 "ASC:STA UART ;TX CHAR
690 "BNE LOOP
740 "EXIT:RTS
```

--o0o--

THE 64 COLUMN

By Mike Todd.

THE EDIT BUG - AND A FIX

In the last of my 64 columns (March Newsletter) I made brief mention of a screen editing bug of which most users of the 64 will now be aware.

If you're a new 64 owner and you've not come across the bug yet, try the following:

Reset the 64 (turning off and on again is quickest) and put the cursor at the bottom left of the screen and type a line number, followed by enough characters to make the cursor wrap-around to the next line. Then press RETURN to enter the line.

This is just a way of forcing lines 23 and 24 to become "linked" as wrap around lines.

Now, with the cursor back at the bottom left of the screen, try to move the cursor left using the DEL key. The result will be to make the 64 crash - you will get "L" on the end of a line, followed by "OAD" which the 64 doesn't recognise and gives a SYNTAX ERROR, then "RUN" (at which point any program within the 64 will actually start to RUN) and then READY.

This is all very well, but as soon as this happens, the keyboard jams up and the only key that appears to be operational is the CBM graphics key which, when pressed, rapidly alternates between graphics and text modes! In other words, the 64 has crashed.

However, if this happens to you, don't despair. Press the left hand shift key and "3".

The result should be a request to PRESS PLAY ON TAPE - but still no keys respond - not even the RUN/STOP key. Now,

assuming you've got a cassette deck attached, do as the instructions tell you and the screen should go blank as if the 64 was reading the tape.

Now you can press RUN/STOP and all will be well! But, this only works in "direct mode" and not if the crash is caused from within a program.

The bug is due to the current character colour being accidentally (and I've not discovered why it occurs) placed into one of the CIA registers at $DC0F and, if bit 1 is 1, this locks out bit 7 of the keyboard scan!

Therefore, the bug only occurs with RED, CYAN, BLUE, YELLOW, LIGHT RED, GREY1, LIGHT BLUE or GREY3 and not with BLACK, WHITE, PURPLE, GREEN, ORANGE, BROWN, GREY2 or LIGHT GREEN. In other words CTRL or CBM-3, -4, -7, -8 will hang the 64, CTRL or CBM-1, 2, 5, 6 are fine. Curiously, YELLOW has a slightly different effect - it produces the PRESS PLAY message without the need for pressing SHIFT-3.

So, if you want to avoid the problem altogether, select one of the okay colours before starting work on the 64.

THE CASSETTE BUG

I also mentioned the fact that there were problems reading cassettes recorded on a Vic-20, although PET recorded tapes read correctly.

The effect that I have noticed is, if you attempt to LOAD a Vic-20 program, the 64 will usually (but not always) find the program and turn the screen on with "FOUND....". As soon as it then starts to try to LOAD the program (after the 13 seconds wait, or earlier if you press the CBM graphics key) there is an OUT OF MEMORY error.

The exact reason for this error message is obscure, but it would appear that the 64 is not reading the data correctly from the tape.

It is possible that the memory pointers read by the 64 are corrupted and out of range, hence the message. But this doesn't explain why they are read wrong in the first place.

Although I've not really done a lot of work on this yet, the problem is almost certainly due to timing differences between the Vic-20, PET and 64.

The clock at the heart of the PET runs at exactly 1000000 cycles per second (1MHz), while the clock on the Vic-20 runs about 10% faster at 1.10 MHz. On the other hand, the 64 runs a little slower than the PET at about 0.98MHz.

These clock signals also determine the pulse timings on the tape and all Commodore computers are capable of coping with a reasonable timing discrepancy. Unfortunately, the difference between 1.10 MHz and 0.98 MHz is just on the edge of this tolerance, and results in read errors occuring.

At present there appears to be no solution to the problem, other than loading the programs into a PET and then re-recording them! At best, this is messy and at worst it is impossible for those without access to a PET.

I'm looking at the problem and hope to have a solution by the time I write the next 64 column.

THE RAM BENEATH THE ROM

In the March Newsletter I also explained (on page 140) how to transfer the ROM set into the RAM at the same address and then switch over to it so that you can use and modify it.

The POKEs to location 1 (the memory map register) are slightly incorrect (sorry!).

Switching in the BASIC RAM is correctly shown as POKE1,54, but switching the KERNAL RAM alone is not actually possible, while both BASIC and KERNAL can be switched together using POKE1,53. POKE1,55 will return it to normal.

The following table shows all combinations of memory mapping bits and the POKE values needed.

bit 76543210	POKE 1, x	A000-BFFF BASIC	D000-DFFF I/O	E000-FFFF KERNAL
00110000	48	RAM	RAM	RAM
00110001	49	RAM	CHARSET	RAM
00110010	50	RAM	CHARSET	KERNAL
00110011	51	BASIC	CHARSET	KERNAL
00110100	52	RAM	RAM	RAM
00110101	53	RAM	I/O	RAM
00110110	54	RAM	I/O	KERNAL
00110111	55	BASIC	I/O	KERNAL

The last configuration is how the 64 is set up to start with and the various options available are clearly seen - note that RAM is available in the I/O space, as well as the character generator.

A simple program to transfer the entire ROM set to RAM and switch over as follows (it'll take about 75 seconds):

```
10 FOR I = 40960 TO 49151: POKE I,PEEK(I): NEXT
20 FOR I = 57344 TO 65535: POKE I,PEEK(I): NEXT
```

This transfers the ROMs into RAM, a POKE to a ROM location automatically going into the RAM beneath. Then we do the switch over, but not without disabling the interrupt just in case one is called during the switch. Although essential if switching the I/O block around, I doubt that if it's really needed when switching the KERNAL ROM and I've had no problems switching the banks with the single POKE 1,53:

```
30 POKE 56333,127: POKE 1,53: POKE 56333,129
```

You can prove that the BASIC is now in RAM by:

```
POKE 41229,80
```

which will change the spelling of the command word LIST - I leave it to you to work out what it's become!

To return to normal, just press RUN/STOP+RESTORE which will reset the values in the control register.

THE 64 CHARACTER SET

In the last Newsletter I published four charts of the Vic/64 character sets. Unfortunately, they were printed in the wrong order; not that this makes much difference, but it does make them a little illogical.

They were intended to appear in pairs - the CHR$ GRAPHICS/TEXT opposite each other and the POKE value GRAPHICS/TEXT similarly.

Anyway, I didn't mention the fact that the 64 characters are not formed the same as those on the Vic-20. This is because the single dots on the 64 are just a bit too narrow for the characters to appear clearly (for those who understand, I'll just mention the words "video bandwidth") and colours would not be very clear. Therefore they have doubled each horizontal dot (and in some cases the vertical dots too) to give increased width to characters.

I don't intend to redo the tables unless I'm inundated with a couple of requests so to do.

VIDEO OUTPUT

Those wanting to use the 64 in a business environment may wish to forget colour and just have a black and white output, especially if the signal is fed to a monochrome monitor.

Putting aside the arguments that colour can be an invaluable aid even in business programs, provided it is carefully used, the improvements in definition are reported to be quite significant.

There is a pure luminance signal which appears on the DIN socket on the back of the 64 and this can be fed straight into a video monitor.

At the recent PET show (sorry - Commodore International Computer Show) several 64 users asked if it was possible to obtain an "RGB" output from the 64, the reason being that the encoding of the colour signal will always degrade the quality owing to the processing that the signal must undergo. There's nothing that can be done about this, it is an inherent defect of the way colour television works.

However, some computers (such as the BBC micro) have separate red, green and blue (RGB) outputs provided at a point before the encoding takes place which provides a very significant improvement in colour quality.

This is not available on the 64 (nor the Vic for that matter) as the signal comes out of the video chip already encoded. It would be possible to decode back into RGB, but that would be a pointless exercise as a colour monitor or TV would be doing that anyway.

The additional processing involved in turning the video signal into a UHF signal suitable for a normal TV set also degrades the quality slightly, and a direct-fed colour monitor will often show an improved picture.

If you've got a TV with a video input you may find this gives an improvement, and Commodore have recently introduced their own colour monitor (the 1701) which also offers a significant improvement in colour quality, although it does cost about 230 pounds (inc VAT). Mind you, this is not a bad price, considering. Maybe Commodore will add a tuner to their monitor in the future and turn it into a domestic TV!

SIMONS BASIC

I had hoped to be able to tell you all about Simons BASIC for the 64 this time round. I've actually had a

preliminary copy for some time now and must say that I have extremely mixed feelings.

These range from total disgust at the way some of the huge number of commands have been implemented to sheer ecstasy at the flexibilty some of the commands provide.

I have to say that the version I have is quite old, and on disk - the release version will be a cartridge and is currently being prepared for release. Until I see this final version, I think it wise not to say too much, as many of the problems may have been put right.

But, just to whet your appetite, here is a list of the new BASIC commands and so on available to the user:

Programming Aids KEY, DISPLAY, AUTO, RENUMBER, PAUSE, LIN, CGOTO, RESET, MERGE, PAGE, OPTION, DELAY, FIND, TRACE, RETRACE, DUMP, COLD, DISAPA, SECURE, OLD

Input & Text manipulation INSERT, INST, PLACE, DUP, CENTRE, USE (PRINT), PRINT AT, FETCH, INKEY, ON KEY, DISABLE, RESUME

Extra numeric functions MOD, DIV, FRAC, EXOR, % (binary number), $ (hex number)

Disk Commands DISK, DIR

Structured Programming IF..THEN..ELSE.., REPEAT..UNTIL.., RCOMP, LOOP..EXIT IF..END LOOP, PROC, END PROC, CALL, EXEC, LOCAL, GLOBAL, ON ERROR GOTO, ERRN, ERRL, NO ERROR, OUT

Sound Commands VOL, ENVELOPE, MUSIC, PLAY

Joysticks and so on PENX, PENY, POT, JOY

Graphics Commands HIRES, REC, MULTI, LOW COL, HI COL, PLOT, TEST, LINE, CIRCLE, ARC, ANGL, PAINT, BLOCK, DRAW, ROT, CSET, CHAR, TEXT

Screen control FLASH, BFLASH, FCHR, FCOL, FILL, MOVE, INV, LEFT, RIGHT, UP, DOWN (scrolling), SCRSV, SCRLD, COPY, HRDCPY

Sprite & character Commands DESIGN, CMOB, MOB SET, MMOB, RLOCMOB, MOB OFF, MEM, DETECT, CHECK

The commands aren't always what they seem, and some have very odd constructions, often different to "industry standard" commands and often rather illogical. One command (RENUMBER) doesn't seem to work correctly - it renumbers the lines okay, but doesn't bother renumbering GOTOs, GOSUBs and so on! Perhaps they have changed on the released version - I'll let you know as soon as I find out.

64 APPLICATIONS

Now that the 64 has been on the road for a few months, the amount of software available is increasing at an alarming rate - there are assemblers, disassemblers, word-processors, business packages, scientific applications, amateur radio packages (which include RTTY, CW and slow scan TV!!), IEEE cartridges, and speech synthesisers.

At the show, Commodore showed two interesting software packages - the first is a talking picture book for children, with some super graphics.

The second is a football game, something like those for dedicated video games machines, only better. The graphics and sound have to be seen to be believed. It's played with "real" looking players and a "real" looking ball (complete with a shadow on the ground!). At half time, the players line up and leave the pitch!

The Z80 card is also expected to be available very soon, and I'm told on very good authority that it works! More of this at a later date.

--oOo--

700 SERIES NOTES

This review of the Commodore 700 is not written in the normal vein of this magazine in that it only refers to the changes in BASIC and normal operations and nothing about the internal workings of the machine. The reason for this is that I am a BASIC programmer in my spare time and I only dabble in machine code and in the hardware features of the PET that I am used to. In January this year I was lent a Commodore 700 without the built-in screen by Chromasonic Business Systems, Junction Road, LONDON, N19. The loan of this machine was a two way deal in that I would write them some demo software and at the same time I would try the machine out to see if I wanted to buy one. The machine I was lent had a serial number of 000025 so the following comments relate to this very early machine.

I did take the top off the machine to look inside and found that the ROMs were in fact EPROMs and I have since learnt that these have already been replaced by two upgrade sets. As usual with Commodore and a new machine there was no manual or documentation of any sort with it. All that was available was the initial publicity material. Just before I had to return the machine I was lent a very preliminary rough photostat technical manual by Supersoft. This was of some use in finding out about the machine but I understand that there is still no manual for this machine at the time of writing (April). A point to note is that the character generator is different on the two different model 700 machines. I will cover the changes to BASIC, the new BASIC commands, keyboard layout, and new escape commands and finally any other improvements not covered above.

DIRECTORY has two new functions, it displays 24 file names then asks 'more?' pressing any key displays the next 24 file names.

The second function is a failing in my opinion if the drive door is open Directory will not initialise the disk but give an error. CATALOG however acts the same as Directory except it will initialise the drive.

INPUT no longer 'bombs out' with a carriage return. INPUT and INPUT# allow the input of up to 160 characters before generating a 'string too long' error. BASIC lines are now allowed to contain 160 characters. The pound sign does not print as such to the 4022P printer and I didn't have an Epson to try it on.

TI is no longer a reserved variable. TI$ is now 7 characters long: hours, minutes, seconds and tenths of seconds.

Some of the error messages have improved descriptions of the errors. The final thing that I found which is a failing in my opinion is that shift and run/stop only loads but does not run the first program on drive 0.

The new BASIC commands that I found are as follows:
BLOAD and BSAVE: These commands load or save a particular 64K block of memory.
BANK: This allows you to access a particular bank of 64K memory.
PRINT USING and PRINT# USING allow printing to the screen or other device in a predetermined format. e.g. 123.44 or 1.30.
PUDEF (print using define): This changes the '.' in the print using statement to a character of your choice. e.g. 123.44 to 123,44 or 123/44.
KEY has two functions: on its own it displays the strings assigned to the individual function keys e.g. 'key 1, Print'. The other way of using this is to assign a particular string to a particular function key. e.g. key11,"run" will display 'run' on the screen when function key 11 is pressed. More on the function keys later.
RESTORE [line no.] performs a restore on all data statements after [line no.].
DELETE [line no.] deletes any lines in the [line no.] statement the same way as LIST functions now.
DISPOSE, TRAP and RESUME are all used to trap errors in your program and then to continue after displaying a suitable error message. These commands are complicated to use and I didn't get them to work as was intended.

INSTRING this finds the occurrence of one string within another. Again I couldn't get this function to work properly.
IF-THEN-ELSE works as it sounds e.g. 100 IF A=124 THEN PRINT "A=124":ELSE PRINT"A<>124". This of course is very powerful when combined with the 160 character lines.

There are several more reserved variables all to be used in conjunction with the DISPOSE, TRAP and RESUME commands. The keyboard has a totally different feel to it than the PET keyboard and to this date I am not sure which I prefer. The 700 has a softer quieter feel and when switched off it makes some funny creaks and groans. The cursor controls are now on four separate keys which I found to be annoying. One very useful key was the NORMAL/GRAPHICS key this is easier than remembering the equivalent of POKE 59468,12. The graphics characters on the top row of keys are obtained by using the CONTROL key instead of the SHIFT key. On the numeric key pad there are three new keys ENTER which duplicates the RETURN key, CE which clears the last numeric entry and 00 which is useful for entering financial data and also line numbers.

I mentioned the function keys earlier. On power up they are assigned as follows:
1: print, 2: list, 3: load, 4: save, 5: open, 6: close, 7: copy, 8: directory, 9: scratch, 10: chr$(.

The keys 11 to 20 are unassigned and the keys 1 to 10 can be reassigned by the user. Keys 11 to 20 are SHIFTED 1 to 10. These are not the ideal choice of statements assigned to these keys in my opinion but as I said they are easily changed.

The machine code monitor may be accessed by SYS8 * instead of SYS 4 and return to BASIC is by g ff87 instead of x. - It is a much improved monitor over the original TIM monitor. [* Not the recommended entry point - Ed.]

Some of the usual PET shorthand has been changed e.g. 'dL' is replaced by 'dl0' for dload.

There are a whole range of ESCAPE functions available. These are too numerous to cover here but here are some examples escape and 'R' reverses the screen 'N' returns the screen to normal. 'U' changes the cursor from a flashing block to the underline character. Virtually every letter has a function with the ESCAPE key. The ESCAPE key toggles so it does not need to be held pressed with the other key.

There is a SID chip on the 700 just like the 64 only the addresses are different. I had only a limited amount of success with this chip even though I had all the necessary addresses it really needs an external amplifier rather than the in-built loudspeaker. There is one annoying thing about this chip, or maybe it's the operating system, that is if the bell is sounded, e.g. at the end of a line, it continues to ring faintly long after it should, it in fact continues until the machine is reset.

There is a soft reset switch provided at the rear of the machine. This is not before time and will restore from most crashes leaving the program intact. The vast amount of RAM available is more usable than at first appears. There is 64K for program space, a further 64K for storing string variables and another 64K for numeric variables. Try dimensioning and filling a 4000 string array on the PET.

This is just a brief review of the 700 and I hope it will be of some interest. I am not sure even now about my feelings about the machine. I will however be buying one later this year when hopefully Commodore will have got the ROMs sorted out and there will be some documentation on the machine.

By Colin Spencer.

--o0o--

ODD BIT

Fujitsu, who hitherto have concentrated on mainframe computers, have introduced a C-64 look-alike.

--o0o--

CBM-710 NOTES

From John Marchant.

Here are some useful addresses on the CBM-710:

Location
- 200 Cursor address low byte
- 201 Cursor address high byte
- 202 Cursor line
- 203 Cursor column (0-79)
- 205 Last character entered (as a number in character table - not ASCII)
- 209 Keyboard buffer character count
- 220 Window top
- 221 Window bottom
- 222 Window left margin
- 223 Window right margin

Other points to note are i) DCLOSE also closes printer if open; ii) OPEN to the printer while a disk file is open - oh, confusion! Some disk commands even go to the printer. There are probably several variations on ROM sets about and the final release version may be different.

--oOo--

PET EPROM PAGERS

The Jaytee Electronic Pager is a EPROM pager that enables you to choose any one from eight EPROMs in one location. This can either be by a DIL switch or by software control. Pagers can be used in any ROM location in the PET, even in the Character Generator socket. If two pagers are used it is possible under software control to switch both to any combination of either board. A good use of this is in PRESTEL where the Character Generator and the UART board have to be switched together. Other ways that the pager can be used are to run 32K byte of machine code program from one of the spare ROM sockets.

There are four configurations in which the pager can be purchased:

8-Way Pager DIL switch controlled. Part No. JTP1/N
8-Way Pager software controlled
(without USER Port Connector) Part No. JTP1/S
8-Way Pager software controlled
(with USER Port Connector) Part No. JTP1/U
2-off 8-Way Pager software controlled. Part No. JTP1/P
(Both connected to one USER Port Connector).

All the above pagers come complete with foam feet, DIP header cable and full instructions. If you require to use the pager for the character generator ROM please specify, as the DIP-to-DIP cable for this socket must be very short to avoid timing problems (The character generator ROM runs VERY fast).

PRICES (for ICPUG MEMBERS)

	£		£
PART No. JTP1/N -	21.00	PART No. JTP1/S -	21.00
PART No. JPT1/U -	25.50	PART No. JPT1/P -	46.50

Note:- The above prices are LESS than 50% of any others advertised and are fully inclusive of postage. ALL BOARDS ARE FULLY GUARANTEED.

Other Goodies:

USER/IEEE Port Connectors complete with covers -	4.50
24inch 24-way DIP to DIP Header Cable -	3.45
12inch 24-way DIP to DIP Header Cable -	3.15
6 inch 24-way DIP to DIP Header Cable -	2.95
BASIC 2/BASIC 4 ROM set pager -	30.00

NO VAT to be added - postage on orders under 10.00 is 80p
over 10.00 nil

John Tiley - JAYTEE Electronic Services, 4, Halford Close, Broomfield, Herne Bay, Kent. CT6 7UN. Tel: Herne Bay (02273) 5254.

--o0o--

COMMODORE COLUMN

Vic vs Victor.

Victor Technologies has filed a trademark infringement action in the US against Commodore but has received a counter claim for $20 million in damages. Victor filed for infringement and unfair competition last December to protect its Victor trademark, claiming against Commodore's use of the name Vic, Vic-20 and Vic-1515. On January 3rd Commodore denied the allegations and sought an injunction against Victor's use of the Victor name and claimed $200 million punitive damages. Since January the two companies have exchanged written questions and replies.

Commodore's Legal advisors state that the initial defence is that there is no conflict in the market place. If it is decided there is a conflict Commodore can show that they have been selling the Video Interface Chip (VIC) since 1977 and that a million Vic-20s have been sold since mid-1981.

CORBY for Commodore.

Commodore is to set up a £ 22 million personal Computer factory in CORBY, and is also looking for a British site for its major European Research and Development centre. The company is investing £ 20m in the site with £ 2m in government aid. It will employ 300 people by the end of 1984, when it will be producing 50,000-60,000 Vic and C64 computers per month. Production should start imminently, transferring current production from West Germany. The german plant will concentrate on the 8000- & 700-series business machines. Of the 700,000 machines expected to be manufactured, 300,000 will be sold in the UK.

Lisa Competitor.

Jack Tramiel announced a competitor for LISA at a recent meeting. The competitor would be offered at a price of $4,000 - less than half the price of LISA - it will incorporate a 'mouse' and will use third party software houses to provide user-friendly software.

VICSOFT.

Commodore are experimenting with the provision of VICSOFT over the counter, instead of being via mail order. Hamleys, the London Toy store, is stocking the VICSOFT goods. The possibility of expanding VICSOFT further by allowing a major retail chain to take on VICSOFT is being considered.

NEW PRODUCTS.

The new portable from Commodore, dubbed the Executive 64, or SX-64, is said to be fully C-64 compatible and incorporate one or two disk drives, 5" screen, and weigh less than 20lbs. This new unit is reputed to be shipped in volume by the autumn, with anticipated sales of 200,000 forecast for the first six months.

New printers are anticipated to include colour, daisy-wheel and high-speed models. Those of you that attended the Commodore Show will have already seen the new 4-colour plotter with a 'Sinclair' price-tag.

The price war in the US is hitting some of the video game companies quite hard. While Commodore are making a handsome profit, Atari's fortunes are not so good. Tired of having their games appearing on so many other machines they have adopted the policy of 'if you can't beat them, join them'. They are now to market games for a range of computers, including Commodore's.

The BASIC compiler from Oxford Computer Systems is to be released in a version for the C-64. The product code is PSC-6440 and the price £ 50. A version for the 700-series is in preparation.

The long awaited 700-series have been delayed while bugs in the software have been removed. The 710 & 715 are due in the UK in mid-July. The 730BX (with 8088 processor and 256K RAM) is due about mid-August only with CP/M-86 (no MS-DOS). So you think your 700 will now be bug-free? Here's a good one that's slipped through. The screen editor has a bug that only appears when column 28 (or is it 29?) is in

rvs, then in the line above, the leftmost column of pixels in the leftmost column lights up. Answers on a postcard to....

Writing this column, one becomes a 'Commodore watcher'. Now in the US, they are advertising for staff with 'knowledge of assembler, C and Pascal' and 'CP/M, MS-DOS, UNIX and other operating systems. Knowledge of graphics and graphics standards'. Also 'familiar with video, magnetic recording techniques, switching and linear power supplies' and 'familiarity with 8-bit and 16-bit microprocessors'. Add this to the fact that the Zilog Z8001 will appear in Commodore's new 16-bit computer (Zilog advert) and one can see the future trends.

Will they, won't they? The 500 that is. Plans to drop the 500 because Commodore could not identify a market for it and wanted to concentrate on the C-64 have met with strong dealer opposition.

The Vic-20 is currently the most popular home computer in the world with over 1.4 million sales to date.

Finally, the Commodore Show was the best attended yet, with over 16,500 visitors - more than double last year's figure. If you saw the 'soccer' game for the 64, weren't you impressed by the graphic animation? - just like watching TV!

R.D.G./T.C.

--oOo--

COMPUTED GOTO/GOSUB & RESTORE(n)

By David Viner.

Over the years there have been quite a few routines published in various magazines that add a computed GOTO or GOSUB to the PET. These all suffer from the fact that the user must remember to load the things in before using them in programs. Also, some of them are only called by a SYS command and not by using the GOTO/GOSUB keyword. An alternative solution is to burn the routines into EPROM and then turn them on at the beginning of a program session,

but this again suffers from the user having to remember a new SYS call and of the possibility of the PET having already used up all its spare ROM sockets. The solution that follows is for BASIC4 users (with EPROM programmers) only and relies on the fact that the BASIC4 ROMs have spare 'gaps' in various places that are not used at all.

The change required to implement the computed GOTO/GOSUB means only altering eight bytes in the 'B' ROM. When this has been done, the PET will still operate as normal but code such as GOTO A or GOSUB X*5 is now possible. Please note that the THEN statement is not affected and a statement such as IF A = 6 THEN K is illegal. This can, of course, be implemented by THEN GOTO K. If the number computed does not occur as a line number then the PET will stop with an error message.

Another function never (until the advent of the 500- and 700-series) implemented was the Restore to line number. Again several routines have appeared over the years but these, too, can be added to the 'B' ROM with only the addition and changing of about 50 bytes. The normal RESTORE is not affected and works as normal by resetting the DATA pointer to the start of the program. RESTORE(n) then works by resetting to line number 'n' where 'n' is either a straight number or computed (i.e. RESTORE(A*F/5) is legal).

The main routine is added in the gap at $BF41 and four other bytes are changed earlier on in the ROM. The byte at $B018 is changed from $B6 to $40, $B019 becomes $BF instead of $B7. This changes the RESTORE pointer so that it now points at the new routine. The bytes at $B831/2 are changed to $69 and $BF which re-routes a subroutine jump to point at a new routine at $BF69. (Don't get these the wrong way round! The $69 goes in $B831 and the $BF goes in $B832).

```
The main routine is as follows:
BF41 JSR $0076  Get chr after RESTORE
BF44 CMP #$28   Is it a bracket '('
BF46 BEQ $BF4B  Yes; so branch
BF48 JMP $B7B7  Else do normal RESTORE
```

```
BF4B JSR $0070  Get next character
BF4E JSR $BF69  Evaluate expression
BF51 JSR $BEEF  Check for ')'
BF54 JSR $B5A3  Search for line no.
BF57 BCS $BF5C  Branch if found
BF59 JMP $B86E  Else print ERROR and
                exit to BASIC.
BF5C LDY $5D    Get new position into
BF5E LDX $5C     Y and X
BF60 BNE $BF63  Position = position - 1
BF62 DEY
BF63 DEX
BF64 STX $3E    Store new position in
BF66 STY $3F    DATA pointer
BF68 RTS        End of subroutine
BF69 JSR $BD84  Evaluate expression
BF6C JMP $C92D  Convert resultant
                number to integer.
```

A hex dump of the routine is here for those without an assembler.

```
BF41 20 76 00 C9 28 F0 03 4C
BF49 B7 B7 20 70 00 20 69 BF
BF51 20 EF BE 20 A3 B5 B0 03
BF59 4C 6E B8 A4 5D A6 5C D0
BF61 01 88 CA 86 3E 84 3F 60
BF69 20 84 BD 4C 2D C9
```

The routine could be added to BASIC2 but there is not sufficient room in any of the ROMs to hold it and a separate EPROM in one of the spare sockets would be needed. Of course most of the subroutine calls would need to be changed to the BASIC2 equivalents before the routine could run.

Please note that the routine works from power up and does not have be turned on or off by any SYS commands.

--oOo--

DISK FILE - SECTOR 7

By Mike Todd.

1540/4040 COMPATIBILITY

Some time ago, I was sent a 1540 disk, and wanted to copy some of the material from it using a 4040 dual drive unit, but persistent read errors on the disk caused me to abandon the operation and start taking the disk drive apart looking for a fault, but none was found.

When the 1540 (and later the 1541) disk drives appeared, there were rumours of some incompatibility between disks written on one being used on the other drive, despite Commodore's assertions that the disks were compatible.

At the Commodore show, several people were talking about this compatibility problem saying that it was impossible to back up a 1540-written disk on a 4040 drive owing to persistent read errors.

Well, I've looked in great detail at the problem and have made some interesting discoveries. I have to stress that the investigations are not complete and so I must advise caution in transferring disks from one system to the other.

It would appear that any disk which has been written to on a 1540 (whether it was formatted on a 1540 or a 4040) should not be used in a 4040 if you intend to write to it. If you do, there is the risk that a read error will occur either at the time of writing or on a later occasion.

I can find no evidence of any problems writing to a 4040 formatted disk on a 1540 - provided, as above, that you don't then attempt to write to it on a 4040.

Fortunately, there are a couple of simple fixes for the 4040 drives that seem to work. The first is for those using a 4040 (using DOS2.1 or 2.2) to write to a disk which has already been written on by a 1540 or 1541.

```
PRINT#15,"M-W";CHR$(217);CHR$(16);CHR$(1);CHR$(8)
```

where OPEN15,8,15 has been used to set up the command channel to the drive.

This makes the 4040 write as if it were a 1540 and therefore the resulting disks ought to be treated as if they had been written to by a 1540.

The second is for those who've already got a disk which is producing the read errors (which should be error 22 - DATA BLOCK NOT PRESENT). It can't guarantee a successful read, but should improve the probability significantly.

```
PRINT#15,"M-W";CHR$(197);CHR$(52);CHR$(1);CHR$(31)
```

If a disk read operation fails, it is normally attempted automatically up to about ten times and I found that many reads would fail on the first couple of attempts, but would produce a succesful read after, say, 5 or 6. The read error is only generated if the read fails after the 10th attempt.

The above line forces the drive to make up to 31 attempts (the maximum that it is capable of) thereby giving the drive a better chance of a successful read.

Of course, extra read attempts take time and so you may have to wait very much longer for the operation to work. It is possible that it is this need to make many attempts at a read operation that has led to the reputation of 1540 disks being very slow on 4040 drives.

If you're happy to know this, then skip the rest of this section - if you'd like to know the reasons why, I'll attempt a description.

When a disk is formatted, in whatever drive, each track is first erased and then each sector is written using "null" data.

Each sector consists of a sector header giving details of the disk ID, the track and of course the sector number.

This is followed by a short gap and the data block itself.

If you've read my earlier DISK FILEs, you should know that the disk encodes each 8-bit byte as 10 bits when writing to the disk, and that there are never more than 2 consecutive zeroes recorded. Also, ten or more consecutive ones are used as synchronising pulses to indicate the start of a header or data block - the end of the SYNC bits is indicated by the first zero which follows.

The header block is preceded by 40 1's used as a SYNC pulse, then 6 "bytes" (60 bits) of the header - the first being an identifier to tell the drive that it is reading a header block and not a data block.

This process is absolutely identical on the 1540 and 4040 drives.

There then follows a gap (known as the "header gap"), another sequence of 1's as the SYNC for the data block, then the data block itself followed by yet another gap (known as the "tail gap") before the start of the next sector.

On the 4040, the header gap is about 90 bits long, followed by 40 SYNC bits. On the 1540, there are only about 80 bits of gap and 48 SYNC bits.

The header blocks are only ever written during the formatting operation, but the data block (including the SYNC bits) are written as required.

Now, imagine a 4040 about to write a block which has already been written on by a 1540 - it waits until it has found the correct header, waits the time taken for about 90 bits to be read and then writes its 40 SYNC bits followed by the data itself.

Unfortunately, the last 10 bits of the gap already contain the start of the SYNC bits which the 1540 had written!

When the 4040 reads this back, it will see these redundant SYNC bits and treat them as true SYNC bits, waiting for the first zero as a signal to start reading the block of data. However, the start of the proper (4040) SYNC bits will usually be a little "ragged" and contain a zero or two!

The result is that the 4040 will see these as the start of the data and procede to read the start of the SYNC bits as data!

The first character that it expects to see is the data block identifier which it doesn't see and so assumes that the data block doesn't exist.

The "raggedness" of the start of the SYNC pulses is often variable, and occasionally will not result in a zero being found and so the end of the sequence of SYNC bits is identified correctly. The second fix, as already mentioned, allows sufficient attempts at reading to give the electronics a fair chance of successfully identifying the end of the SYNC pulses.

The first fix forces the 4040 to generate a shorter gap (80 bits instead of 90) which will completely over-write the original (1540) SYNC pulses, but this then means that the new block has become 4040-write-incompatible.

Common sense says that the 1540 drive should also have problems reading such corrupted SYNC pulses, but I have no immediate evidence that this is so - but it must happen, unless the electronics are more sophisticated and read the SYNC pulses without the "raggedness".

Of course, using a 1540 to write to a disk formatted on a 4040 should not cause the problem.

Some may be interested to know that, in order to investigate this problem, I have devised a method of reading a complete track into the PET, allowing me to examine a it bit by bit, exactly as it was recorded.

The technique is fairly simple, but requires reasonable electronic knowledge to build the connector which connects the innards of the disk drive to the PET and I'll happily give anyone interested an outline of the technique if they want to experiment.

READING THE DIRECTORY

I promised that I'd give a couple of programs to read the directory from a disk. Unfortunately the time that I've spent investigating the 4040/1540 problems has meant there's only time and space for one program.

The program I've included is one which will read the complete directory from a disk in its "true" form, rather than the "program" format normally read when using LOAD"$",8.

The program consists of two main subroutines. The first is at 20000 and is used to initialise the directory reading operation. Line 20020 initialises the disk (reading the directory as a sequential file doesn't "auto-initialise") and line 20030 opens a channel from the directory.

There are two "utility" subroutines at 21000 and 22000 which are used to read two or just one byte from the disk and return the ASCII values (in A or A and B).

20040 reads the very first character which identifies the format of the disk. "A" or CHR$(1) indicates that it is in the format used by the low density disks (4040, 1540 etc), whilst "C" indicates the 8050 format. From this information the number of BAM blocks to be skipped is set in NB.

Then, lines 20090-20110 actually skip the BAM blocks - this is the section of the directory which holds the disk

```
10000 REM ***** READ ONE ENTRY
10001 IF NOT(OK) THEN RETURN
10002 IFNC<>0 THEN GOSUB21000

10010 GOSUB 22000
10020 TY=A

10100 GOSUB 21000
10110 TR=B:SC=A

10200 N$=""
10210 FORI=1TO16:GOSUB22000
10220 IFA<>160 THEN N$=N$+A$
10230 NEXTI

10300 GOSUB 21000
10310 TS=B:SS=A

10400 GOSUB 22000
10410 RS=A

10500 K=4:GOSUB 20100

10600 GOSUB 21000
10610 RT=B:RS=A

10700 GOSUB 21000
10710 NB=A*256+B

10800 IF NOT (OK) THEN CLOSE2:CLOSE1

10900 NC=NC+1:IFNC=8THENNC=0
10910 RETURN


20000 REM ***** INITIALISE
20010 OK=-1:NC=0
20020 OPEN1,8,15: PRINT#1,"I0"
20030 OPEN2,8,2,"$0"

20040 GOSUB 22000
20050 IFA$="A" OR A$=CHR$(1) THEN NB=1:GOTO20080
20060 IFA$="C" THEN NB=3:GOTO20080
20070 PRINT"UNRECOGNISED FORMAT":CLOSE2:CLOSE1:END

20080 FM$=A$:IFA$=CHR$(1)THENFM$="1"
20090 K=(NB*254)-1

20100 REM **** SKIP K BYTES
20110 FOR I =1TOK:GET#2,A$:NEXT
20120 RETURN

21000 REM ***** GET TWO
21010 GET#2,B$:B=ASC(B$+CHR$(0))
21020 IF ST<>0 THEN OK=0

22000 REM ***** GET ONE
22010 GET#2,A$:A=ASC(A$+CHR$(0))
22020 IF ST<>0 THEN OK=0
22030 RETURN
```

name, its ID and the "map" containing data on which blocks on the disk are free, and which have been used.

That completes the setting up. I've deliberately not included the ability to read the disk name or ID - I'll leave that up to you to experiment with.

From now on, every time you want to read an entry from the directory, call the subroutine at 10000. Line 10001 will RETURN if the end of the directory has been reached.

Note the inclusion of line 10002 which skips the first two bytes of each file entry, except for the first in each block of 8. This is because the first two bytes in each block contain the track and sector number of the next directory block and are not returned as part of the read process.

Lines 10010-10020 fetch the file type. 128=DEL, 129=SEQ, 130=PRG, 131=USR, 132=REL and 0=file deleted. 1, 2, 3 or 4 would indicate that the file had not been closed properly.

10100/10110 fetches the track and sector of the first block in the file (into TR and SC), and lines 10020-10230 read the file name into N$. The last few file names in the directory may end up as "null" file names (type=0) where space has been reserved for additional files in the last block in the current directory.

10300-10310 gets the start of the first side sector of relative files and 10400-10410 sets the record length in RS. 10500 skips 4 bytes (not used) and 10600-10610 fetch the track and sector if the file was "@" replaced and lines 10700-10710 set NB to the number of blocks used by the file.

The variable OK is used to flag when the directory is finished. It is set to false during the byte read subroutines if the end of the file is reached, and then checked at the end of the main subroutine. If it is false, line 10800 closes both of the files. It is also checked at the start of the routine to prevent it trying to GET from a closed file.

OK should be checked by the calling program, and the subroutine at 10000 should not be called if it is false. Perhaps using something like IF OK THEN GOSUB 1000.

The program's not exactly sophisticated, and is rather slow, but it does provide an insight into the way the directory is built up. You will even get details of files that have been scratched, but their entry has not yet been over-written in the directory (although the blocks on the disk will probably have been used).

SCRATCHING A FILE? THINK TWICE!

If you've ended up with an unclosed write file on the disk (which is indicated by an asterisk before the file type in the directory listing) then the way is clear for all sorts of unexpected disasters to occur - not just now, but possibly some time in the future!

The reasons are complex and I'm not going to bother to explain them, other than to say that the problems will occur if you try to scratch the file as it could result in files becoming "interlinked".

The only really reliable way to eliminate these unclosed files is to perform a VALIDATE operation (or COLLECT in BASIC4) which will get rid of the file, and tidy up any potential problems on the disk.

In fact, validating the disk is a wise precaution to be taken periodically anyway. If, as a result of a validation operation, the number of blocks free changes, then there was the potential for disaster to strike some time in the future.

My own rule-of-thumb is to do a total of all the blocks used by files and the blocks free, and if they don't total the original capacity of the disk, then I invoke the validation.

All this is fine if the file you were writing was

aborted deliberately and the data was not required. But, a write file can abort for several reasons and, if not closed immediately, certain operations (such as reading the directory, initialising the disk and so on) will result in the channel being closed with no hope for closing the file.

If this happens, the data is lost. Or is it? It's still on the disk (unless you've attempted a subsequent write operation on the disk, in which case the data probably won't be there) and Commodore have kindly provided a command to get it back!

This is the file mode "M" which I mentioned in a disk file a long time ago, and couldn't find a use for.

The obvious way of reading the data back is to open the unclosed file to read - but you'll get a WRITE FILE OPEN error. However, the "M" mode allows a read to an open write file, and you can happily read the data back and save it somewhere. But DON'T, whatever you do, write it straight back to the same disk. Instead, write it to tape, the other drive or hold it in memory until the disk has been validated. The format is:

```
OPEN 2,8,2,"0:FILE,S,M"
```

and read the data back in the normal way. But watch out, the ST variable will not be set to indicate the end of file. So you will have to know when the end of file has been reached (you will start getting garbage from the same, or even other files, if you continue to read beyond the end). Also, you must remember that the last block of data would never have been written to the disk and so the data will be incomplete - at least it's a way of retrieving some data.

Well, that's all for this time. The time taken to investigate the 4040/1540 problems has meant that a lot of the stuff I was going to include has had to be put off until a later date - that includes the fuller review of DISK REVEALED. Hopefully, it'll be here next time.

--oOo--

```
**********************************************

10  STILL WAITING TO AFFORD YOUR COMMODORE 700,
    500 OR EVEN C-64?

20  IF A VISIT TO YOUR BANK MANAGER IS
    LIKE A RETURN WITHOUT GOSUB ERROR
    THEN STOP.

30  BREAK OUT OF THE CONTINUOUS LOOP.
    GET AN INCOME FROM YOUR PROGRAMMING
    SKILLS.

**********************************************
```

If you think that your programming ability could provide you with an income, but are hampered by lack of expensive hardware, then **Mr.Software** may have the answer.

We're currently looking for programmers, skilled in both "BASIC" & 6502 Assembly Language who have a flair for graphics and a lively imagination. If the description applies to you, why not send **Mr.Software** a sample of your program at the address below.

If your work shows talent and promise we will lend you the hardware to enable you to earn an income from your programming skills.

Mr.Software
18-20 Steele Road,
Park Royal Industrial Estate,
London NW10.

COMAL CORNER

By Brian Grainger.

I think this article of COMAL Corner is going to mark then end of an era for COMAL. This is not to say I have finished with COMAL, but that the language has grown up and will begin to be used more and more.

I am happy to say that ICPUG has produced the first COMAL for the Commodore 64. Being based on Version 0.12 COMAL it provides virtually the full COMAL Kernal definition, over 12000 bytes of user space and simple disk handling on a 64 while still being usable if only cassettes are available. Like all ICPUG programs it is FREE. At the same time, the COMAL Handbook, the definitive manual, is now available in quantity from Prentice Hall. There is now no reason why COMAL should be rejected.

Looking to the future, Commodore seem to have committed themselves at last, if their leaflets at the Commodore show are to be believed, to producing a COMAL cartridge for the 64. This will be upward compatible with the ICPUG disk and includes a new set of commands to control the graphics and sound as an additional option. There are even rumours that a Commodore machine with COMAL as standard may appear, but as yet this is only rumour.

With all the new interest that the new COMAL will produce it is clear that some definition and usage instructions must be available for newcomers. The article 'What is COMAL', elsewhere in the Newsletter, is the first of such articles.

To return to the other COMAL versions I have a veritable pot-pourri of news. Firstly Malcolm Friend is considering converting COMAL for use with old ROMs (there are still such users around!). Best of luck, Malcolm.

Phil Oliver wrote me the sort of letter usually printed in newspaper advertisements to endorse a product. He tells me he has an 8096 and COMAL 1.02 in his dental practice. He is rewriting his large BASIC programs e.g.

Payroll, into COMAL. He finds COMAL much easier than BASIC to use. The use of long variable names makes his programs meaningful just by reading, without the need to keep lists of what his variables are used for. He also finds debugging is made easier because of the procedure facilities. Finally he says the programs run 4-5 times faster than BASIC which cannot be bad!

Graham Baxter, another user, has highlighted a potential problem. COMAL only allows up to 256 identifiers. An identifier is a variable name, PROC or FUNC name or label name. The implementers of COMAL for the CBM have been requested to look at increasing this capacity. However the use of closed procedures will allow increased identifiers because a further 256 are allowed in the scope of each closed procedure. If you should reach the limit it becomes difficult to reduce the number of names. Deleting an identifier from the program will NOT delete it from the table of identifiers. To do this the program must be LISTed to disk and reENTERed.

Graham was also irritated by the inability to format text in the PRINT USING command. So was I until I found out how to do it. The following little program should give you the idea to solve the problem.

```
0010 DIM NAME$(5) OF 10, DEPT$(5) OF 10
0020 DIM CLOCK'NO(5), TEL'NO(5)
0040 FOR I:=1 TO 5 DO
0050  READ NAME$(I),CLOCK'NO(I),DEPT$(I),TEL'NO(I)
0060  NAME$(I):=NAME$(I)+"<10spaces>" // PADS STRING
0070  DEPT$(I):=DEPT$(I)+"<10spaces>" // DITTO
0080 ENDFOR I
0090 FOR I:=1 TO 5 DO
0100  PRINT USING NAME$(I)+"   ###   "+DEPT$(I)+"   ####":
      CLOCK'NO(I),TEL'NO(I)
0110 ENDFOR I
0120 //
0140 // TEST DATA
0150 //
0160 DATA "CHARLES",123,"SALES",6258
```

```
0170 DATA "BRIAN",456,"QUALITY",6349
0180 DATA "BARBARA",789,"AUDIT",6610
0190 DATA "STAN",135,"PROJECT",6310
0200 DATA "GERRY",246,"MANAGEMENT",6985
```

During May I had the opportunity of going to Commodore in Slough to see Len Lindsay, of COMAL Handbook fame, as well as the new COMAL 2.0 with graphics board running on an 8096. Those who came to the Commodore show will also have seen this version and I am sure will have been impressed by the graphic facilities. I want to devote a separate article to this version when it becomes available but all the previous rumours I gave considering the version are true and more. It is very, very impressive and I am told its speed is close to obviating the need for a compiler. The graphics were magic! A map of Denmark was drawn on the screen and slowly town names and natural features were overlayed as if they were sprites on a C-64. The code that did this was so concise as well.

To finish this article I want to mention that Mr. Want wished to use COMAL in an environment where several machines were accessing a single disk. Obviously the error messages could not come from disk in this environment. I thus made a modification for version 0.12 which printed an error number rather than an error message. I close with details of the modification should others require it.

1) DLOAD"COMAL80-0.12/1"
2) SYS 1024 to enter the monitor.
3) Use the .M command to modify memory as follows:

```
.:1586 86 3A A2 A0 A9 16 20 38
.:158E 27 A4 3A A9 00 20 20 5C
.:1596 A2 00 A5 3A A0 10 D9 A7
.:159E 15 D0 01 E8 88 D0 F7 86
.:15A6 35 60 05 0B 0D 0E 15 18
.:15AE 19 20 21 23 25 26 27 28
.:15B6 2A 35 EA EA EA EA EA EA
.:15BE EA EA EA EA EA EA EA EA
.:15C6 EA EA EA EA EA EA EA EA
```

```
.:15CE EA EA EA EA EA EA EA EA
.:15D6 EA EA EA EA EA EA EA EA
.:15DE EA EA EA EA EA EA EA EA
.:15E6 EA
```

4) Type .X to return to BASIC

5) DSAVE"COMAL80-0.12/1EM"

--o0o--

SIPOD NOTES

Sipod is essential for those PET owners who have also bought a C-64 or Vic-20. The hardware consists of a thin 5-way ribbon cable with a user port connector for the PET end, and a serial port connector at the other.

The software is a program which when loaded into the PET and run, will enable the 64 or Vic to drive any CBM disk drive and printers connected to the PET. The PET must be left running all the time that access to CBM peripherals is required, and a reset of the 64 or Vic will not affect the disk or printer.

The latest version of Sipod (v184) works on all BASICs (even BASIC 1 !). The disk commands used are those recognised by the 64 or Vic, that is BASIC 2. If you have DOS Support for your 64 or Vic, then that will also work. I have used the interface for several weeks without any trouble.

Mick Bignell, who developed Sipod, is a member of ICPUG and a small discount is offered to members. Sipod is available from Microport, 7, Clydesdale Close, Borehamwood, Herts, WD6 2SD. Tel: 01-953 8385. Price £ 25.00+VAT.

Jack Cohen.

--o0o--

REVIEW

C-64 PROGRAMMERS REFERENCE GUIDE £ 9.95

I am deeply indebted to John Durrows, a North Herts ICPUG man who lent me his 64 Programmers reference guide. It is a copy of the early version from the States but how long can we wait for the Commodore to produce the English version! By the time you read this, hopefully, they will be in plentiful supply, but I have said that before.

Down to business. This is the book the serious 64 programmer has been waiting for. Dare I say it is the best manual Commodore have produced. To give some idea of its scope I shall reproduce the contents and an indication of what is covered:

Introduction

Overview of BASIC + Techniques

BASIC Vocabulary (Keyword by Keyword)

Graphics- (standard character mode, multicolour mode, extended colour mode, Sprites, Hi-res)

Sound- All about waveforms and the SID facilities

Machine Code- Complete Kernal definition, memory map, opcodes explained

I/O devices

In addition the appendices cover character POKE and CHR$ codes, Music note values, VIC register Map, I/O Pinouts, error messages, specifications for all the new chips and a circuit diagram for the 64.

This book is not written for the beginner but for the serious programmer or hardware buff. Nevertheless the guide will be useful to anybody who has understood the mediocre user guide that comes with the 64.

As far as I am aware there is no facet of the 64 left uncovered and I guess that means the purpose of the book is fulfilled. One of the things I learnt was that one could set up raster interrupts so that the screen could be split, one portion in standard character mode, the other in Hi-res mode.

I found some errors in the guide, notably that the CHR$ table was a copy of the VIC equivalents and therefore omitted the new colours. A handy Sprite definition chart only included 20 rows of 24 bits when 21 rows are needed! On game ports it suggests the fire button is pressed when bit 4 of a certain location was 0. An example program suggests that fire button is pressed when bit 4 of that location is 1.

One interesting point was in the description of cassette load. If no filename is specified it searches for the first file and stops. If the CBM key is pressed it loads, if a space is pressed it continues to look for the next says the guide. On my 64 it ALWAYS loads the first program it finds. I wonder if this is a software bug in the 64 or a drop off in the manual.

Hopefully some of the more noticeable errors will be removed from the English edition. With any sense it will remove the suggestion that 'good' programmers 'crunch' their program lines. Anybody who has read my COMAL articles will know what I think of those pearls of wisdom??

All in all, an excellent book which should lessen the need for Raeto West to produce 'Programming the C64'! One word of warning. Make sure you only pay the Commodore recommended retail price (£ 9.95). Some shops have been known to charge more.

Brian Grainger.

--o0o--

40/80-COLUMN CONVERSION

By Colin Spencer.

I read with interest the article in the recent issue of this magazine about a 40/80 conversion. I recently had my 4016 upgraded to 32K and switchable 40/80 by Mick Bignall of Microport. Mick performed this conversion while I waited (about two hours). His modifications need the whole PET not just the circuit board. This is because he implements all the 80-column features. This is accomplished by removing the locking capability of the SHIFT/LOCK key, this turns it into the equivalent of the CONTROL key on an 80-column machine. All the 80-column functions such as: set windows,tab,delete line and scroll up. These functions are all possible by holding down the SHIFT/LOCK key and one other key or it shifted. This means that all 80-column software works on this machine with no other modifications.

I find the set window and delete rest of lines particularly useful as the majority of my time is spent programming. The tab facility is also very useful when using word processors such as Superscript on which this was written.

The cost of this conversion from Mick is £ 89 for the permanent upgrade or £ 105 for the switchable upgrade. The question you must ask is 'Do I need to be able to switch my machine'. I find that my machine spends 95% of its time in 80-column mode. The other 5% of the time I use the CBM 4032 program by C.CHEE to make my machine into a 40-column one.

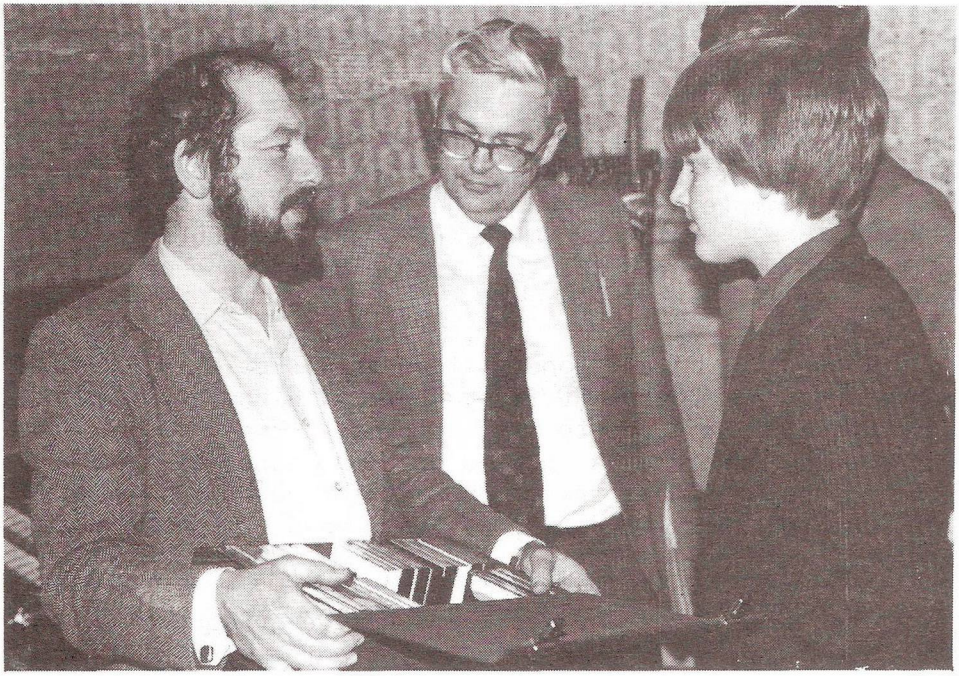
All of the above refers to 12" screen PETs but by the time you read this, 9" screen PETs WILL be upgradeable to 80 columns by Mick with the addition of a small circuit board inside your PET and all the above features will be possible on your machine. The cost of this upgrade will be £ 149.

Was the conversion good value for money? The answer must be yes my 4016 cost about £ 400 and the upgrades another £ 150. I now have a 4032/8032 with the advantage of the graphics keyboard which in my opinion is far superior to the business keyboard for a programmer.

If you are interested in the upgrades that I have mentioned you can contact Mick Bignall at MICROPORT, 7, Clydesdale Close, Borehamwood, Herts. Tel: 01-953-8385.

--o0o--

Slough Region first birthday celebrations



"Psst, want to buy some software...."

STRICTLY FOR BEGINNERS

By Ray Davies

RELATIVE FILES.

This to me is an extremely complicated subject to explain. But I will do my best. For further information, see, for example, Programming The PET/CBM by Raeto West (an ICPUG member). Relative files are only available (easily) with BASIC 4.0, and so this is the basis upon which this article is written.

A relative file is one written to diskette in the same manner as a sequential file, but with another file of pointers also written to the diskette, which point out where each 'record' is placed on the diskette. The user need not bother about the second file of pointers, which is what the DOS (Disk Operating System) uses to place and to find each record.

The main important difference between Sequential and Relative files is the ability to write to, and read from, any record on the file DIRECTLY. i.e. without having to read all the records sequentially before finding the one you want. Not only that, but one can write to or read from particular bytes in any record if required. These differences speed up access considerably. My programs will demonstrate the speed of access by using the Record Number, which is the quickest way of searching Relative files. But they will also demonstrate normal search techniques.

As I have said before, the best way to learn is by example. Here is a program to demonstrate Relative Files in action:

```
10 PRINT"<clr>LINES 600 TO 630 HAVE BEEN RESERVED FOR"
20 PRINT"YOU TO INSERT YOUR OWN TITLES. FOR A"
30 PRINT"NEAT APPEARANCE ON SCREEN THESE SHOULD"
40 PRINT"BE OF EQUAL LENGTH, FILLED OUT WITH"
50 PRINT"EITHER FULL STOPS OR SPACES"
60 REM PROGRAM PROPER
70 PRINT"FOR 18 DATA INPUTS OF UP TO 29 CHARS.<cd>":
   DIMB$(18),D$(30)
```

```
75 REM RECORD LENGTH, L, INCLUDES A CARRIAGE RETURN
80 FORA=1TO18:READB$(A):NEXT:PRINT"FILENAME IS 'FILENAME'":
   F$="FILENAME"
85 REM READ TITLES & DEFINE FILENAME
90 DOPEN#1,(F$),D0,L30:IFDSTHENPRINT"<cd>"DS$
95 REM OPEN A CHANNEL WITH TITLE, DRIVE NUMBER AND RECORD
   LENGTH SPECIFIED
96 REM ALSO PRINT ERROR MESSAGE IF ERROR ON OPENING CHANNEL
100 INPUT"<cd><rvs>R<off>EAD       <rvs>W<off>RITE
    <rvs>Q<off>UIT<rt><rt><rt>Q<lft><lft><lft>";A$
110 IFA$="R"THEN240
120 IFA$="W"THEN150
130 IFA$="Q"THENPRINT"<clr><12dn><14rt> GOOD-BYE !":
    FORI=1TO750:NEXT
140 DCLOSE:END
150 INPUT"<dn>RECORD NUMBER";R
160 PRINT"<clr>":FORA=1TO18
170 PRINTB$(A):NEXT
180 PRINT"<home><dn>";:FORA=1TO18:INPUT"<6rt>"D$(A)
190 IFLEN(D$(A))>29THENPRINT"<dn>TOO LONG":GOTO170
195 REM CHECK ENTRY IS LESS THAN 29 CHARACTERS LONG -
    RE-ENTER IF TOO LONG
200 NEXT
210 RECORD#1,((R-1)*18+1)
215 REM ACCESS RELATIVE FILE NUMBER R. ALLOW FOR 18
    SEPARATE INPUTS
220 FORA=1TO18:PRINT#1,D$(A):NEXT
225 REM WRITE DATA TO RELATIVE FILE NUMBER R
230 IFDS<>0THENPRINT"<dn><rvs>"DS$
235 CHECK FOR ANY DISK WRITE ERROR
240 GOTO100
250 RESTORE:FORA=1TO18:READB$(A):NEXT
255 REM RESTORE DATA POINTER TO BEGINNING OF DATA & READ
    DATA AGAIN
260 PRINT"<clr>SEARCH BY:"
270 INPUT"<dn><rvs>R<off>ECORD NO., <rvs>N<off>AME, OR
    <rvs>A<off>DDRESS";Q$
275 REM INDICATE FIELD TO BE SEARCHED
280 IFQ$="R"THEN400
290 IFQ$="A"THEN520
300 IFQ$<>"N"THEN260 : REM TO PREVENT ACCEPTANCE OF ANY
    OTHER KEYPRESS
```

```
310 INPUT"<dn>NAME";N$:L=LEN(N$):TI$="000000"
315 REM TI$ IS ONLY FOR  TIMING (OR FUN)- NOT  AT  ALL
    NECESSARY FOR PROGRAM
320 R=0
330 R=R+1:PRINT"<rvs>"R
335 REM PRINTING R TO SCREEN TO SHOW YOU HOW THE SEARCH IS
    PROGRESSING
340 RECORD#1,((R-1)*18+1)
350 IFDS=50THENPRINT"<dn>"DS$:GOTO10
355 REM ERROR 50 'RECORD NOT PRESENT' - YOU HAVE REACHED
    THE END OF THE FILE
360 FORA=1TO18:INPUT#1,D$(A)
370 IFLEFT$(D$(1),L)<>N$THEN330
375 REM N$ NEED ONLY BE ENOUGH CHARACTERS TO IDENTIFY THE
    NAME YOU REQUIRE
380 NEXTA
390 GOTO430
400 INPUT"<dn>RECORD NO.";R:TI$="000000"
410 RECORD#1,((R-1)*18+1):IFDS=50THENPRINTDS$:GOTO100
420 FORA=1TO18:INPUT#1,D$(A):NEXT
430 PRINT"<clr>":FORA=1TO18:PRINT"<rvs>"B$(A)"<off>"D$(A):
    GOTO460
440 PRINT"<clr>":FORA=1TO18:PRINT"<clr><rvs>"B$(A)"<off>";
    D$(A)
450 GETQ$:IFQ$=""THEN450
460 NEXT:PRINT"<dn>RECORD NO.";"<rvs>"R"<off> FOUND IN ";
470 PRINTMID$(TI$,3,2)" MIN."MID$(TI$,5,2)" SECS."
480 INPUT"<dn>REPEAT<3rt>N<3lft>";Q$
490 IFQ$="Y"THEN640
500 GOTO100
510 INPUT"<dn>ADDRESS";C$:L=LEN(C$):TI$="000000"
520 R=0
530 R=R+1:PRINT"<rvs>"R
540 RECORD#1,((R-1)*18+1)
550 IFDS=50THENPRINT"<rvs>"DS$:GOTO100
560 FORA=1TO18:INPUT#1,D$(A):Y$=D$(3)
565 REM D$(3)=ADDRESS FIELD
570 NEXTA
580 IFLEFT$(Y$,L)<>C$THEN530
590 GOTO430
600 DATANAME,F'NM,ADD1,ADD2,ADD3
610 DATADOB.,HIST,R.E.,L.E.,BIN/,
```

```
620 DATAM&F.,DATE,DIST,NEAR
630 DATAOTH.,COST,ORD.,P &T
640 PRINT"<dn>ALL(A) OR ONE AT A TIME(O) ?"
650 GETQ$:IFQ$=""THEN650
660 IFQ$="A"THENPRINT"<dn>":FORA=1TO18:
    PRINT"<rvs>"B$(A)"<off> ";D$(A):GOTO460
670 PRINT"<clr>":FORA=1TO18:PRINT"<rvs>"B$(A)"<off>";D$(A)
680 GETQ$:IFQ$=""THEN680
690 NEXT
700 PRINT"<dn>REPEAT 'Y' OR 'N'?"
710 GETQ$:IFQ$=""THEN710
720 IFQ$="Y"THEN640
730 GOTO100
```

I have copied this as faithfully as I could from a printout of my own program. I trust that it is accurate, (I have renumbered as I went along so that you only need to type in the lines having a line number ending in 0), and that it works. If not, please send me a diskette, packed so as not to be bent in the post, and sufficient return postage, and I will send you a copy of the original and definitely working program.

As an aside to this, the above program can be altered in several respects to both improve speed of operation and to save on memory. Anybody care to comment how this can be done? No prize for the best suggestions.

Here is a skeleton program to demonstrate how to read or write to particular bytes in a record. (Courtesy of Robert Watts, one of our Derby & District Branch of ICPUG members.)

```
1 DATA20,20,20,20,20,20,6,6,6,20,6,3,3,20,20,3,3,3,4,4,2,
2,2,20,20,4
2 DATA4,10,10,4,20,20,7,20,7,20,7,20,6,6,6,6,6,6
5 DIMR%(50),N$(50)
10 R%(1)=1
20 FORA=1TO44:READX:X=X+1
50 R%(23)=1
55 R%(A+1)=X+R%(A)
60 NEXT
```

```
 100 DOPEN#1,"REL DATA",L(254),DO
1000 INPUT"RECORD NO.";R
1010 IFR<1THENDCLOSE:END
1100 INPUT"READ OR WRITE";A$
1200 IFLEFT$(A$,1)="R"THENF=0:GOTO2000
1210 IFLEFT$(A$,1)="W"THENF=1:GOTO1600
1500 GOTO1100
1600 FORA=1TO44:PRINT"FIELD NO.";A:INPUTN$(A):NEXT
2000 GOSUB6000
3000 FORA=1TO44:PRINTN$(A):NEXT:GOTO1000
6000 R1=R*2-1:R2=0:K=0
6003 FORK=1TO44
6005 IFK>22THENR2=1
6010 RECORD#1,(R1+R2),R%(K))
6020 IFF=0THENINPUT#1,N$(K)
6023 IFLEN(N$(K))=0ORN$(K)="↑"THENK=50:GOTO6030
6024 IFF=1THENIFN$(K)=""THENN$(K)="*"
6025 IFF=1THENPRINT#1,N$(K)
6030 NEXT:RETURN
```

The data statements in this program, written specifically at my request for use in my professional career as an Ophthalmic Optician, refer to the length of the 44 data fields I wish to input, i.e. the number of bytes each piece of data will occupy. These are all integers, hence the R% array to hold these numbers.

Care is needed in the choice of how many fields are required, and the length of these inputs. In this case, to make neat screen prints, we decided to print half to the screen at a time, and purely by coincidence this worked out perfectly. Each half added up to 254 bytes! So two screenfuls of my information were contained in each twc Records. Had this not been the case, I would have had to have increased the number of records to three, and not had such an even screen presentation. But that is a minor detail.

Line 10 sets the first byte pointer to 1. This is where the record starts to read or write. Line 20 reads the byte lengths, and stores them in an array. Note the X=X+1

to allow for the carriage return at the end of each input. Lines 50 and 55 set up the second record, which stores the final 254 bytes of my data. (A record cannot be longer than 254 bytes).

Line 100 opens a channel and a file called in this case 'Rel Data', telling the disk the record length to expect, and the drive upon which to work. At this stage, the disk will create space for the first record. Line 1010 is a way to exit the program, by entering 0 for the record number. By the way, if you STOP your program at any time, always DCLOSE so that any open files are closed properly. This may be one cause of data corruption which is at present being blamed on the use of the "@0: etc" command!

Lines 1200 and 1210 set flags 'F' for either read (0) or write (1). Line 1600 accepts your information into each field. Line 2000 then takes you to the 'meat' of the program. The GOSUB 6000 part is the actual Relative Access part of the program. Because my information is too much for one record, it is spread into two records. But I only wanted one record number, so the formula R1=R*2-1 is required to find the value of R. Then during the inputting of the data, line 6005, R2 is brought into use to fill up the second record. This is why in line 6010 we record to file number R1+R2. The extra R%(K) sets the file pointer to R%(K) bytes past the previous position. Line 6020 checks the flag F for read or write, and we branch accordingly to either PRINT#1 or INPUT#1. Line 6023 checks whether the record is blank or if it has not been written to before.(In which case DOS will fill the record with ↑'s). If so it goes on to the next input.

Line 6024 checks first for the Write flag (F=1). Then if the record is blank it writes a '*' into it.

Finally one more program courtesy Rob. This is another way of using relative files, and also demonstrates screen printing techniques combined with inputs. (N.B. Rob uses strange sequences of line numbers, so I have renumbered this program to be more easily typed in by beginners).

```
 10 BL$="<29spaces>"
 20 DOPEN#1,"REL DATA",L254
 30 GOTO160
 40 REM
 50 N$(2)=LEFT$(N$(2),7)
 60 N$(8)=LEFT$(N$(8),7)
 70 PRINT"<home)<3dn>SURNAME ";N$(1)
 80 PRINT,,"<up>:FIRST NAME ";N$(2)
 90 PRINT"<2dn>ADDRESS ";N$(3)
100 PRINT"<8spaces>";N$(4)
110 PRINT" ";N$(5)
120 PRINT" ";N$(6)
130 PRINT"<2dn>LEFT LEG ";N$(7)
140 PRINT,,"<up>:RIGHT LEG ";N$(8)
150 RETURN
160 FORA=1TO8:N$(A)=BL$:NEXT
170 PRINT"<clr>":GOSUB40
180 POKE16,1
190 INPUT"<home>PATIENT NO. ";P$:P=VAL(P$)
200 IFP=0THENDCLOSE:PRINT"<clr>":POKE16,0:END
210 FORA=0TO7
220 RECORD#1,(P),(30*A+1)
230 INPUT#1,N$(A+1):IFN$(1)="↑"THENA=7
240 NEXT
250 GOSUB40
260 POKE16,1:INPUT"<home><16rt>ALTER Y/N ";A$
270 IFLEFT$(A$,1)="N"THEN160
280 IFLEFT$(A$,1)<>"Y"THEN260
290 POKE16,1
300 IFDS=50THENA=A
310 PRINT:INPUT"<2dn>SURNAME ";N$(1)
320 PRINT:PRINT,,
330 INPUT"<up>:FIRST NAME ";N$(2)
340 PRINT
350 INPUT"<2dn>ADDRESS ";N$(3)
360 PRINT
370 INPUT"<8spaces>";N$(4)
380 PRINT
390 INPUT" ";N$(5)
400 PRINT
410 INPUT" ";N$(6)
420 PRINT
```

```
430 INPUT"<2dn>LEFT LEG ";N$(7)
440 PRINT:PRINT,,
450 INPUT"<up>:RIGHT LEG ";N$(8)
460 POKE16,0
470 FORA=0TO7
480 RECORD#1,(P),(30*A+1)
490 PRINT#1,N$(A+1)
500 NEXT
510 GOTO160
```

The POKE16,1 is a 'foolproof input' ploy, which ensures that the program will not crash if you press RETURN by mistake. In fact the program will not move on until you enter something. POKE16,0 returns the program to normal.

Note the syntax for RECORD. Instead of giving a figure after the #1, a variable is used. These variables must be enclosed in brackets as shown. Also if file names have been allocated to a string variable, e.g. Y$="File Name", then this too must be enclosed in brackets - RECORD#1,(P),(Y$).

Once again, if any of these programs do not work, this will be due to my mistyping or the processing. So if you wish to obtain a working copy without typing, just send me a well-packed disk plus return postage and I will copy them all onto your disk for you. My address is 105, Normanton Road, Derby, DE1 2GG.

--o0o--

MEMBERS PRIVATE SALES & WANTS

For sale, CBM3022 printer in near mint condition. Price £ 250 o.n.o. from D.A.A.Fagandini, 6, Alleyn Park, Dulwich, London, SE21 8AE. Tel: 01-670 0547.

Commodore disk drive wanted for 4032, contact E. M. Alexander, 46, Prestwich Park Road South, Prestwich Manchester, M25 8PE. Tel: 061-773 3883.

--o0o--

FORTH COLUMN

Interrupts.

Two methods are presented here to put Forth hi-level words in an interrupt service routine.

1. As a CODE definition:
Assign a name to the ISR and code as
HEX CODE <name> <assembly code for ISR> RTI, END-CODE
then having tested the assembly code, one only needs to point the IRQ vector to point to it thus:
' <name> (fetch pfa) 0090 ! (address of IRQ in PET)
2. As a code fragment:
Code is assembled into the dictionary and the start loaded into the IRQ vector.

```
HEX ASSEMBLER ( include in assembler vocabulary)
HERE ( locate dictionary address & put on stack)
<assembler code for ISR>     RTI,
0090 ! ( store dictionary address in vector)
```

Code should be provided to enable and disable the interrupt routine e.g.

```
CODE DISABLE SEI, <restore vector> CLI, NEXT JMP, END-CODE
```

Sometimes it may seem somewhat heavy to load the Forth Assembler in its entirety when only a few bytes of code are required. In BASIC one does this occasionally with a few well-chosen POKEs. In Forth it can be done simply by compiling literals into the dictionary and arranging for the code field address (cfa) to point to it. The code must point back to the inner interpreter NEXT. The following is an example:

```
0 ( KBD -- GETS A KEY IF PRESSED USING JSR $FFCF  RDG8304)
1 FORTH DEFINITIONS HEX
2
3 CREATE KBD                          ( GET A KEY TO TOS   *)
4          0586 , 20 C, FFE4 , FBF0 , 05A6 , 48 C, 00A9 ,
5          4C C, 063B , ( PUSH )    SMUDGE   DECIMAL   ;S
```

Square roots:
The following definition will return an integer which is the square root of the value on the stack. Only 16-bit values are catered for.

```
0 ( MATHS - 1      SQRT                          RDG-830219 )
1 : SQRT ( N --- SQRT                  PERFORMS SQUARE ROOT *)
2      2 BEGIN OVER OVER / SWAP DUP ROT + 2 / SWAP
3              OVER - ABS 2 <   UNTIL SWAP DROP ;
```

R.D.G.

--oOo--

ROUND THE REGIONS

During May the Watford Group had a record attendance at which Micronet 800 was on demonstration.

Slough/Berks Group had a change of venue to a different drinking establishment where Steve Beats of Commodore was booked to bring along some new product releases.

South Hants Region meet on the second Tuesday of each month, most of which are of an informal nature. A recent evening had a 'ham radio' presentation at which computers were used to improve communication. Members' equipment covers almost the entire Commodore range.

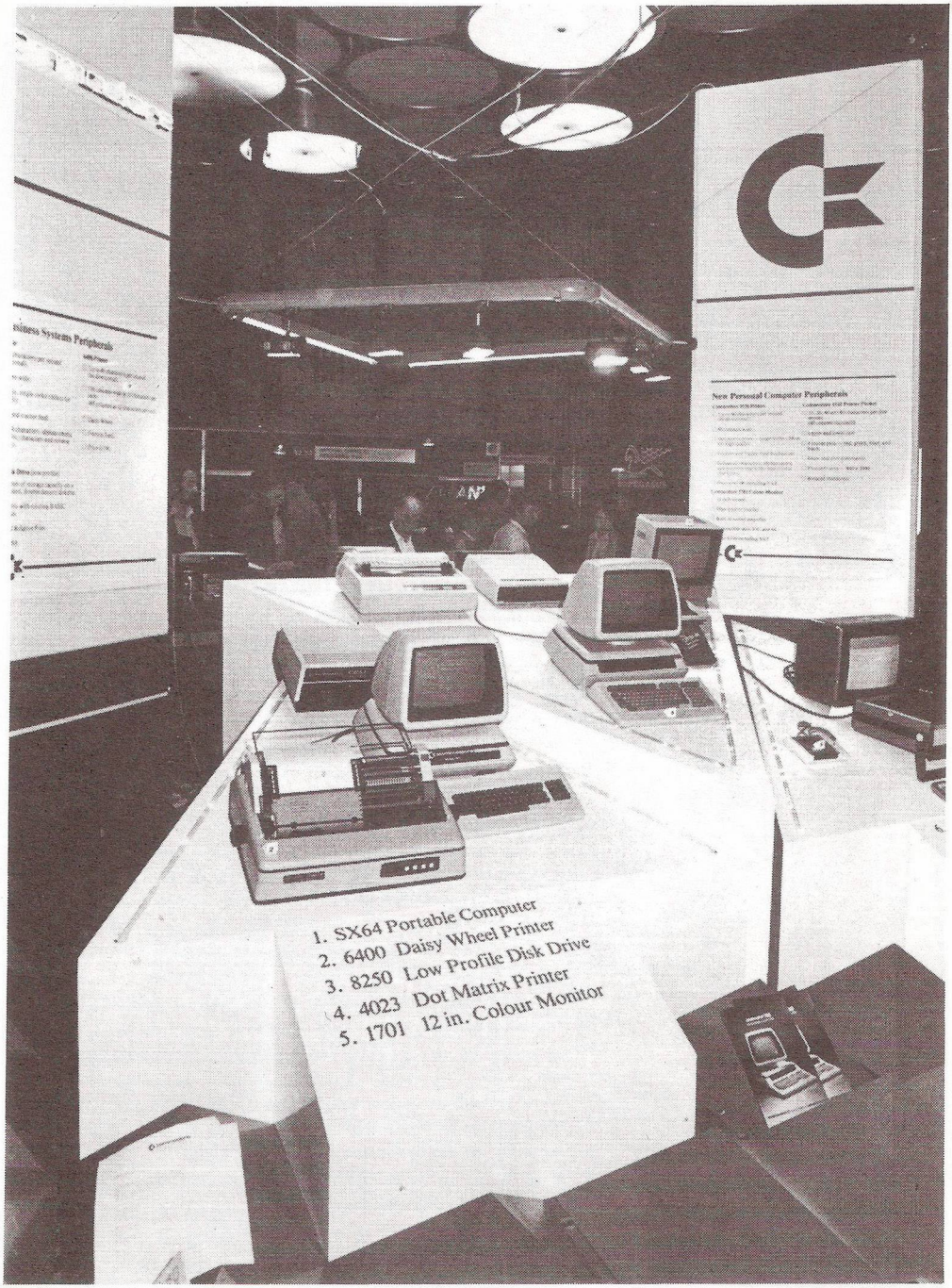
North Hants Region are looking for a larger venue, but meanwhile meetings continue and recently a programmer's clinic proved very popular - didn't realise so many members were having problems with their programs, a repeat session will no doubt be held in due course.

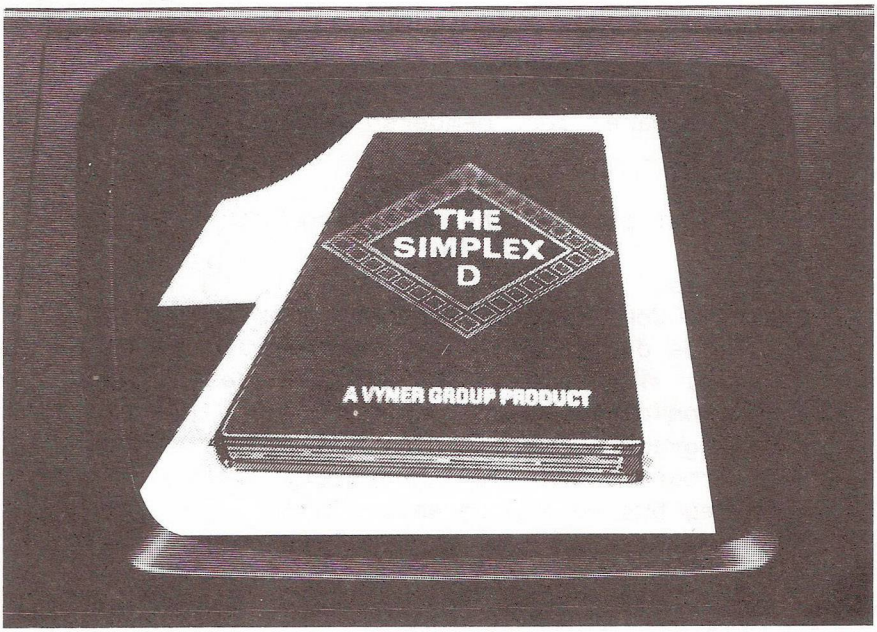
A new group has sprouted at Gosport. Meetings commence at 7p.m. on the 1st Friday of each month (excluding August) at the Gosport Community Association (next to the Cottage Hospital), Bury House, Bury Road, Gosport, Hants, PO12 3PX. Contact Tony Cox, Fareham (0329) 280530 for further details.

--oOo--

THE 'ICPUG' STAND

Barnes
Products


New Personal Computer Peripherals
1. SX64 Portable Computer
2. 6400 Daisy Wheel Printer
3. 8250 Low Profile Disk Drive
4. 4023 Dot Matrix Printer
5. 1701 12 in. Colour Monitor



The Electronic Cash Book

Micro-Simplex makes Retailers Accounts and Stock Control simple...

Unique features:

- Based on Britain's No. 1 cash book system
- Uses Britain's No. 1 business micro computer
- The only one recommended by Vyners, publishers of Simplex books
- The only one offering all retailers special V.A.T. schemes

Other features include ...

- Stock control linked to cash registers
- Simple and familiar layouts
- Easy to use
- Automatically produces:
  (a) Statements to customers
  (b) Lists of unpaid bills
  (c) Simple profit and loss accounts

MICRO-SIMPLEX

MICRO-SIMPLEX

OFFICIALLY APPROVED BY COMMODORE

commodore COMPUTER

Contact any Commodore Dealer

or

Micro-Simplex Limited,
8, Charlotte Street West,
Macclesfield, Chesire.

Tel: 0625 615000

THE SOFTWARE LIBRARY

By Bob Wood.

The software library situation is such that most of the library's administrators are overloaded with work. The following notes are for members' guidance in using the library.

There are at present the following disks:

1. PET Programs:
(a) All of the Commodore Workshop programs
(b) 2 utilities disks, 3 educational disks, 2 music disks, 1 Comal disk, 1 business disk, 3 games disks and 1 assembler add-on disk.
2. Vic-20 Programs 4 disks - a mixed bag but mainly games.
3. CBM 64 Programs Only 1 disk as yet, mainly games but including a few business programs.

Members may write to Bob Wood requesting a list of any programs that are available for any of the three machines. They must enclose a stamped, addressed envelope for return. They should then copy the list and return the original when requesting programs.

Members may obtain copies of programs from the library on either disk or tape, subject to the following restrictions:
TAPE - up to 4 programs per request.
DISK - 4040 ONLY - 1 or 2 disk copies per request.
Please note that only complete disks will be copied. It is not feasible to provide a selection of programs from different disks.
Members should send the following:
(a) The media - tape or disk(s)
(b) Adequate re-useable packaging
(c) Sufficient postage for return
(d) Self addressed adhesive label
(e) Original list
(f) State - What you require
Your system configuration

8050 disk programs are available from Stephen Rabagliati, c/o: IGD, Grange Lane, Letchmore Heath, Watford, Herts WD2 8DQ.

Members may submit programs to the library for evaluation. Usually they will be accepted, provided they are not subject to copyright, duplicates of other holdings, or rubbish.

--oOo--

MICRONET NOTES

Micronet have not provided Commodore Users with the priority that they were promised since the days when ICPUG did most of the development work. ICPUG had Prestel on display on the CBM range well over a year ago. The BBC did not successfully develop their equipment until last September, yet the BBC version was launched with great publicity and 'red carpet' treatment, although it did not fully qualify for launch on Micronet - one condition was that 100 programs had to be available for it. Despite 'stealing' the ICPUG library software to put on Micronet in the BBC name, it was launched with less than the requisite number of programs.

At the time there were no modems were available to the CBM owner and there had been a rumour, strongly denied by Micronet, that by the time the CBM version was launched, all of the first 10,000 cheap modems would have been allocated to BBC computer owners.

At least one member has discovered that his 9" 4032 cassette-based system is unacceptable to Micronet and has had his application returned. Despite careful examination of the literature, no specific reference was found to this exclusion.

Looking at the money invested in Micronet - £ 500K each from Prestel, Prism Microproducts and East Midlands Allied Press, and £ 250K from the Department of Industry for technical development and staffing - one cannot be too amused to find that the original advertised prices have now escalated by over 50%.

Our Chairman had complained to Micronet that even as an Information Editor for ICPUG pages, he was unable to look at the CBM software that was available on Prestel. Micronet had tried to put the blame on the ICPUG effort, when in fact Micronet had been under-resourced and the database was not ready when promised. Similar dissatisfaction had been expressed by the Amateur Computer Club (ACC) Committee. The ACC are threatening stronger action if the situation is not improved.

Apparently Commodore are not particularly impressed and are seriously taking matters in hand with their own scheme. Commodore have been interested in networking for some while and this could bring matters to a head. Having seen Micronet demonstrated at the Watford Region, it was interesting to see the error-rate being blamed on the switch-board connection. Commodore had their cheap direct-connect modem feeding a C-64 with no problems.

Mike Todd and I both applied to Micronet last March. Mike had his cheque returned with a note to say that Commodore users would be subject to delay. My application was processed as a BBC user. I received and duly returned a Beeb modem. Several letters, some polite, others not so, have failed to change matters. I have been billed by Prestel for a service that I do not have and cannot receive. I have cancelled by application. I still have not received my money back and as I write, I have just received my 'free manual for the new BBC systems software'.

In September, Business Micronet will be launched. My experience of the typical businessman is that they will not be so tolerant as the amateur of Micronet's shortcomings.

R.D.G.

--oOo--

Advertisement

THE VIC COLUMN

By Mike Todd.

INTRODUCTION - FOR ALL NEW MEMBERS

First of all, let me express a warm welcome to all the new ICPUG members who have joined as a result of seeing us at the recent shows. Let me first introduce myself.

I write the 64 and disk columns in the Newsletter as well as anything else that may be of interest. So, although I look after the Vic side of ICPUG, there are many other aspects of Commodore computers that have my attention.

My interests range across the board from the way that the machines work to business applications, from explaining the fundamentals of using Commodore products, to showing how you can get the most out of the machine you are using.

ICPUG has a variety of "officials", all of whom have some specialised knowledge or area of interest and these officials, together with the many non-committee members who are also experts in their field, provide a huge information resource. Technical queries should initially be directed to our technical secretary, Jim Tierney, who will forward them to the individuals responsible. Other queries can be directed to the individuals listed in the front of the Newsletter.

All ICPUG "officials" devote their time and efforts to the Group on a voluntary basis and in their spare time. Nearly all have full time jobs to keep themselves occupied when they're not working for ICPUG!

So, if you do have to contact any of us, please bear this in mind. We'll always try to get back to you as soon as possible, but it may be a matter of a couple of weeks before we can find the time to catch up with our mail. And if you do write, enclose a SAE if possible - you're much more likely to get a quick reply, and will save ICPUG funds from being used up on postage.

For my own part, I always prefer queries in writing as I can fit them in to my schedule far better than personal calls, or phone calls. That way, you'll get a far better response. But, remember to give as many details as possible about the set up that you are using and the problem that you have. I often get queries which simply say things like "I have a Vic-20 and keep getting an error in line 20 of Vic-Writer" which is of little use in identifying the problem!

VIC BOOKS

There is now a huge range of books available for Vic users, and I thought I'd devote the Vic Column this time to a look at as many as I could get my hands on.

I would have liked to have tried every example program and read every book in great detail, but time could not possibly allow such a luxury. Therefore, most of the reviews that follow are really confined to a description of the books with only a small amount of comment.

I have tried to indicate the level at which the book is aimed, so that beginners and "experts" alike can see which will suit their needs. But, don't forget, a book that is just a little more advanced than you might immediately want could be usefully employed to learn new techniques.

I would welcome any additional comments from members and will include these in future columns.

All books are readily available either at good bookshops or through ads in the computer press, but watch out for pricing differences, especially when it comes to imports.

The prices I quote are the official publisher's price, unless followed by (*) which indicates that I don't know this price and that the price quoted is the price I paid for the books (yes! I bought many of them). I've also mentioned size and number of pages.

ZAP! POW! BOOM! - Mark Ramshaw
Interface Publications - £7.95 (*)- 8"x12" - 52pp

A collection of 30 games program listings, all copied from the Vic printer. The print quality is not brilliant but is as readable as most magazine listings.

Games include Star Trek, a primitive adventure game, one armed bandit, draughts, a plane lander game, Space Invaders, Asteroids, Nightmare Castle, Missile Command and a Lunar Lander. As you can see, some old favourites are there, but cut down to fit into an unexpanded Vic.

All screen POKEs assume an unexpanded (or at most, 3K expanded) Vic and there appears to be no use of the special graphics available.

I suppose at 26.5p per game, it's probably cheaper than many of the computer magazines!

SYMPHONY FOR A MELANCHOLY COMPUTER - Tim Hartnell
Interface Publications - £6.95 - 8"x12" - 64pp

Another 24 games, again copied straight from a computer printout but a little easier to read than the last one. The games, which were written by a variety of authors, include Fruit Machine, Othello, Breakout, Mastermind, Checkers, Hangman and Evolution (Life).

Some programs are rather trivial, but there are quite detailed explanations of how they work and most have photographs of their screen displays to give you some idea of how the game looks.

There is a useful appendix section, with reference tables of screen locations and character codes, sound generation and colours.

Like the last book, fair value, but some of the programs really are trivial.

GAMES FOR YOUR VIC-20 - Alastair Gourlay
Interface/Virgin Books - £2.95 - 5"x8" - 123pp

A collection of 32 games (that's less than 10p each!) with the usual collection of Breakout, Fruit Machine, Hangman, Othello as well as Pontoon, simple piano keyboard and sketch board programs and some new variations on old games.

Listings are fairly easy to read, if a little small, and all should run on an unexpanded Vic.

There is a section at the end, written by the editor of the series, Tim Hartnell, about writing your own games programs. This is followed by a glossary of computer terms and a bibliography.

Undoubtedly good value at the price.

VIC INNOVATIVE COMPUTING" - Clifford Ramshaw
Melbourne House Publishers - £6.95 (*) - 5"x8" - 147pp

Another 30 games listings including Battleships, Rat Trap, Hangman, Golf, Snakes and Ladders and, wait for it .... CHESS! It's the real thing (well, sort of), but with lots of the usual refinements missing and unlikely to be of Grand Master class!

The listings are extremely easy to read - better than any I've seen before. This is mainly due to the fact that they have been "processed" to separate out the graphics characters and make them easier to identify. Even the onerous task of counting and typing spaces is helped by a special space symbol.

There are colour photographs of the screens of the games, and simple descriptions of the programs.

You can buy the programs on three cassettes, but why buy the book if you're going to cheat and buy them ready typed?

VIC PROGRAMS - VOL 1 - Nick Hampshire
Duckworth - £6.95 - 5"x8" - 184pp

Although this book consists mainly of games, there are a couple of utility-style programs such as a character editor for creating your own high-resolution characters, and a machine code monitor program.

The games include Hangman, One-armed bandit, Moon Lander, Rubiks Cube, Supermind (= Mastermind) and Breakout.

Most of the programs appear quite good and many use the Vic's capabilities better than the other books. Mind you, I can't help feeling that I've seen many of these programs somewhere before!

There are actually several programs which demonstrate the use of the high resolution mode on the Vic and all programs are preceded by a description of the hardware needed to run them. In most cases that is an unexpanded Vic, although some need 3K expansion and/or a joystick.

Again, should you want to cheat, there are two cassettes available direct from the publisher at £7.95 the set. Pretty good value, I would say.

VIC GRAPHICS - Nick Hampshire
Duckworth - £6.95 - 5"x8" - 185pp

I mentioned this in the previous VIC COLUMN, so won't go into too much detail other than to say that it contains a variety of programs to demonstrate the use of graphics using the SUPER EXPANDER.

It doesn't always explain what it's doing very well, but with a bit of reading between the lines many of the examples could be applied to your own programs - there are also two cassettes available to accompany this book.

LEARNING TO US THE VIC-20 COMPUTER - Ron Geere
Gower Publishing - £4.95 - 6"x8" - 76pp

Yes, it is our own Ron Geere who wrote this one, or rather adapted the book for the Vic-20 to accompany a whole series of "Learning to use.." books for other micros.

It introduces the concept of the Vic and takes the beginner through the first traumatic stages of turning the computer on and using the cassette deck to LOAD programs.

Some of the BASIC programming language is explained with a few simple example programs.

The book's objectives appear confused, and it is unlikely to be other than of limited use for the total beginner - although the book does explain how to draw a butterfly on the screen and make it flap its wings (not in high resolution graphics I might add). There's also a brief mention of how BASIC programs are stored.

I'm sorry, but it wouldn't be my first choice.

START WITH BASIC ON THE COMMODORE VIC-20" - Don Munro
Tiny Publishing - £4.95 (*) - 7"x10" - 124pp

This book, lavishly sprinkled with illustrations by Bill Tidy, is a good introduction to programming in BASIC.

It's first step is to demonstrate how to write a two line program, and from that point, BASIC programming is devloped through a series of simple examples. There's even a chapter on POKEs and what we use them for.

The examples illustrate the concepts fairly well - giving the newcomer a chance to get the Vic doing something right from the start. Like most books, a reasonable familiarity with elementary mathematics is assumed.

The chapters also cover colour and sound generation, the concept of simple graphics (not high-resolution, which would have been out of place in such a book), simple sorting and data handling as well as introducing the mathematical functions available in a chapter which can be omitted if you're not up to it.

VIC-20 FOR CHILDREN - Tony Noble
Sigma Technical Press - £5.95 (*) - 10"x7" - 152pp

This book is really an educational resource book for parents and teachers and starts by examining educationalistic (I have a feeling I've just invented a new word!) aspects of computing. The introduction says it won't be long before your children are asking "Will you show me how to program?", and it is part of the aim of the book to help you cope with this.

Using a series of step by step examples, the child is given simple programs to type in and run. It takes him key by key through a series of simple programming examples which will fill the screen with the child's name and generate some graphics - all of which should be quite appealing.

The rest is mainly for adults and looks at more advanced programming concepts, including the writing of flowcharts. There are a couple of listings of simulation programs which in themselves could be useful teaching aids.

The book ends with a series of games-style programs, which are all educationally based and would serve as valuable computer familiarisation exercises (provided that someone had already typed them in!). The concept of computer keyboard literacy is really very important, perhaps more than the accepted ideas of computer literacy, and these example programs would provide a valuable resource in this area.

It's aimed at teaching adults how to program, but with the thought that there will inevitably be a child looking over your shoulder - a useful and practical book for home and school alike.

MASTERING THE VIC-20 - A.J. Jones, E.A. Coley, D.G.J. Cole
Ellis Horwood - £5.95 (*) - 7"x10" - 177pp

If you already have a reasonable working knowledge of BASIC programming and want to get deeper into the Vic, writing more advanced programs using high-resolution graphics and sound, peripherals and machine code, then maybe this is the book for you.

The contents range from describing Vic BASIC, the Vic's structure, graphics, peripherals (including a brief examination of disk, cassette and printer), accessories (including joystick and using the expansion ports), system architecture and machine code programming. With the appendices containing a Star Trek game listing, high resolution (including high-res printing of the screen on the printer) and some reference tables, this could be a very valuable book to have around.

It's not an easy book to read by any means, and is really for the those with enough background and experience to be led fairly quickly through the concepts. But, what it does, it does reasonably thoroughly and well - in fact, in many ways, it's what VIC REVEALED ought to have been.

The description of machine code programming, like the rest of the book, is treated in the form of a text book rather than a tutorial. Therefore, it would require more than just a read through to learn the concepts described.

It looks at the Vic's system architecture and the way that machine code is used on the Vic. Many of the internal locations and routines in the Vic are mentioned, but there is a lack of detail in this area.

Certainly not a beginners book but seriously worth considering if you are going to do more than BASIC programming. I liked it very much.

VIC REVEALED - Nick Hampshire
Duckworth - £9.95 - 5"x9" - 267pp

I think that I've said more than enough about this book already, so I'm simply going to say don't buy it. And that means both the first and second editions.

BEGINNERS ASSEMBLY LANGUAGE PROGRAMMING - Dr. P. Holmes
Glentop Publishers - £15.00 (*) - 6"x8" - 201pp

Learning machine code on any computer isn't easy, and assembly language is a way of making it less difficult. To remember that the jump command (equivalent to BASIC GOTO) is given by the number 76 (or $4C) and is followed by a two byte address, in the wrong order, is more difficult than remembering that it is given by JMP followed by the address.

The purpose of an assembler is to allow programs to be written using this much easier form, as well as allowing you to allocate storage space (in the form of single or multiple bytes) by assigning names to them, rather than constantly having to refer to them by their actual addresses.

It describes techniques needed for assembly programming, and even includes listings of a couple of simple assembler programs (the price of £15.00 includes a cassette containing these programs). Also included are a simple machine code monitor (written in BASIC!) and binary/hex tutorial programs.

The scope of the book is really quite wide, with an adequate teaching style, but the mnemonics used are not the accepted standard which could make transferring to a full scale assembler rather difficult.

It's a fair introduction to a difficult subject, with some coverage of the Vic's internal workings and more advanced techniques such as interrupts, screen output, the USR command, floating-point numbers as well as multiplication and division techniques.

COMPUTE'S FIRST BOOK OF VIC
Compute Magazine - £10.95 (*) - 6"x9" (spiral bound) - 212pp

This American book is a distillation of 36 articles from COMPUTE magazine, which has for a long time been one of the best magazines available for users of Commodore machines.

Unfortunately, it has been difficult, not to mention expensive, to obtain in the UK - so it is very gratifying to see the Vic-oriented articles being reprinted as a separate volume.

The articles are grouped into six sections, the first is called "Getting started". This covers the origins of the Vic, of computers in general, using the Vic as a calculator, generating large letters and using games paddles and joysticks.

Section two has five articles under the heading "Diversions - recreation and education". They are mainly games, with well written commentaries.

"Programming techniques" is by far the largest section and ranges from using the PRINT statement to appending programs, from the vagaries of the quotes key to getting the most out of the memory available.

The fourth section covers colour and graphics, including high-resolution and custom character and finally sections five and six cover the internal structure of the Vic and machine code.

There's even a full dump of TINYMON, a machine code monitor for the Vic.

The "infamous" Jim Butterfield (sorry Jim!) is a regular contributor to the magazine, and four of the articles included are by him. The rest of the book is up to the same high standard, and it's really like buying two years subscription to the magazine, but at a fifth of the price!

VIC-20 PROGRAMMERS REFERENCE GUIDE
Commodore - £9.95 - 6"x9" (spiral bound) - 284pp

I've already said a lot about this in other columns, but to summarise, this is a useful reference book for both BASIC and machine code, and is, I guess, the definitve source book from which most of the others are compiled.

It's not always easy to find what you want in it, nor is it always 100% accurate, but it is a very useful book for all Vic owners (other than those who are just going to plug in games cartridges).

I can't think of anything that it misses out, although it doesn't always give "recipes" for doing things, but at least it does give enough information for you to work it out for yourself - which is really what computing is about anyway.

TRICKS FOR VICS - Sam D. Roberts
Elcomp Publishing - £5.50 (*) - 5"x8" - 115pp

An odd collection of a few games programs (including Air Battle and Submarine), machine code programs and hardware projects, this book is an interesting read.

It's certainly not a teaching book, more of a projects book and even contains some hardware information for those building their own expansion boards. It also includes printed circuit board layouts for these.

The section on programming the input/output chips (mainly the user port) is a useful section for those wanting to interface appliances to the Vic. But, beware, the book is of American origin and is based on American techniques, although the components used should be readily available in the UK.

It's not the sort of book I would go out of my way to buy, but there will be some who will find it invaluable.

SUMMARY

Well, that's a look at fifteen of the books available to the Vic user - there are others, and I would welcome comments on books I've not covered as well as those I have. With so many to choose from, it is really a difficult task to select the best of the bunch, especially as different users have different needs.

It is a popular misconception that, if you buy a computer and a good book, you'll be writing superb programs within a couple of weeks - or, add a disk drive, and start writing business programs a week later! You wouldn't expect to buy a Jumbo jet, and learn to fly it after two weeks of reading a book, would you?

Learning about computing is no easy task, but the motivation is often greater than learning differential calculus or the History of the Ancient Roman empire. It is here that the books provide resource material with the computer providing motivation. But no book will substitute for hands-on experience, the brain power and enthusiasm.

Some books, like the games books, require very little knowledge to type in the programs - but they do need an understanding of the keyboard, the graphics characters, the screen editing and the cassette deck. And, if the programs don't work, they need patience to find out where the typing errors are, as it's almost certainly you at fault, rather than the program - or at least in most cases!

So, what do I recommend? Well, "Games for your Vic-20" is the best value, and worth buying, although the others aren't bad. For more advanced programming, the "Programmers Reference Guide" is worth having, as is "Mastering the Vic-20". As a general book, I would strongly recommend "Compute's First Book of VIC". I would also strongly recommend trying to see a copy of the book before you buy it!

--oOo--

SUPERSPELL NOTES

By Barry Biddles.

As a Superspell user, I have come to rely heavily on it and it came as a surprise to find spelling errors getting through the checking. The explanation is not entirely to the discredit of Superspell, but merely to that of the manual. The instructions in section 2.1.4 on p7 say that the 'display' (of unrecognized words) may be aborted using the <STOP> key, but fail to mention that it is then INVALID to go on and EDIT the file as usual.

The program makes two passes through the text file; the first to build up the statistical analysis, and the second to flag any unrecognized words. So if the second pass is aborted, Superspell DOES NOT KNOW ABOUT subsequent spelling mistakes. Ideally, Superspell should refuse to accept an edit in these circumstances. Failing that, the manual should make the point clear, and Precision Software assure me that the correction will be included in the next printing.

Editing the Master Dictionary.

Tom Cranstoun suggests that users of Superspell should print out and very carefully check the contents of their user dictionaries, before merging them into the master dictionary, because once a spelling mistake has been merged, it cannot be edited. This is good advice, because although there IS a sneaky way to edit the master dictionary, it is a little complicated, and is not the kind of procedure that one would want to repeat regularly.

The key is on p12 of the manual, where we are told that if a word appears in BOTH of the dictionaries, then during an attempt to MERGE them there will be a WORD ALREADY EXISTS message, with the offer DELETE (Y or N). If you enter 'Y' the word will be deleted from the master dictionary. Clearly, this procedure could be used to edit the master dictionary, if only one could get the word into the user dictionary first.

This would normally be impossible, since if the word is in the master dictionary, it will not be flagged as an error, with the option of being learned into the user dictionary. The trick is to take a new disk and INSTALL a master dictionary with nothing in it! The procedure is explained in section 1.3. When a text file is run using this master, EVERY word in it will be flagged as an error, and may be learned into the user dictionary. Then go back to the original user dictionary, and do a MERGE.

Use of Wildcards ('*' etc.).

When specifying the name of the text file to be processed by Superspell it is sometimes tempting to use the usual CBM pattern matching, with the wildcards '*' and '?'. The file will indeed be found, and will be processed, but Superspell will crash when it attempts to write the file back to disk under the same name, because it doesn't know the full filename!

The Final Bug ?

It has been a long time now since a bug was discovered in Superscript. Mick Ryan has discovered that when setting up text with a text window much wider than the screen, it is possible to reach the bottom of text memory sooner than one expects. From this position, using a TAB that would ordinarily take you to the same place on the next line will instead take you into the no man's land outside text memory, and Superscript will hang up. The problem has been brought to Simon Tranmer's attention, and of course he cured it in five minutes flat!

Reproduced courtesy SE Region.

--o0o--

COMAL - AN INTRODUCTION

By Brian Grainger.

I have recently been justifiably criticised by some ICPUG members because I talk of COMAL as if everyone knew about it. With the advent of COMAL for the C-64 and thus a lot of potential newcomers for the language I intend to put right this anomaly and start from scratch in explaining the purpose and use of COMAL. This first article will serve as an introduction to the language. I hope to continue next time with the start of a series on how to program in COMAL. You have 2 months to get your copy of COMAL and be ready!

WHAT IS COMAL ?

COMAL is a program language designed to be easy to use, like BASIC, while having powerful features like PASCAL.

COMAL programs can be entered from the keyboard and run immediately without need for intermediate steps.

COMAL programs can be edited easily using the Commodore Screen Editor. List the lines, move the cursor to the line to be changed, change it using the cursor movement keys if necessary, hit 'return' and the line is revised.

COMAL expands on the syntax of BASIC making it easy for beginners to learn and easy for those already programming in BASIC to understand the new commands.

COMAL adds to Commodore BASIC those missing features which you need to make programming easier.

COMAL supports structured programming.

COMAL HAS THE FOLLOWING FEATURES:

Auto line numbering, renumbering and block delete functions are standard.

COMAL checks for syntax errors when program lines are input and positions the cursor over any error for ease of correction.

COMAL allows variable names to be up to at least 16 characters long, all being significant, thus eliminating the use of codes for variable names.

COMAL allows program subsections to be called by name and can pass parameters to and from the subprograms.

Variables in subprograms can be isolated without fear of conflict with identically named variables elsewhere in the program.

COMAL allows standard routines from a program library to be merged into existing programs, thus saving development time.

COMAL includes disk commands with a simple syntax.

COMAL includes IF...THEN...ELSE, WHILE...ENDWHILE, REPEAT...UNTIL, FOR...ENDFOR and CASE...ENDCASE structures.

COMAL supports redirected output to a printer.

COMAL allows user-defined functions, (with no restriction on definition length), which can be called from the keyboard or programs as if they were part of the language.

COMAL programs, when LISTed, are AUTOMATICALLY indented to show the program structure.

COMAL adds certain keywords for readability automatically without the user typing them.

COMAL is FAST. On average 4 times faster than Commodore BASIC but string handling can be up to 55 times faster!

WHO SHOULD USE COMAL.

Those beginning programming who want to make use of simple commands and the language's error checking facilities and easy editing.

Those teaching programming who want to follow the lead of the Danish, Swedish and Irish education Authorities in teaching an easy language which introduces concepts needed for programming in later life.

The businessman who wants to reduce his program development and maintenance time and get his computer working FOR him rather than against him.

WHAT IS AVAILABLE:

The COMAL semi-compiler is available as public domain software, (read FREE), for the following systems:

CBM 3032 with disks
CBM 3032 with cassette
CBM 4032 or 8032 with disks
CBM 4032 or 8032 with cassette
CBM 8096 with disks
****
COMMODORE 64 with cassette and/or disks
****

The COMAL ROM board is available for any 3000-, 4000- or 8000-series computer, of any memory size and with cassette and/or disks.

The COMAL cartridge is coming for the Commodore 64 with cassette and/or disks.

HOW TO GET IT!

All public domain versions are available through and supported by ICPUG. Send a C-60 cassette or diskette, (preferably for 4040/1541/2031 but 8050 format supported), return postage, and details of version required to:

Brian Grainger,
73, Minehead Way,
Stevenage,
Herts.
SG1 2HZ

The COMAL ROM board is available from:

Ellis Horwood Ltd., Market Cross House, Cooper St., Chichester. PO19 1EB.

The COMAL cartridge is not yet available. Look out for news in the ICPUG Newsletter.

--oOo--

SHOP WINDOW

Now that computers are so cheap (compared to a few years ago), the problem is to find a cheap printer. Well, here's one at £ 79.90 (incl. VAT) more or less. In fact it entails getting a Sinclair ZX-Printer from Sinclair Research, Camberley, GU15 3BR, at £ 59.95 and a Softex 'Printerface' from Softex Computers, 37, Wheaton Road, Bournemouth, BH7 6LH at £ 19.95, plus £ 1.00 p&p (tel: 020 0202 422028). Features include high-res graphics and plotting, full PET/Vic and user-defined character sets, 42-column printout, and Commodore PRINT/LIST command compatible. 8000-series PET users should enquire before ordering.

Sometimes there are problems with paper in the area of your printer. £ 28.50 (incl VAT & delivery) buys the Advanced Desk-top Printer Stand which accomodates the printer on top, the supply paper underneath (in its carton if required), and the output neatly fan-folded into a tray. Suitable for 80-column printers and available from Advanced Resources, St. Gabriels, Much Birch, Hereford, HR2 8HY. Tel: (0981) 540262.

If you are heavily into interfacing in a professional environment, you may be interested in the Wild Multiplexer EMP1. It can interface up to four RS232 ports to the IEEE-488 bus with up to 2K of buffering. The unit and interface parameters are fully programmable. These include baud rate, character length, RS232 and IEEE end characters, SRQ enable and 8 buffer lengths. The unit is very

comprehensive and commands a price tag of £ 1,846 plus VAT. Contact P. Broxham, Wild Heerbrugg (UK) Ltd., Revenge Road, Lordswood, Chatham, Kent, ME5 8TE. Tel: Medway (0634) 64471.

The Cleveland Interface Domalarm is a 16-channel interface for the Vic-20. Each channel can be individually programmed to monitor normally open or normally closed contacts, and provide delayed, or immediate response as required. Useful for process control, many other ideas and examples are supplied on cassette. Available in kit form or built and tested. Prices £ 24.95 built & tested, £ 19.95 as a kit, including VAT & postage. ICPUG discounts 12+1/2% on kit, 10% built up - quote membership number. Available from Cleveland Interface, 18, Chelmsford Avenue, Fairfield, Stockton, Cleveland.

Supersoft now produce a catalogue specifically for C-64 users. Many of the products come from among the best-selling items in the well-established PET range. The MIKRO assembler is there, now on a cartridge with a monitor which includes HUNT and TRANSFER commands. GRAPHIX-64 is based on the software from the Hi-res boards for the PET, but in the 64 version colour commands are added, and a window command gives split screen capability. It loads into spare RAM above BASIC. VICTREE adds over 40 commands, including BASIC4-style disk commands and enchanced 'Toolkit' functions. ZOOM is a powerful machine code monitor with facilities galore. No space to describe them here, or any of the other goodies. Obtain the 64 cataloue from Supersoft at Winchester House, Canning Road, Wealdstone, Harrow, HA3 7SJ. Tel: 01-861 1166.

JCL's Business ROM is now available in enchanced form for the 700-series. In addition, they produce a ROM/RAM board for the 700 with 3 x 8K, any of the three being selected as ROM or RAM. The C-64 IEEE cartridge with many enhancements, including mini-DOS Support and SRQ interrupt in BASIC. JCL Software are at 47, London Road, Southborough, Tunbridge Wells, Kent. Tel: (0892) 27454.

R.D.G.

--oOo--

THE ADMINISTRATOR

The original transaction processing system which can learn and run repetitive functions. It allows the user to create his own records and also his own menu for job selection. Widely used by all types of business.

STOP PRESS
CBM 64
DATA BASE
LAUNCH

TYPICAL LIST OF APPLICATIONS

Appointments Planning	Office Administration
Bank Accounting	Parts List
Bonus Schemes	Personnel Records
Book Keeping	Petty Cash Ledger
Bureaux de Change	Plant/Asset Register
Contract Costing	Portfolio Management
Cost Ledgers	Price Lists
Credit Control	Property Management
Customer Files	Purchase Ledger
Diaries	Rota Planning
Equipment Leasing/ Rental/HP	Route Planning
Estate Agents	Royalty Payments
Estate Management	Sales/Purchase order files
Expense Accounting	Service Records
Invoicing	Statistics
Job Costing	Stock Control
Mailing	Test Data Storage
Medical Records	Theatre Event Costing
Membership Accounting	Time Costing
Name & Address files	Vehicle Costing

and many others

The Businessman's Dream

STAGE ONE SOFTWARE

300 Ashley Rd. Parkstone
Poole Dorset 0202-735656

SOME MORE MEMBERS QUERIES

By Mike Todd.

NUMERIC KEYPAD FOR VIC-20

Derek Hoare, of Crawley, has seen an article in Electronics Today International (June 83) which describes the addition of a numeric keypad to the Acorn Atom and wonders if the project could be extended to the Vic. He has an 18-way calculator keypad and would like to use this for the characters 0123456789+-/*=., and DEL.

Has anyone tried and succeded in doing this?

VIC TYPEWRITER

The Commodore VIC WRITER program is designed to work with the Commodore 1515 printer with its narrow paper.

Gary Marker of Huntingdon is using it with the 1525 printer which allows wider paper to be used - but, as a result, the text is no longer centred on the page. He would like to have this problem solved. I have no experience of this program and have only been able to have a peep at a listing which indicates that the bulk of the printer handling routines are in machine code.

Has anyone modified the program to allow the text to be centred on the page?

ODD COLOURS ON THE VIC

Some time ago, I had a letter from David Fairhurst who has experienced some very odd colour effects on the Vic when a 16k RAM pack is added, with an expansion board. There appears to be no obvious explanation and I wonder if anyone has experienced similar problems, or, better still, solved them.

--oOo--

REVIEW

WORDPOWER — Reviewed by Alan Birks.

WORDPOWER is a word processing package written by Kevin Pretorius.

The program will run on 3000-series, 4000-series, Fat-40 and 8000-series PETs and may be used with CBM 2040 and 4040 disk drives. It can also be used with cassette units if you do not have disk drives. Text may be printed on CBM or standard ASCII printers.

The program can be supplied on cassette or floppy disk and is accompanied by a 19-page Guide. The Guide is principally aimed at the disk user but I found it perfectly usable in describing how to use the cassette option. For some people (me) the temptation of a new piece of software is too great and it is loaded and run before the instructions are read. Having done that and duly crashed the PET (I had loaded the wrong program) I settled down for a read. The Guide includes a helpful example which introduces the method of working quite painlessly.

WORDPOWER provides facilities to perform all the jobs which I would expect a word processor to do. Just to run through the facilities: there are 15 modes of operation and the commands available vary according to the mode. On entering the program you are in EDIT mode. In this mode as with all other modes, the available options are displayed on the top four lines of the screen, particularly helpful if your memory is less than perfect. The Edit mode's function is to permit access to other modes. It is the top level menu giving access to specific functions and can be returned to from those specific functions by pressing the '@' symbol. The '@' symbol cannot be directly inserted in text because of the control key use. The function modes available are:-

INSERT - for typing in new text or adding to existing text.

DELETE - for removing unwanted characters or lines

LOAD - permits a file of text to be loaded in from disk or cassette.

SAVE - permits the text currently in the PET to be saved onto disk or cassette. On disk it is saved as a sequential file.

COPY - allows you to hold selected text in a separate area . This has various uses, for example it can be used to shuffle paragraphs of text into a new order or to insert a common phrase a number of times while only entering it in full once.

EXCHANGE - this is actually a means of overwriting existing text.

DISK OPERATION - this allows normal disk operations for example a look at the disk directory from within the program.

DEFAULT REVISE - allows the function of the program to be changed from disk to cassette operation and back again. You may also specify the type of printer, etc. Once the defaults are set they may be stored and will be used in future sessions until you choose to change them.

FIND/REPLACE - will search through the current text for a given string and replace it with another. If no replace string is given then FIND without replace occurs.

AGAIN - repeats the most recent FIND command.

TOP - moves cursor to the start of current text.

BOTTOM - moves cursor to end of text.

NEXT - takes cursor to the next page of the text.

PREVIOUS - takes cursor to the previous page of the text.

WRITE - all important, this gives the results. Text may be displayed on the screen or printed, and a useful set of facilities for setting margins, centering text, leaving header and footing space, multi-line spacing, multiple copies and specialised formatting are provided. The format commands are embedded in the text by embedding them within square brackets.

When in EDIT mode most subsidiary modes are accessed simply by pressing the key matching their initial letter but 'x' is used for exchange, the RVS key is used to revise selections and for disk commands '>' is used.

I would not claim to have rigorously experimented with every facility of the word processor but I have used it for my correspondence for over a month. After that trial would I recommend it to other people? Yes, at forty pounds it has a substantial price advantage over most competitors and for the person who does not possess disk drives the ability to use cassettes as an alternative is all important. Another attractive feature is the provision of a guide to the currently enabled functions at the top of the screen. There are drawbacks as well. To get the '@' symbol into the text I had to type a '#' move the cursor back and then FIND the '#' and REPLACE it with '@', while square brackets cannot be used at all. These are minor limitations though. I also found use of the EXCHANGE command disconcerting at first since one can only overwrite the number of characters originally in the target line. Never-the-less while WORDPOWER may have minor flaws it is a workmanlike robust package and I rate it a good buy for anyone seeking an economical word processor.

--o0o--

MATTERS ARISING

In the January issue (p66), it was implied that the COPY/ALL upgrade, COPY/FAST will copy from anything to anything. Maybe so, but its limitation is that it is written for BASIC 4.

--o0o--

REVIEW

And So Forth T. Huang, 1983 revised

The book is described as a 'College level textbook' and comprises over 370 pages. The text is divided into four parts (what else), called know why, know how, Victor Forth and Appendices.

The first two chapters cover the philosophy and history of Forth, the latter being something which will be more of interest to older hands at computing.

Chapter two (they number from zero up) is unusual, but important in that it discusses software quality. In Forth, quality of software can make all the difference between a well-documented work and a 'write-only' program.

D-charting is detailed in the chapter headed 'Tools'. A D-chart is an alternative style of flowchart. Normal flowcharting is not very successful, which is probably why it is rarely used. D-charts overcome many of the limitations of conventional flowcharts and are used in planning Forth programs.

The 'know how' section explains the language fundamentals in one chapter and the system editor in another. The book also contains one of the best explanations I have seen on Forth's virtual memory system and gives an example of a virtual array.

'Know how' continues with the Forth model, but for the Intel processor family (8080/Z80 and 8088/8086 - yes I know that the Z80 is not Intel's) and with CP/M disk format, neither of which resemble 6502 implementations in detail, but the basics are there. One of the rare pieces of information shown is the USER variable area map including offset indices.

Chapter 10 discusses Forth's string handling operations, including numerical conversion. Successive chapters cover interpreters, compilers, defining words and vocabularies and it is these chapters that can lead the

Forth novice into to more deeper aspects of the language environment.

Part 3 is perhaps the least useful to Commodore users, referring as it does to the Victor 9000 machine with 128K RAM and CP/M-86. At this point one discovers that the author is responsible for Victor-Forth on the machine's 8088 processor. Chapter 16 is entitled 'Only for Victor' and describes the Victor-Forth word set. Some of these definitions may provide a source of inspiration from the description, but the actual definition is not in the book. Part 3 ends with a chapter of examples, but contains nothing outstanding.

Part 4 contains a description of a line editor, two glossaries, one alphabetic, one by function and a guide to the screens on Victor-Forth (pretty useless without the system). The book ends with some exercises, but gives no answers.

I hesitate to recommend this book, for excellent though it is, it is machine-specific in too many places. It is a large book, telephone directory size and at nearly £ 20, not cheap. However, it contains some good in-depth material and you may not mind having to pay for the irrelevent stuff to get it. I recommend you try and see a copy before deciding.

R.D.G.

--oOo--

ODDS AND ENDS

By Brian Grainger

Some useful, and not so useful snippets from the past two months.

1) Want to know how to ensure a tricky repair job! Remove your video connector and reconnect it such that the white lead on the end does not make contact first. Because the video area is not sufficiently grounded a high voltage can

persist for a considerable period of time which blows up a transistor in the video circuitry when the lead is reconnected incorrectly. The transistor does not cost much but it isn't easy to get at. Solution, connect a large resistor (at least 820K) between the white lead and the chassis.

2) The value to send on Secondary Address 6 with an 8023 is one third of that required with a 4022 printer to get the same line spacing.

3) To test for a STOP key on the C-64 check if location $91=145 is equal to $7F=127.

4) CHR$(10) characters will now be accepted by a cassette connected to the C-64, unlike the PET where the operating system filtered them out.

5) If you change the interrupt vector on the C-64 and subsequently use the cassette, your vector remains intact when control is returned to the 64. Clever the 64, the PET used to crash !

6) Which company gave a product code to some public domain software ? Why Commodore to the ICPUG COMAL for the C-64 of course !

7) Much new software for the 64 came to the club at the Commodore show including some on Jim Butterfield's latest disk. It includes SUPERMON for the C-64.

8) Why did the person who wrote SUPERMON for the C-64 use the delightful colour combination of black on blue ?

9) After extensive tests on one CTV and one monochrome I consider the ideal C-64 readable colour combination suitable for both is white(1) on medium grey(12). Orange(8) on Brown(9) came a close second. For CTV only Green(5) on White(1) was quite colourful.

--o0o--

40 TO 80 COLUMN CBM CONVERSION

By R.C.Harvey.

The March 83 issue of the ICPUG magazine contained a review of the HARJIN 40 to 80 column switchable conversion for a 4000-series machine with a 12" screen (page 169). Since the review a modification has been made which now allows the ESC and TAB keys to be implemented. The modification does not involve any changes to the hardware but it does mean that any software intended for the 8032 machine requiring the ESC and TAB keys, can now be run without any problem (e.g. Wordpro 4, Superscript and Ozz). The price of the conversion remains at £ 90.00 incl. and further details are available from Robin Harvey, 30, Wimborne Close, Coombe Glen, Cheltenham, Glos. Tel: (0242) 27588.

--oOo--

SOFTWARE CLEANUP

The newly-formed Computer Trade Association, which represents the interests of retailers, distributors, software houses, manufacturers and consultants, is waging a campaign to control the growth of unauthorised software libraries. It also aims to fight against software piracy and clarify the legal position.

This does not, of course, affect our ICPUG software library, since its contents are Copyright-free. It could affect those companies handling programs on a try-it-and-see basis, and those 'renting'programs in the way that public libraries loan music recordings.

R.D.G.

--oOo--

TURBO-PET.

Greenwich Instruments, of "Instant ROM" fame, have produced an exciting new enhancement to the CBM/PET machines. This is a 'Turbo-PET' conversion of the 6502 which, when tested against a well known series of benchmarks, makes the PET outperform all other machines, by nearly 40%. BASIC, compiled BASIC and machine code all run four times faster. The speed is obtained by using Greenwich Instruments NRV64, a 64K non-volatile RAM as all of the PET's memory, together with a 4MHz 6502.

The contents of the BASIC ROMs together with any utility chips are down loaded into the NRV64 when the machine is switched on. As NRV64 is non-volatile an applications program can be left in the PET at all times, ready to be used at next power up, with an auto-run facility. Of course this feature can be ignored, if desired, and programs loaded normally. BASIC/utility chips can be altered as desired as they are now in RAM. All or part of it can be replaced by another language or special program. This can be loaded from disk and can stay in memory until the user desires to switch back to BASIC.

Disk, tape, printer, the PET's internal clock and all other input/output work normally. 16K PETs are automatically upgraded to 32K when Turbo-PET is fitted.

Turbo-PET simply plugs in, no soldering or hardware modifications are required (returning to a normal PET is, obviously, a simple operation).

Greenwich Instruments have announced a competition in conjunction with the release of this system. "Although the 6502 runs at 4Mhz, TURBO-PET in fact gives a speed improvement of 4.3 in BASIC programs. The first letter received (by Greenwich Instruments) giving the correct explanation for this will receive a free adaptor for using NRV64 in a turbo-pet system."
Greenwich Instruments can be contacted at: 22, Bardsley Lane, Greenwich, London, SE10 9RF.

T.C.

--o0o--

TECHNICAL TIPS

Vicmon, Saving BASIC and Machine Code.

There are several ways to save a combination of BASIC and Machine code (usually called a Hybrid program); details of the easiest method (which will allow the program to be loaded with only one typed command) are given here.

It should be noted that when developing a hybrid program with VICMON you must always enable a Virtual Zero Page (VZP) each time you enter Vicmon, to ensure that the vital pointers to BASIC are protected.

Rules for a VZP are:

1) If you have a hybrid the VZP must exist above BASIC.
2) The VZP must be positioned away from your machine code program.
3) The actual locations useable depend on the amount of RAM available, options could be:

A. No expansion or 3K expansion	VZP=$1D00
B. 8K expansion	VZP=$3F00
C. 16K expansion	VZP=$5F00
D. 16K+8K expansion	VZP=$7F00
E. 16/8K +3K expansion	VZP=$0400

It should be stressed that these are typical values, it depends on the location of your Machine Code routines. Remember that a VZP will occupy 256 Bytes of RAM, so if you have a VZP at $1D00 it would occupy all the locations from $1D00 to $1DFF.

Various methods of creating your Hybrid:

Method 1). Convert the machine code to data statements within a BASIC program which pokes the code into memory each time the program is run. (A program is available from the CLUB library which will do this automatically).

Method 2). Save the machine code program followed by BASIC. To use the program type:

```
LOAD"MACHINE CODE",1,1 <return>
NEW                    <return>
LOAD"BASIC",1          <return>
RUN                    <return>
```

Note that the BASIC should contain the code to protect the machine code from being overwritten.

Method 3). Save BASIC and machine code as per method 2. The BASIC should contain something similar to the following:

```
 1 IF PEEK(56)=29 THEN 10
 3 POKE56,29:CLR:LOAD"MACHINE CODE",1,1
 10 REM MAIN PROGRAM....
```

Line 1 checks to see if the top of memory pointer has been moved (location 56 is the high byte, 55 is low). The location checked and value checked for vary depending on the RAM in use and the size of the machine code program. In this example an unexpanded Vic location 56 would normally hold 30 and 55 would hold 0. For every 256 bytes of machine code you can reduce the value in 56 by 1.

Line 3 lowers the top of usable RAM, performs a CLR to reset the Top-of-Strings pointer and then loads the machine code into the top of RAM. After completion of the load, the program is automatically re-run, hence the need for line 1.

To use this Hybrid type:

```
LOAD"BASIC",1 <return>
RUN           <return>
```

It is always a good idea to put a message on the screen to let the user now what is happening while machine code is being loaded. e.g.

```
2 PRINT"LOADING MACHINE CODE" : PRINT"PLEASE BE PATIENT"
```

Method 4). Small machine code programs can be contained in REM statements at the start of the program. This requires some familiarity with the operation of the Vic and has two main limitations:

i) the routine must be less than 88 bytes long to fit.

ii) There must be no '0' bytes in the routine as the Vic treats these as end-of-line markers.

(e.g. LDA #$00 is not allowed). If you desire a zero it must be produced some other way, (e.g LDY #$FF :DEY:TYA is equivalent to LDA #$00 at higher byte count and destroys Y register).

Method 5). This is the 'BEST' method and involves appending the machine code to the end of BASIC. This should only be done when both BASIC and Machine code have been written and are working properly.

Find the end of BASIC by peeking 45 and 46 ($2D-$2E) and relocate the machine code so that it starts at this address. Now using VICMON (don't forget to enable a VZP) save the block of memory from the start of BASIC to the end of your Machine Code+1 (you will need to calculate the end of Machine Code yourself). You should keep separate backup copies of the BASIC and Machine Code as you cannot edit the BASIC program without destroying the machine code (a useful protection tip).

C64 and 1515 Printer.

Under certain conditions, particularly when printing long lines, the 1515 printer may hang at the beginning of a new line. This may be overcome by switching the screen off before the start of a printer routine with POKE 53265,11 and switching it back on at the end with POKE 53265,27 or whenever screen activity is required. This 'Hangup' does not occur when doing a listing.

[Ed's note: The 1526 printer also has a bug (what hasn't?). In use, switch the printer on last, even then, it can still prevent the C-64 loading from a 1541 disk drive. The fault is a timing problem and does not occur on the Vic-20. Commodore will probably issue a new ROM to correct this one.].

Using the IEEE Card.

The C-64 IEEE card is soft-loaded from ROM when power is applied. It locates itself at $C800-$CFFF the cartridge

then checks for the presence of another cartridge (Z80, Easyscript, etc.) and executes this cartridge in the normal way.

The IEEE software contains a self relocator code which can be used to shift the code if an application demands access to $C800 by using the following assembler code:

```
 ieebse = $c800            ;start of ieee code
 ;
 move  lda #<offset     ;load A with lo byte of addr offset
       ldx #>offset     ;load X with hi byte of addr offset
       sei              ;turn off interrupts
       jsr ieebse+$39   ;move routine always at  ieebse+$39
       cli              ;restore interrupts
       rts
```

Software must not use the RAM/test restore vectors or modify vectors in page 3, if they are to work with the IEEE routines.

Software can check for the presence of the IEEE cartridge by checking that location $0258 contains a $49.

Providing all software uses the vectored i/o routines then all IEEE i/o should be transparent.

C-64 and RS232 operation.

The 64 user port can be used as an RS232 interface with the addition of the VIC1011A RS232C level converter. The line protocol is set up as per charts on pages 352-353 of the C-64 Programmer's Reference Guide.

When in multi-line mode the DSR signal must be true for the transmitter to operate, loss of this line results in a fatal error. If a permanent high is not available from the printer DSR can be connected to RTS which is held high.

Before closing an RS232 file it is essential that the buffer is empty or its contents will be lost. If however CTS is lost during buffer emptying transmission will stop, therefore before closing a check must be made to determine if the buffers are empty and if not transmission must be restarted.

RS232 channel OPEN/CLOSEs perform an automatic CLR which causes BASIC to lose all variables. Therefore from BASIC RS232 OPEN must be performed at the start of the program, CLOSE at the end.

Example:

```
 10 open2,2,0,chr$(8)+chr$(17)
 20 for x=1 to 30: i$=str$(x)
 30 print#2,"The quick brown fox jumps over the lazy dog"i$
 40 next
 50 if (peek(669)<>peek(670)) and (st and 143)=0 then
                    sys(61480):goto 50
 60 close2
```

T.C.

--o0o--

HEX DUMPS TO DISK WITH SUPERSCRIPT

By Simon Tranmer.

In response to an enquiry, here is how to send a hex dump of memory to disk as a file that can be read from Superscript:

OPEN 8,8,8,"0:filename,S,W":CMD8 <cr>

The third '8' is the secondary address. The '0' could of course be drive 1 if preferred. Now go into monitor, but not with the usual SYS4, which cancels the CMD.

SYS 54386 (BASIC4) - Kevin Viney tells me that you can use 'MON' if running PLUSDOS

sent to the CMD device, usually the screen by default, but now the disk.

.X to leave monitor.

PRINT#8:CLOSE8 will close the file.

When loading the file into Superscript, treat it as an ASCII file, i.e. press <rvs> when asked for the file name, before giving the name that you used before.

--o0o--

CHARACTER SET CONVERTER

By David Viner.

The original 2001-8 PET has got a character generator that, in lower case mode, works in opposite sense to all the later models, i.e. you have to shift to get the lower case letters. As the chip used in these old PETs is usually a 6540 it is hard to replace it with an EPROM as the 6540 pin-out is incompatible. There has been a design for a converter board published (Commodore club news - November 1981) but this then leaves the user with the problem of converting all his old programs.

For those who wish to have a character generator that can handle both old and newer styles, I have devised a small add-on circuit that allows change over at the flick of a switch.

Figure 1 shows the full circuit which consists of two cheap LS-TTL ICs (about 12p each), one resistor and one capacitor. The circuit can be easily built on a small piece of Vero-board and mounted inside the PET.

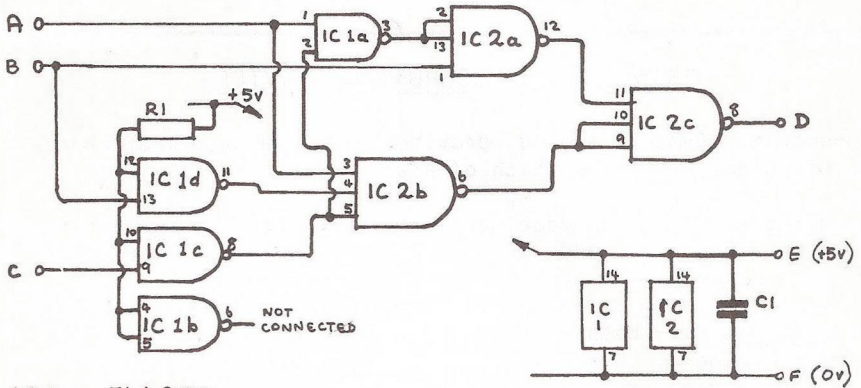
The only alteration to the PET is to cut the pcb track going to pin 11 of the character generator. Figure 2 gives details on how to wire the board up and add the switch.

The circuit works by looking for the sequence on the logic lines to the 6540 that mean the chip has been switched to lower case AND is printing the letters A to Z. This can be done quite easily as the letters are all in one block. Unfortunately there are six other characters in the block with them. These are '@ [ ] ↑ ← \' which means they also get switched over and show the shifted equivalent on the screen. As they are not commonly used in text I do not find this too much trouble.

The circuit has been in use for about 18 months on my own PET without causing any trouble and has certainly saved me from having to convert any of my programs.

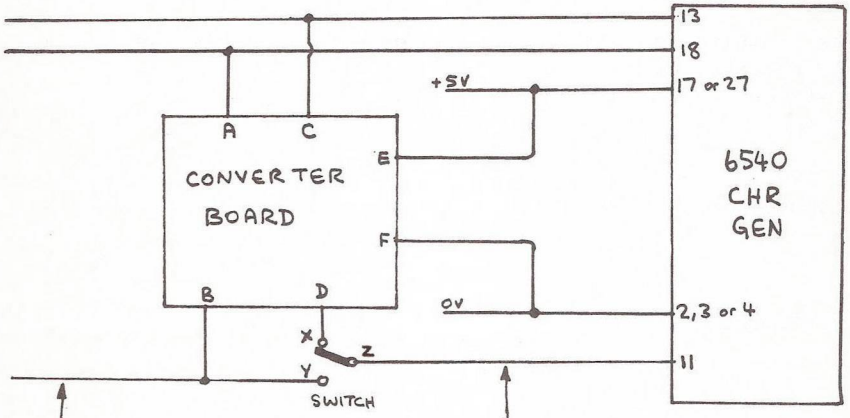
--o0o--

FIG 1.



IC 1 = 74LS00
IC 2 = 74LS10
R1 = 1KΩ
C1 = 0·1μF

FIG 2.



TRACK TO PIN 11 OF 6540 IS CUT AND NOW GOES TO 'B' ON CONVERTER BOARD AND ONE SIDE OF THE SWITCH.

PART OF TRACK STILL CONNECTED TO PIN 11 OF 6540 IS NOW ONLY CONNECTED TO THE SWITCH. WHEN SWITCH IS IN POSITION 'Y' THEN NORMAL 'OLD ROM' IS AVAILABLE. WHEN IN POSITION 'X' THEN NEW ALTERNATIVE CHARACTER SET IS ACTIVE.

THE 6540 CHIP IS LOCATED NEAR THE USER PORT CONNECTOR.

Compuprint Computers Ltd

PRINTING AND COMPUTING SERVICES

Compuprint Computers Ltd provide a unique combination of printing and computing which offers;

Wordprocessing - producing letters, reports, booklets, brochures and price lists at very competitive prices.

Mailing Lists - both clients closed lists and open lists for general use are processed. Extensive sorting and coding makes the system ideal for directories, indexes etc as well as address work.

Bureau Services - for accounting, payroll etc.

Software - special packages for Commodore equipment.

Printing - full range of general commercial work inluding full typesetting facilities. Let us quote for your,- leaflets, letterheads, brochures, booklets etc.

The combination of the above give a complete service to advertisers, publishers of all sizes. We can set, print and distribute a wide range of material. Ready processed work on Commodore discs is accepted.

If any of these services interest you then please contact Grahame Worth on 091 488 8936

---oOo---

Continuous printed stationery and labels, price on request

---oOo---

SPECIAL OFFER TO ICPUG READERS

Boxed sets of personal stationery, normal selling price £7.95, available for limited period for £6.00 inc P+P

Each set includes 40 sheets printed in italics, 40 plain sheets and 40 envelopes, all made from finest quality laid paper. Available in Cream, White or Blue.

Compuprint Computers Ltd

COMPUTER SUPPLIES

All our supplies are available in small quantities if required. The prices shown below includes VAT. Postage will be charged by weight, add up the weights of your selected items, postage and packing is £1.50 plus 25p per kilo.

Orders are accepted on a cash with order basis only. We regret we are unable to accept credit cards.

	Min order quantity	Price	Weight
Discs top brand names.			
Single side S/D for 3000/4000	5	14.35	0.1
Single side D/D for 8050	5	17.25	0.1
Double side D/D for 8250	5	20.00	0.1
Cassettes (C12)	5	3.00	0.1
Ribbons			
Fabric type for 3022	each	3.00	0.1
4022	"	8.28	0.1
8023	"	6.90	0.1
8024	"	7.24	0.1
Single Strike 8026/7	"	3.95	0.2
Paper			
Plain/Ruled			
1 part 9.5" x 11"	500	4.95	1.9
2 part 9.5" x 11"	500	9.90	3.8
Labels (Nominal sizes)			
3.5" x 2" 1 wide	1000	4.00	0.9
3.5" x 1.5" 3 wide	1000	3.45	0.7
2.75" x 1.5" 2 wide	1000	1.65	0.2
2.75" x 2" 3 wide	1000	2.90	0.4
other sizes/widths available on request.			
Plain A4 paper (80g)	500	3.75	2.5
Binders-Plastic for 11" x 9.5	each	2.70	0.1
Card for 11" x 14"	each	1.60	0.1
Program pads (50 sheets)	each	1.75	0.25

COMPUPRINT COMPUTERS LIMITED,
4 SANDS ROAD, SWALWELL, TYNE AND WEAR, NE16 3DJ
TELEPHONE 091 488 8936

MACHINE CODE INTERFACE TO BASIC.

By Tom Cranstoun.

A question people have always asked is how do you get a value from BASIC into machine code programs? There are three main ways depending on the complexity of the number.

1) easiest of all is the POKE method. You poke the number into the location where you want it: e.g. POKE PLACE,A puts A wherever you want it, but A must be between 0 & 255 inclusive.

2) use the USR function: e.g. X=USR(A) This will accept any number, but puts it in floating point notation in what is known as the floating point accumulator #1. This is a horrible format for a beginner to understand, but it can be converted into something reasonable using the routine BASIC uses to get a poke address - $D6D2/BASIC2, $C92D/BASIC4, $B7F7/C64, $D7F7/Vic-20 which will take the floating point number and convert it into a positive integer in location LINNUM ($11+$12 for PETs [BASIC 2 & 4], $14+$15 for Vics/C-64). This number is stored at LINNUM in the usual 6502 lo byte, followed by high byte fashion. The number must be in the range 0 & 65535, inclusive, which is a greater range than poking it. The USR method only allows reading in one number per machine code routine.

3) use BASIC routines directly from your machine code, creating your own syntax for reading numbers, and you can read as many as you want.

The routines are very easy to use, say you wanted a 'print at' type of function where you wanted to pass X,Y co-ordinates to position the cursor on the screen you would want to do something like:

```
SYS826,X,Y
```

You could use the routines used by BASIC. Your machine code would look like:

```
CHKCOM = $CDF8 ;BASIC2,  $BEF5 for BASIC4,  $AEFD for C64,
                         $CEFD for Vic-20
```

```
            ;CHKCOM checks for a comma as the next character
             from BASIC.
;
EVLCHK = $CC8B ;BASIC2,  $BD84 for BASIC4,  $AD8A for C-64,
                         $CD8A for Vic-20
;
;EVLCHK evaluates a number from BASIC, throwing out strings
;
INTEGR = $D6D2 ;BASIC2,  $C92D for BASIC4,  $B7F7 for C-64,
                         $D7F7 for Vic-20
;INTEGR  converts floating  point  number  in  FPAC#1  into
         integer in LINNUM
;
    LINNUM = $11 ;PETS
                 ;Vic/64 =$14
    ;
    *=826        ;PET's 2nd cassette buffer
                 ;64 use $C000, Vic depends on Ramsize.
                 ;
ENTER JSR CHKCOM ;Check for a comma in BASIC
      JSR EVLCHK ;Evaluate expression, typecheck
      JSR INTEGR ;Integer at LINNUM
      LDA LINNUM ;get lo byte only
      PHA        ;save it
      JSR CHKCOM ;again
      JSR EVLCHK ;evaluate second number
      JSR INTEGR ;integer value at LINNUM
      PLA        ;restore old value
      TAX        ;put in X register
      LDY LINNUM ;Y register contains Y value
;
;You would now perform the x,y cursor placing routine.
;
;for 64/Vic users this is a kernal function
;
;Vic/64
;******
      SEC
      JMP SETXY
```

```
;PET users don't have it so easy..
;perhaps easy thing is cursor controls??
;
;PET
;***
      LDA #$13  ;home character
      JSR $FFD2 ; print it
                ;
                ;now do X position
      LDA #$1D  ;cursor right character
LOOPX JSR $FFD2 ;print cursor right
      DEX       ;1 less
      BPL LOOPX ;more to be done
;
;all X done now Y
;
      LDA #$11  ;cursor down character
LOOPY JSR $FFD2 ; print it
      DEY
      BPL LOOPY ;more to be done
      RTS       ;finished
;
```

Assembling the above for a BASIC4 PET and turning into a simple hybrid program gives us a PRINT AT function.

```
100 GOTO 190
110 PRINT AT ON PET
130 BY TOM CRANSTOUN
150 (c) ICPUG 1983
190 PRINT"<clr>";:GOSUB 300
200 SYS826,10+(RND(1)*20),5+(RND(1)*10):PRINT CHR$(166);
    :GOTO200
210 :
220 REM***DATA FOR MACHINE CODE***
230 DATA 826,32,245,190,32,132,189,32
240 DATA 45,201,165,17,72,32,245
250 DATA 190,32,132,189,32,45,201
260 DATA 104,170,164,17,169,19,32
270 DATA 210,255,169,29,32,210,255
280 DATA 202,16,250,169,17,32,210
290 DATA 255,136,16,250,96,-1
```

```
300 REM
310 READ SR:
320 READ V:IF V=-1 THEN RETURN
330 POKE SR,V
340 SR=SR+1:GOTO 320
```

Line 200 produces a block of chequered signs randomly being built up in the middle of the screen. The Graphics character is shift-& on a PET.

--o0o--

6502 ADDRESS MODE PROBLEM

Some people encounter problems with certain peripheral chips when POKEd with apparently correct values. The reason is not particularly obvious, but may arise as follows. Some i/o devices require that their registers be written to in a predefined order before being read; others have differing actions for read and write (e.g. 6850 or 6551 ACIA chips). Now on the 6502 the Indexed and Indirect addressing modes compute an effective address which appears on the address bus a full clock cycle before the R/W or any derived signal. Thus any intended memory write with either of these addressing modes has a read of the location beforehand. This can result in modifying status or interrupt flag bits with potential problems arising. It is not obvious that POKE addr,byte does just that.

One way to circumvent the problem is to use a small machine-code routine at address M thus:

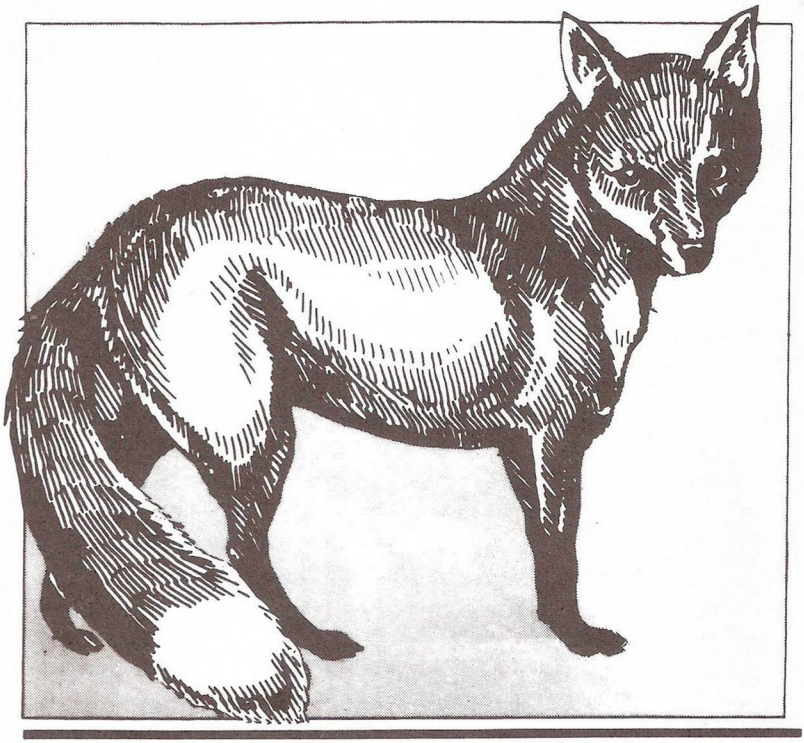
```
POKE M+1,B
POKE M+3,A-INT(A/256)*256
POKE M+4,INT(A/256)
 SYS M
The machine code is M LDA #00
                      STA 0000
                      RTS
```

R.D.G.

--o0o--

FOX ELECTRONICS

FOX ELECTRONICS



FOX ELECTRONICS

FOX ELECTRONICS 141, Abbey Road, Basingstoke, Hants. RG21 9ED

TEL: BASINGSTOKE 20671 (AFTER 6 P.M. - TIM OR JOAN)

ALL PRODUCTS ARE INCLUSIVE OF VAT. OVERSEAS CUSTOMERS PLEASE ADD £2.50 POST AND PACKING.

FULLY GUARANTEED FOR ONE YEAR.

Deliveries 10 days from receipt of order.

FOX ELECTRONICS

FOX ELECTRONICS

Products for the VIC 20

The VIXEN RAM CARTRIDGE.
for the Vic 20
Switchable between 16K or 8K & 3K. Gives you the option of full 16K RAM or 8K and 3K RAM in one package. When added to the standard Vic gives 16384 bytes of extra memory in blocks 1 and 2 or 3092 bytes of extra memory into the 3K memory block AND 8192 bytes switchable between memory blocks 1 and 3. Simply plugs into the rear expansion port and fully compatible with all motherboards and modules available. No re-addressing of existing BASIC programs needed.

£39.95

TANDEM
Expandable Expansion System, gives 4 expansion slots for Vic 20 cartridges. Custom designed case. Plugs directly into computer. Further expanded by using Tandem System ROM socket. No extra power supply needed.

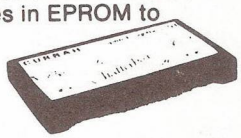
£33.00

VIC LIGHT PEN
A high quality light pen which plugs straight into your Vic with no special interface needed.

£19.50

or for PET 12" Screen **£22.50**

CHATTERBOX. Speech synthesizer with an infinate vocabulary of spoken words out of a number of sound units. Fully programmable and simply plugs into Vic or motherboard. Includes a series of software routines in EPROM to facilitate the programming.



£57.00

Please send me .....

ITEM	QUANTITY	PRICE	TOTAL

Name .....
Address .....
.....

CLUB DISCOUNT - £1.00per item. No P & P

TEXT-WRITER

By D. Viner.

Reading through the November 1982 issue of ICPUG I came across Ron Geere's call for text for publication to be sent in on cassette in a form that can be read on a PET if one doesn't have a word processor. As I didn't fancy typing in all those PRINT#'s I decided to write a program to do it for me. The result, Textwriter, takes text input and stores it in lines at the end of the main part of the program. After all the text has been entered then lines 100 to 980 can be deleted and the result should keep Ron quite happy!!

Line 100 should be typed in exactly as it appears, as the numbers after the REM are used later on and the program expects them to always be in the same place. Do not enter any lines before 100 either. The program writes its text lines at the start of the run (if you press option S to set them up) and the first number (1000) in line 100 is the start of these lines, the second (2000) is the last. The third number (40) sets the line length. All three may be changed though care is needed to ensure that normal program lines are not overwritten.

As the program is written entirely in BASIC the text entry part works a bit on the slow side and HALT will occasionally appear on the screen. This means that the text is being written into a text line and if you type ahead at this point you may lose some characters. One finger typists (myself included) will probably not find this a problem.

The routine that sets up the PRINT# lines depends on certain lines always having the same line number so do not attempt to renumber.

The only editing key usable when entering text is DELETE and this only for the characters entered since the last HALT. To exit back to the menu type SHIFT/CLR. The program does not attempt to right justify the output but never breaks up words so some lines containing long words may appear to be rather short.

Normally text is put into the PRINT# lines from 1000 on but if you want to start at any other line then press option L after pressing T to enter text. The other option B is the normal start.

When printing the text to the screen the space bar must be pressed to enable each page to be displayed. This article was written entirely on Textwriter.

```
100 REM:1000:2000:40:          :START:END:LINE-LENGTH
110 REM LINE 100 MUST ALWAYS BE THE FIRST LINE OF THE
    PROGRAM
120 REM***   TEXTWRITER   ***   *** D. VINER 29.12.82 ***
130 POKE59468,14:GOTO160
140 GETZ$:IFZ$=""THEN140
150 RETURN
160 PRINT"<clr><3dn>"
170 IFP=0THENPRINT"<dn>     S   SET UP PRINT# LINES"
180 PRINT"<dn>     T   ENTER TEXT":
    PRINT"<dn>     P   PRINT "
190 PRINT"<dn>     E   EXIT PROGRAM"
200 GOSUB140:IFZ$="E"THENPRINT"<clr>":END
210 IFZ$="T"THEN570
220 IFZ$="P"THEN740
230 IFZ$<>"S"ORP=1THEN200
240 GOSUB660:A=Q
250 GOSUB660:IFA=RTHENP=1:GOTO160
260 IFB=20THENB=0:PRINT"<clr>"A"IFD=3THENWAIT59410,4,4":
    GOTO290
270 PRINT"<clr>"A"PRINT#1,"CHR$(34)"<40spaces>";
280 PRINTCHR$(34):REM ALTER NO. OF SPACES IN 270 IF 40 IN
     LINE 100 IS CHANGED
290 PRINT"<dn>A="A"+10:B="B"+1:GOTO250"
300 POKE623,19:POKE624,13:POKE625,13:POKE626,13:POKE158,4
    :END
310 K=PEEK(J)+256*PEEK(J+1):J=K:K=J+4
320 F=0:IFPEEK(K)=152THENF=1:K=K+4
330 IFPEEK(J+1)=0THENF=255
340 RETURN
```

```
430
350 J=1025:PRINTT$
360 GOSUB310:IFF=0THEN360
370 IFF=255THEN690
380 PRINT"<clr><2dn>";
390 GOSUB660:A$="":LL=L:C=0
400 POKE167,0:GOSUB140:POKE167,1
410 Z=ASC(Z$):IFZ=13THEN560
420 IFZ=147THEN160
430 IFZ=20THEN530
440 IFZ<32OR(Z>127ANDZ<160)THEN400
450 A$=A$+Z$:PRINTZ$;:C=C+1:IFC<=LTHEN400
460 X=ASC(MID$(A$,LL,1)):IFX<>32THENLL=LL-1:C=C-1:GOTO460
470 POKE32768,136:POKE32769,129:POKE32770,140:
    POKE32771,148
480 C=C-2:FORY=1TOC:POKEK+Y-1,ASC(MID$(A$,Y,1)):NEXT
490 GOSUB310:IFF=0THEN490
500 IFF=255THEN690
510 A$=MID$(A$,C+2):C=LEN(A$):LL=L
520 FORV=0TO3:POKE32768+V,32:NEXT:GOTO400
530 IFLEN(A$)=0THEN400
540 IFLEN(A$)=1THENA$="":C=0:PRINTCHR$(20):GOTO400
550 A$=LEFT$(A$,LEN(A$)-1):C=C-1:PRINTCHR$(20);:GOTO400
560 C=C+2:PRINT:GOTO470
570 PRINT"<clr><rvs> TEXT ENTRY <off>":T$="<clr>PLEASE
    WAIT A FEW SECONDS."
580 PRINT"<3dn>START FROM THE <rvs>B<off>EGINNING OR A
    <rvs>L<off>INE      <dn>NUMBER ?"
590 GOSUB140:IFZ$="B"THEN350
600 IFZ$<>"L"THEN590
610 PRINT"<3dn>START AT LINE ? ";:OPEN9,0:INPUT#9,X$:
    CLOSE9:PRINT
620 N=VAL(X$):IFN<10000RN>10000THEN570
630 J=1025:PRINTT$
640 GOSUB310:IFN<>PEEK(J+2)+PEEK(J+3)*256THEN640
650 GOTO380
660 B$="":FORW=1031TO1034:B$=B$+CHR$(PEEK(W)):NEXT:
    Q=VAL(B$)
670 B$="":FORW=1036TO1039:B$=B$+CHR$(PEEK(W)):NEXT:
    R=VAL(B$)+10
680 B$="":FORW=1041TO1042:B$=B$+CHR$(PEEK(W)):NEXT:
    L=VAL(B$):RETURN
```

```
690 PRINT"<clr><2dn>NO MORE PRINT# LINES LEFT !!"
700 PRINT"<2dn>ALTER THE NUMBERS IN LINE 100 AND RE-RUN
710 PRINT"<dn>THE PROGRAM. ALL OF YOUR TEXT HAS BEEN
720 PRINT"<dn>SAVED SO-FAR (EXCEPT POSSIBLY THE LAST
730 PRINT"<dn>WORD OR SO!).":END
740 PRINT"<clr><3dn>    <rvs>S<off>CREEN OR <rvs>P<off>
    RINTER ?"
750 GOSUB140:IFZ$="S"THEND=3:GOTO780
760 IFZ$="P"THEND=4:GOTO780
770 GOTO750
780 PRINT"<clr>":OPEN1,D:GOTO1000
980 REM DELETE LINES 100 TO 980 AFTER TEXT IS COMPLETE
990 PRINT"<clr>":D=3:OPEN1,D:REM D=4 FOR PRINTER
```

--o0o--

8032 KEY-PRESS DETECTION

By Joe Griffin.

A number of programs, including Tom Cranstoun's 'PET REVAS', use detection of the 'equals' key to control program flow. This is readily achieved in machine code as follows:

```
BIT $E812     ; does not alter A, X or Y
              ;but only alters N, V and Z flags
BMI BRANCH1   ;will therefore test for '=' pressed
BVC BRANCH2   ;test for '.' pressed.
```

On the 8000-series machines (and presumably Fat-40s) these tests do not work because of different keyboard decoding. However, instead of an '=', the keyboard scan looks for a character 22 (not 20 as given in Rae West's book). This is 'DELETE TO END' which can be generated on the 8000 by pressing '←', 'q' and '4' together (use the '4' on the keypad).

Instead of '.', the 8000 looks for character '4' and, unfortunately, this cannot be simulated.

--o0o--

HEX INPUT

By David Viner.

So there you are with the latest issue of PRACTICALLY PERSONAL COMPUTING TODAY containing the new version of SUPEREXTRADISKMONTHINGETC., and all you have to do is type in about ten pages of hex code.... either that or go down the pub! But fear not, here comes HEXINPUT, a utility program that will allow you to type the whole thing in - in about half the time it would normally take.

How is it done, I hear you ask. Well, when you normally type in a hex dump using the machine code monitor you spend quite a lot of the time entering cursor rights/spaces/carriage returns etc. With HEXINPUT all these extra keypresses are unnecessary. Also, a lot of time is wasted when the letters A to F are wanted as you have to move from the numeric keypad to the main keyboard. HEXINPUT gets around this by re-coding the '.-=+*/' keys as A to F. After a while you find you do not even have to look at the keyboard while you are typing.

The screen displays each memory location with its contents seperately and, by using the DEL and HOME keys, it is possible to move backward and forward through memory to correct mistakes or miss entire sections.

Pressing the STOP key returns you to BASIC and the 'M' key will let you into the monitor.

```
100 REM ** HEXINPUT / D.J.VINER 1983 **
110 POKE48,0:POKE49,16:POKE52,0:POKE53,16:POKE59468,12
120 PRINT"<clr><dn>THIS ROUTINE ALLOWS MUCH EASIER INPUT
130 PRINT"<dn>OF MACHINE CODE PROGRAMS FROM HEX DUMPS.
140 PRINT"<dn>USE THE <rvs>DEL<off> KEY TO BACKSPACE OR
    CORRECT
150 PRINT"<dn>ERRORS, THE <rvs>HOME<off> KEY MOVES YOU
    FORWARD,
160 PRINT"<dn>THE <rvs>M<off> DROPS INTO THE MONITOR AND
    THE
```

```
170 PRINT"<dn><rvs>STOP<off> RETURNS YOU BACK TO BASIC.
180 PRINT"<dn>KEYS .-=+*/ HAVE NOW BECOME A TO F SO
190 PRINT"<dn>THAT ALL TYPING IS DONE ON THE NUMERIC
200 PRINT"<dn>KEYBOARD.
210 PRINT"<2dn>PRESS SPACE TO CONTINUE"
220 GETZ$:IFZ$<>" "THEN220
230 PRINT"<clr>THE ROUTINE CAN BE LOADED IN THREE
240 PRINT"<dn>DIFFERENT LOCATIONS :-
250 PRINT"<2dn>A   7906 ($1EE2 HEX)
260 PRINT"<dn>B  16098 ($3EE2 HEX)
270 PRINT"<dn>C  32482 ($7EE2 HEX)
280 PRINT"<4dn>PRESS A,B OR C TO LOAD ROUTINE"
290 GETZ$:IFZ$=""THEN290
300 IFZ$="A"THENJ=7906:GOTO340
310 IFZ$="B"THENJ=16098:GOTO340
320 IFZ$="C"THENJ=32482:GOTO340
330 GOTO290
340 K=J+22:PRINT"<dn>OK. LOADING MACHINE CODE"
350 READA$:IFA$="A"THEN400
360 IFA$="B"THEN430
370 IFA$="X"THEN460
380 A=VAL(A$)
390 POKEJ,A:J=J+1:GOTO350
400 IFZ$="A"THENA=30:GOTO390
410 IFZ$="B"THENA=62:GOTO390
420 A=126:GOTO390
430 IFZ$="A"THENA=31:GOTO390
440 IFZ$="B"THENA=63:GOTO390
450 A=127:GOTO390
460 PRINT"<2dn>ENTER ROUTINE BY SYS";K
470 DATA169,147,32,210,255,162,24,189,227,B,157,0,128,
    202,16,247
480 DATA169,17,32,210,255,96,169,A,133,53,133,49,169,240,
    133,52,133,48
490 DATA32,226,A,32,84,215,165,251,133,1,165,252,133,2,
    32,141
500 DATAB,169,0,133,0,32,225,255,32,228,255,240,251,201,
    77,208
510 DATA1,0,201,20,240,68,201,19,240,58,32,190,B,72,32,
    179
520 DATAB,10,10,10,10,133,96,104,32,210,255,169,1,133,0,
    32
```

```
530 DATA225,255,32,228,255,240,251,201,20,240,31,201,19,
    240,21,32
540 DATA190,B,72,32,179,B,5,96,133,96,104,32,210,255,160,0
550 DATA165,96,145,1,32,123,B,76,18,B,165,0,240,3,76,18
560 DATAB,32,130,B,76,18,B,230,1,208,2,230,2,96,198,1
570 DATA165,1,201,255,208,2,198,2,96,169,13,32,210,255,
    169,17
580 DATA32,210,255,165,1,133,251,165,2,133,252,32,23,215,
    32,46
590 DATA213,160,0,177,1,32,34,215,32,46,213,32,46,213,96,
    201
600 DATA58,8,41,15,40,144,2,105,8,96,201,46,208,2,169,65
610 DATA201,45,208,2,169,66,201,61,208,2,169,67,201,43,
    208,2
620 DATA169,68,201,42,208,2,169,69,201,47,208,2,169,70,96
630 DATA5,14,20,5,18,32,19,20,1,18,20,32,1,4,4,18,5,19,19,
    32,40,8,5,24,41,X
```

--o0o--

REVIEW

PET-Forth - for 8000 Kobra Micro Marketing

This implementation of Forth originates from the Swedish company Datatronic AB. Readers will note that it costs some three times the price of the version available from Supersoft (reviewed p185) and may wonder what one gets for the extra money.

PET-Forth comes on 8050 disk only and is complete with handbook and security ROM at $A000 (UD11). The handbook is A4 size with 322 glossy pages. The text appears to have been prepared on a word processor and has obviously been translated from the Swedish, well at least 99% of it, for in parts the odd trace of untranslated text still remains. It was also felt that the layout of the handbook could have been improved, especially as booting up the system is not covered until chapter 11 on p161.

I have a personal dislike for 'security' ROMs. If you only run one program and only have one ROM then no problem, but if the ROM is paged, then PET-Forth will crash if the correct ROM is not active. The ROM contains 2K of the lower dictionary nucleus words, so that frequent references to

the ROM occur from elsewhere in the dictionary. In normal use, it is totally transparent to the user once installed.

Once booted up the system refers to an electives screen from which are loaded the assembler, some extensions and the editor. Space is provided for the user to add one's own resident application load commands.

The extensions provided include double-precision words, loop indices I', J and K, and a dump utility plus printing in other bases without changing the number BASE.

The library routines provided with PET-Forth include a comprehensive string handling package, a 'tests' structure, a 'case' structure, clock and timing functions, video screen words, random number generator, matrix & vector functions (arrays), CB2 control words, a cross-reference utility and a dump utility.

Screens on the Datatronic disk follow normal Forth conventions and include the author's initials and date at top right on the header line, plus a comment ending in '*)' on definition lines for subsequent retrieval by the cross-reference utility.

The inevitable 'Towers of Hanoi' program is included and it can be quite instructive for the novice to examine the technique.

One problem which was not resolved was that of interfacing to a printer which requires a line-feed after carriage return. The library word PAPER diverts output to device #4, but without line-feed. The obvious way round this was to use the words OPEN and HPOUT (=CMD) with a file number of 128 or more. Unfortunately, obvious though it may be, it did not work and when all else fails - refer to the handbook. It wasn't in there either. Eventually I had to pull the printer apart and reset some internal switches after which all was well.

Good value for money for professional users at £ 179. (£ 139 for ICPUG members). Also available much cheaper as a cartridge version for Vic-20 and C-64.

R.D.G.

--oOo--

Printed and distributed by COMPUPRINT COMPUTERS LTD.
4 Sands Road, Swalwell, Tyne and Wear. Tel (091) 488 8936