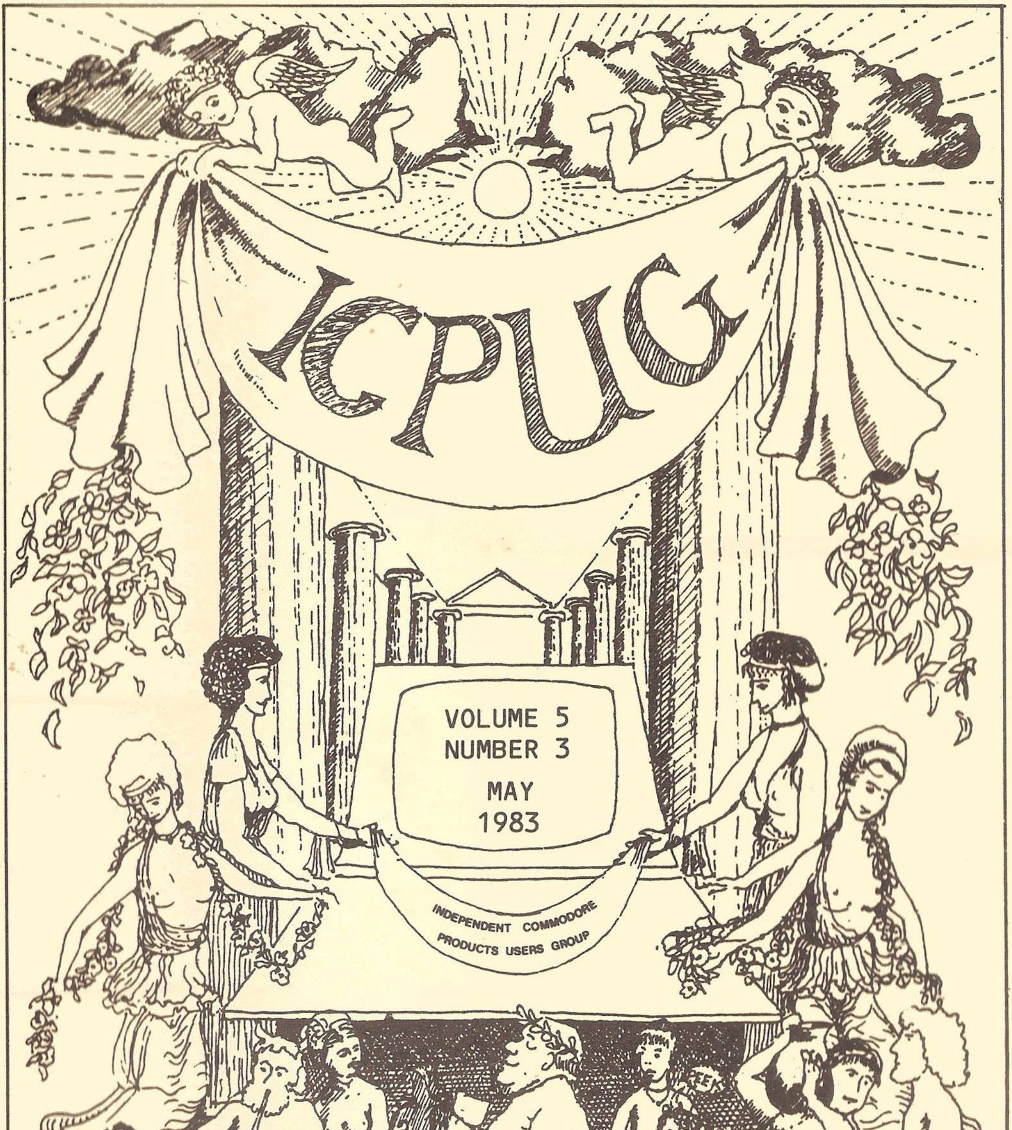

ICPUG
VOLUME 5
NUMBER 3
MAY
1983
INDEPENDENT COMMODORE
PRODUCTS USERS GROUP
POKE

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 3

Newsletter

MAY 1983

Europe's first independent magazine for PET users

Page	Contents

EDITOR'S NOTEBOOK

Commodore appear to have succeeded in changing their image in the U.S.A. due mostly to the success of the Vic-20 and the 64. In the U.K. too, the smaller machines are proving very popular. This popularity has resulted in a number of Commodore specific publications, such as 'The Commodore Gazette', 'Commander' for Vic-20/64/Max (it says) and 'Commodore User'. A number of new magazines are also in the pipeline.

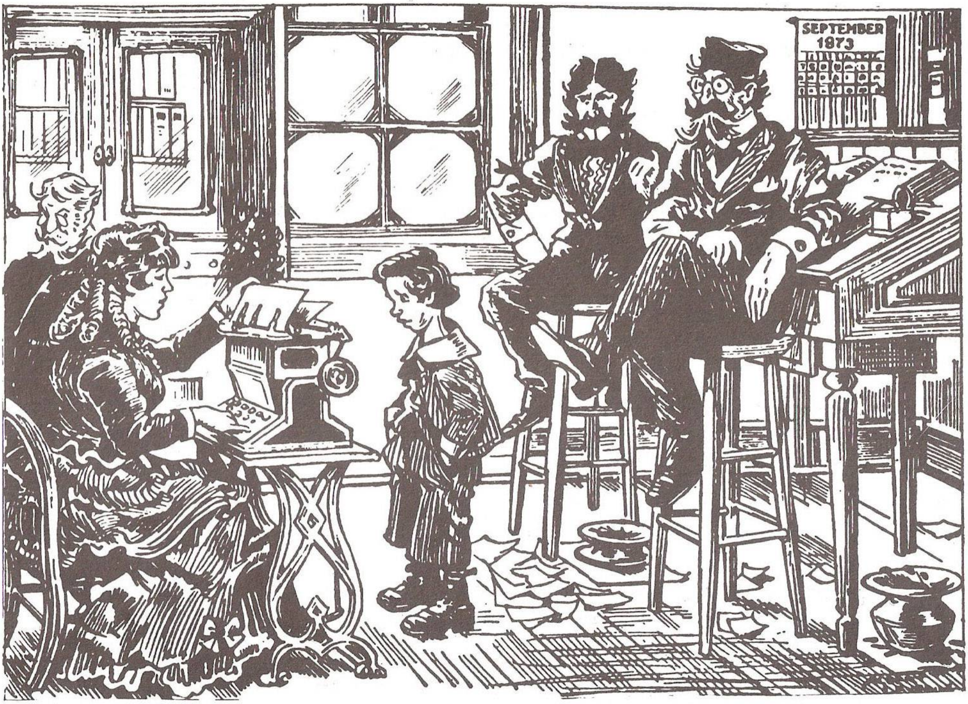
On the business side, Commodore appear to be losing ground. Could it be that 8-bit machines are not fashionable with the media? For word-processing little is gained with a 16-bit machine and the performance margin when the 16-bit price is considered is not that great for a substantial number of business applications. Or could it be the lack of availability of the new machines and information. At the February Commodore 'open day', I was hoping to have a good session with the 500 and 700-series machines, but in many cases the dealer had only seen the computer for the first time that day. They were the lucky ones. One dealer had to cancel his hotel booking because he was too late to get any equipment to demonstrate. Since then the situation has somewhat worsened and by April there were more products unavailable than obtainable.

I had an interesting time as 'honorary salesman', but failed miserably in explaining what one gets for an extra £ 300 difference between the 500 and 700 models other than the loss of colour. After much coffee and some excellent sandwiches, myself and three other engineers came to the same conclusion - insufficient data.

We did find some interesting items, though, such as ?FRE(0) returns zero bytes free which I suppose makes sense with bank switching, and of course the clock functions are different. But with little 'ready-to-run' software we had to write our own off the cuff and get used to yet another keyboard layout!

What we didn't find was that ?FRE(1) gives the number of bytes free in Bank#1, and so on.

Much of the editorial mail is good stuff of a standard acceptable for this magazine, but recently I had the ultimate in junk mail, addressed to me as editor of the User Group and enclosing my 'lucky draw certificate'. I only got away from 'Time-Life Books' by moving house three times.



I sometimes wonder if 'Superscript' would help...

SHOP WINDOW

Trouble with your mains can give rise to all sorts of consequences and even the leaflet warns 'unfiltered power can damage your computer's health'. 'The Plug' is an RFI filter and transient suppressor incorporated in an oversize 13A plug. Priced at £ 12.95 each plus post and packing, total £ 15.50 per plug and available from Power International Limited, 2A, Isambard Brunel Road, Portsmouth, Hampshire, PO1 2DU. Tel: (0705) 756715.

Of course if the power glitch is large enough, like half-an-hour, you may want something more expensive. A Sola Mini UPS (uninterruptable power supply) gives a clean, no-break, no-dip regulated power source from your noisy mains and for up to 24 minutes after your supply has died. Various models are available from Sola, 28, Luake Street, Bedford, MK40 3HU. Tel: (0234) 40094. Alternatively 'The Power Bank' will give similar results from Power Testing (Sales) Ltd., 65a, Shenfield, Brentwood, Essex, CM15 8HA. Tel: Brentwood (0277) 233188.

A range of stepping motor driven, programmable positioning devices controlled via the IEEE-488 bus is available from Bentham Instruments Ltd., 2, Boulton Road, Reading, Berks. RG2 0NH. Tel: (0734) 751355.

I have seen the PET plot, speak, receive data over the 'phone and now it can see! MicroSight1 is a CCTV-based vision system which uses a standard CCTV camera to capture images and a 'Micro Eye' interface to process them into 8-bit digitised video. The system is suitable for text recognition and robots for example. MicroSight comes complete with documentation, listings and explanations. Complementing Microsight1 is MicroScale, an image analysis sofware package at £295. MicroSight1 is £495 - MicroSight2 is £ 2995. Details from Digithurst Ltd., Leaden Hill, Orwell, Royston, Herts., SG8 5QH. Tel: (0223) 208926.

Mator Systems have appeared in these columns a number of times, in this instance mention is made of the Mator Dolphin 3 which is a PET protocol converter costing £ 850. The unit is considerably smaller than the Dolphin 1 or 2 units and is designed to interface between IEEE-488 and RS232/V24 ports using a choice from a number of mainframe protocols. Full details and prices are available from Mator Systems Ltd., 134-140, Church Road, Hove, Sussex, BN3 2DL. Tel: Brighton (0273) 726464.

R.D.G.

--oOo--

COMAL CORNER

By Brian Grainger

COMAL Corner this time brings a mixed bag of news and tips. I shall start off by highlighting some points about string variables. Following severe head scratching by a friend of mine writing some business programs in COMAL for the 8096 it is clear that some explanation is necessary.

Unlike BASIC one must declare the length of a string in COMAL by means of the DIM statement.
e.g. DIM NAME$ OF 12 will allow NAME$ to hold a maximum of 12 characters.

If you try to assign a string longer than this to NAME$ the long string will be truncated to 12 characters and the remaining characters lost. In COMAL, substrings can be referred to simply by specifying the start and end character of the substring.
e.g. NAME$(5:7) refers to the 5th, 6th and 7th characters of NAME$.

When a COMAL string variable is first DIMensioned it has 0 length. When a string value is assigned to the variable the length of the string variable will be the length of the string value, EVEN IF THE STRING VARIABLE HAS BEEN DIMENSIONED TO A HIGHER VALUE.
e.g. NAME$:="FRED" will mean NAME$ has a length of 4 NOT 12.

Once the above points are noted there is a simple rule to remember. Do not refer to substrings which include characters above the length of the current string value.
e.g When NAME$:="FRED" an attempt to set NAME$(6:10):="12345" would result in an error as would PRINT NAME$(6:10).

The above operation of strings is very useful because it means they can be redefined to a new value without the worry of any trailing characters being left. So that if NAME$:="GEORGE" originally and is then set to "ADA" then PRINT NAME$ will result in ADA not ADARGE.

However, what do you do if you wish to refer to a substring when the string has not been assigned a value. The solution is to fill the string with spaces. This can be done by e.g. NAME$(1:12)="". Note that NAME$="" is not sufficient.

Another point to be careful of with strings is when they are INPUT. As with BASIC, the INPUT command takes the input from the screen line, NOT up to the point where the return key was pressed. So if, for example, you have a pretty screen layout which accepts input inside a graphic pattern take care that the string length is not so long that the graphic characters will be input as well. This can lead to some very confusing results!

A last point to note with strings is that when outputting strings to a Commodore printer under FORMATTED output, the print separator must be CHR$(29) NOT ','.

Moving away from strings now I have found another source of problems with people using COMAL is that logical file numbers must be in the range 2-254. COMAL itself uses 1 and 255 so if you use these values strange things may happen.

I have found in using the new versions of COMAL that the DO command is optional in both the one-line and multi-line FOR and WHILE statements. Previously it was only optional in the multi-line structures. As usual the interpreter will add the word automatically.

With regard to the new versions of COMAL to appear, I have heard that the version 2.0 for the new Commodore machines is likely to have a FIND command, DEL Procname facility as well as enhanced entry time error checking. These are in addition to the facilities I mentioned last time.

Finally for this time here is a FUNCtion which will give COMAL a bitwise AND, OR, EOR (Exclusive OR) facility.

```
3000 FUNC BITWISE(OPERATOR$,NO'1,NO'2) CLOSED
3010  RESULT:=0
3020  FOR I:=1 TO 8 DO
3040   BIT'1:=SGN(NO'1 MOD 2↑I - NO'1 MOD 2↑(I-1))
3060   BIT'2:=SGN(NO'2 MOD 2↑I - NO'2 MOD 2↑(I-1))
3070   CASE OPERATOR$ OF
3080   WHEN "AND"
3090    TEMP:=BIT'1*BIT'2
3100   WHEN "OR"
3110    TEMP:=SGN(BIT'1+BIT'2)
3120   WHEN "EOR"
3130    TEMP:=BIT'1<>BIT'2
3140   ENDCASE
3150   RESULT:+TEMP*2↑(I-1)
3160  ENDFOR I
3170  RETURN RESULT
3180 ENDFUNC BITWISE
```

COMAL

An example of use would be:

A=BITWISE("AND",27,59)
which would be equivalent to the BASIC: A=27AND59.

--oOo--

MICRO-APL

Devotees of APL on the 9000-series machines may be interested in a newsletter put out by MicroAPL Ltd. About the same size as this issue, it contains information and articles on APL as implemented on a number of microcomputers, mostly 16-bit jobs with mini-computer price tags. The magazine is circulated to customers "and to others at our discretion" without charge. If you would like to join the mailing list, write to: The Editor 'MicroAPL News', MicroAPL Ltd., 19, Catherine Place, London, SW1E 6DX.

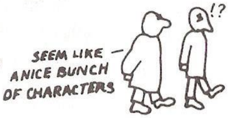
--oOo--

MORE 8032 NOTES

By Paul Barden.

Further to Martin Guy's 8032 notes (Jan issue p74) the following may be of interest:

rvs/esc/k	produces delete line.
rvs/esc/shift/k	produces insert line.
esc/k/z	produces set top.
esc/shift/k/z	produces set bottom.
esc/left shift/k	produces scroll down.
k/c/s	produces scroll up.



A sort of explanation for this can be found by inspecting the keyboard decoding chart on p438 of Raeto West's book. The keys given define three corners of a rectangle, the fourth of which is a normally unused byte which is CBM ASCII for the character produced and has been left lying around in the table for no obvious reason. Where a shift key is used without specifying which one, this is just to add 128 to the ASCII code produced. I've no idea what the mechanism of all this is.

The most useful of the above characters is set top. This can be used for block deletion (to perform DELETE n1-n2) thus:

a) List n1-n2
b) Position cursor immediately after line number n1.
c) Type esc/k/z to set top.
d) Type <clr> to delete the BASIC text leaving the line numbers.
e) Type <home><home> to reset top.
f) Delete any inadvertently produced k's or z's.
g) Use <return> to delete lines as as required.

--oOo--

PHOTO

A new C2N cassette unit has been introduced in Britain having a new sturdy and more visually appealing casing. The price remains unchanged at £45.95 inc VAT.

--oOo--


C2N DATASETTE UNIT
Model 1530
commodore

****

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.

BASIC-PASCAL COMMUNICATION.

By John Palmer.

It would be unfortunate if people were deterred from using the superior facilities of the UCSD Pascal system by the incompatibility of its data files with those of the Commodore Basic system. The programs described here supply a simple solution to this problem (for the 8050 disk drive; the 4040 is presumably similar).

The key to transferring data between the two systems is the rule by which the 512-byte blocks of the UCSD system are mapped onto CBM relative files. This rule is not to be found in the UCSD user manual. However, it is quite simple: the record length of the three CBM files "pascal1", "pascal2" and "pascal3" is 128 bytes. There are thus 4 CBM records to each UCSD block. The UCSD blocks are stored in their natural order in the CBM files, blocks 0-334 being "pascal1", 335-669 in "pascal2" and 670-1004 in "pascal3".

The programs were developed on a CBM8096. The UCSD utility Patch was put to good use for inspecting the contents of files written by both the BASIC and Pascal systems.

INSTRUCTIONS

for the use of the programs 'BASICTOPASCAL' and 'PASCALTOBASIC':

Both of these programs run in the BASIC environment. BASICTOPASCAL takes a CBM sequential file and copies it to a UCSD Datafile, which to Pascal is of type Text (or File of Char). PASCALTOBASIC takes a UCSD Datafile (type Text or File of Char) and copies it to a CBM sequential file. In both cases the original file is on the right-hand drive and the new copy on the left-hand drive of the disk unit.

BASIC TO PASCAL:

Before running the program, first use the Make option of the UCSD Filer to reserve a file of adequate size for the data to be copied. The file must NOT be created with the suffix '.text' as it must be a Datafile to the UCSD Filer (this does not preclude it being of type Text to Pascal).

Next use the Ext-dir option to ascertain the position and size of the reserved file: note the starting block (to the right of the date) and the length in blocks (to the left of the date). Write these down, now terminate the UCSD system.

Load and run 'BASICTOPASCAL'. It will prompt you to insert the files in the drives and to give the name of the BASIC file and the start block and length of the UCSD file. The file will then be copied. On reaching the end of the BASIC file the remainder of the current UCSD block is padded with nulls [chr(0)] and any completely unused blocks will contain rubbish.

When reading the resultant file in Pascal, do not rely on the 'eof' function to detect end-of-data; it will not be true until the end of the last block reserved by the Filer, which may be beyond the end of the valid data. BASICTOPASCAL will not write beyond the end of the reserved file.

PASCAL TO BASIC:

The file to be copied must be a Datafile that has been written by Pascal as type Text or File of Char. The last character written MUST be <etx> [chr(3)]. This is important as BASIC cannot recognise the file-end information kept by the UCSD system.

Use the Ext-dir option of the UCSD Filer to ascertain the start block and length in blocks of the file to be copied. Next terminate the UCSD system, and in the BASIC environment, load and run 'PASCALTOBASIC'. The program will prompt you to insert disks and to give the name of the destination file (which must not already exist) and the start and length of the UCSD file. The file will then be copied. The final <etx> will be omitted.

Both programs display 'End of file' on completion.

```
  2 rem program pascaltobasic;
  4 rem for commodore 8050 disk units;
  6 rem copies an ucsd datafile (file of char to pascal)
    on drive 0 to
  8 rem a basic sequential file on drive 1;
 10 print "Program PascalToBasic"
 12 print "Put UCSD disk in drive 0"
 14 print"Number of first block of UCSD file?"
 16 input b0
 18 if b0<6 or b0>1004 then print"Illegal block":goto 14
 20 print "Number of blocks in UCSD file?"
 22 input b1: b1=b0+b1-1: rem last block number
 24 if b1>1004 then print"Impossibly long":goto 14
 26 sf=1: if b0>334 then sf=2: if b0>669 then sf=3
 28 open 1,8,5,"0:pascal"+right$(str$(sf),1)
 30 if ds>19 then print ds$:print#1:close 1:goto 12
 32 print "Put BASIC disk in drive 1"
 34 print "Name of BASIC file (must not already exist)?"
 36 input f$
 38 open 2,8,6,"1:"+f$+",s,w"
 40 if ds>19 then print ds$:print#2:close 2:goto 32
 41 rem both files open
 42 gosub 100: rem limits of file pascal<sf>
 48 record#1,((b0-bs)*4+1)
 50 bn=b0: en=0: rem blocknumber, eof flag
 51 rem loop
 52 gosub 300: rem transfer a block
 54 if bn=b1 or en=1  then close 2:close 1:print"End of
    file":stop
 56 bn=bn+1
 58 if bn<=bf then 52
 60 rem when end of file pascal<sf>
 62 close 1
 64 sf=sf+1
 66 open 1,8,6,"0:pascal"+right$(str$(sf),1)
 67 gosub 100: rem limits of file pascal<sf>
 68 goto 52
100 rem proc filelimits; extreme blocks of file pascal<sf>
```

```
102 if sf=1 then bs=0: bf=334
104 if sf=2 then bs=335: bf=669
106 if sf=3 then bs=670: bf=1004
108 return
300 rem proc transferblock: must end new file at <etx>
302 ib=0
304 qb$="": rem qb$ is a quarterblock
306 jb=0
310 get#1,a$
312 if a$=chr$(3) then 330: rem expects <etx> to mark eof
314 qb$=qb$+a$
316 jb=jb+1
320 if jb<128 then 310
322 print#2, qb$;
324 ib=ib+1: if ib<4 then 304
326 return
328 rem eof return
330 en=1:print#2,qb$:return
```

```
    ==============================================
  2 rem program basictopascal;
  4 rem for Commodore 8050 disk units;
  6 rem copies a BASIC sequential file on drive 0 to an
    ucsd file on drive 1;
  8 rem the latter is a datafile to the filer, and file
    of char to pascal.
 10 print "Program BasicToPascal"
 12 print "Put BASIC disk in drive 0"
 14 print "Name of Basic file?"
 16 input f$
 18 open 1,8,5,"0:"+f$+",s"
 20 if ds>19 then print ds$:print#1:close 1:goto 12
 21 print "Put UCSD disc in drive 1"
 22 print "Number of first block of UCSD file?"
 24 input b0
 26 if b0<6 or b0>1004 then print"Illegal block":goto 22
 28 print "Number of blocks reserved for UCSD file?"
 30 input b1: b1=b0+b1-1: rem last block
 32 if b1>1004 then print"Impossibly long":goto 22
 34 sf=1: if b0>334 then sf=2: if b0>669 then sf=3
```

```
 36 open 2,8,6,"1:pascal"+right$(str$(sf),1)
 38 if ds>19 then print ds$:print#2:close 2:goto 21
 40 rem both files open
 42 gosub 100: rem limits of file pascal<sf>
 48 record#2,((b0-bs)*4+1)
 50 bn=b0: en=0: rem blocknumber; eof flag
 51 rem loop
 52 gosub 200: rem transfer a block
 53 if en=1 then close 2:close 1:print"End of file":
    stop
 54 bn=bn+1
 56 if bn>b1 then close 2:close 1:print"New file space
    too small":stop
 58 if bn<=bf then 52
 60 rem when end of file pascal<sf>
 62 close 2
 64 sf=sf+1
 66 open 2,8,6,"1:pascal"+right$(str$(sf),1)
 67 gosub 100: rem limits of file pascal<sf>
 68 goto 52
100 rem proc filelimits: extreme blocks of file pascal<sf>
102 if sf=1 then bs=0: bf=334
104 if sf=2 then bs=335: bf=669
106 if sf=3 then bs=670: bf=1004
108 return
200 rem proc transferblock
202 for ib=0 to 3
204 qb$="": rem qb$ is a quarterblock
206 for jb=0 to 127
210 if en=1 then qb$=qb$+chr$(0):goto 220: rem pad with
    nulls
212 get#1, a$
214 if st<>0 then en=1: rem eof flag
216 qb$=qb$+a$
220 next jb
222 print#2, qb$
224 next ib
226 return: rem from transferblock
```

--o0o--

THE VIC COLUMN

By Mike Todd

In the last VIC COLUMN, I briefly mentioned two chess playing programs for the Vic, and David Beasley from Reading has written in with the following review of BOSS, SARGON II and a third, BUG-BYTE CHESS.

BUG-BYTE CHESS

This requires 16K expansion and takes quite a while to load from cassette.

You first set the required level of play (1-9.99) and you have the choice of playing black or white with the board re-orienting itself through 180 degrees if you choose black.

The graphics are adequate, but not as good as the other two programs and, during the game, there is no indication that the computer is thinking, no indication of level of play, no list of previous moves, no clock - no nothing!

The instructions suggest a starting level of play of 2.10, but at this level the game is fairly weak and David (who, by his own admission is not a great player) won fairly quickly and at level 5.55 he still won without too much bother although the game took longer to play.

Each piece flashes just as it is about to move, and again when it has moved and, when a pawn reaches the other side, you are given the choice of promoting it to a Queen, Rook, Knight or Bishop.

Any position can be set up, the level of play can be changed during the game, moves can be listed and if you run out of time, the position can be saved to tape for continuation at a later date.

BOSS CHESS

This requires only 8K of expansion and, on running, it assumes that you want to play white and at level 1, although both of these can be changed immediately if required.

The display is excellent with very good graphics and an indication that the program is "thinking". There's also an indication of level of play, how far the program is looking ahead, the last move made and there's a clock for both players.

Response time is fast at the lower levels, and you get a very good game although it is sometimes difficult to see which pieces have moved because they move so quickly.

You can't set up a game or save a game to tape, but despite that the program is very good.

SARGON II

This is a games cartridge with very good instructions provided. The graphics are good, and you can even change the screen colours.

As well as the keyboard, a joystick can be used to move a cursor around the screen and "pick up" pieces and each piece flashes as it is moved. There's also a list of moves displayed at the side of the board together with an indication that the program is "thinking".

Any position can be set up which can be useful for solving chess problems but there are no facilities for saving a game on tape. It is even possible to ask the program for a suggested move.

All in all, the program is well presented and plays a very good game.

SUMMARY

At £7.95, BUG-BYTE is the cheapest but doesn't offer a very challenging game; but it does have features not available with the others.

SARGON II offers nearly everything that you could want, but is the most expensive of the three at £24.95.

BOSS costs £14.95 and offers the best of everything, the only serious limitation being that you can't set up positions with it. As long as this is not a serious restriction, David suggests that this is the best buy of the three.

He also adds that a fourth game, Audiogenic's GRAND MASTER is also available, although he hasn't tested it. It costs £19.95 and could be worth considering.

THE SOFTWARE TOP-20

Whilst on the subject of games, in the last issue I covered the TOP-20 software products. This in fact was the situation for January 1983.

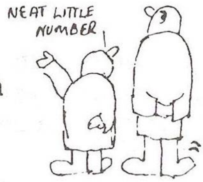
Commodore have now released their TOP-20 for March 1983 and on the following page is a list of these items, with the position in February given in brackets.

It is interesting to see that the SUPER-EXPANDER and PROGRAMMER'S AID now feature in the list and that INTRODUCTION TO BASIC parts 1 & 2 are both still at the top, although SARGON II has now pushed part 2 down to number 3.

It's also curious to wonder why the Adventure games are only represented by ADVENTURELAND, which has dropped from 6th position in January to 19th position in March. I suspect that people are begining to realise that Adventure games have a limited life.

VIC-20 TOP-20

1	(1)	Introduction to BASIC part 1
2	(14)	Sargon II chess
3	(2)	Introduction to BASIC part 2
4	(3)	Blitz
5	(4)	Hoppitt
6	(29)	Race
7	(12)	Strategic Advance
8	(22)	Lighthouse and Subtraction
9	(20)	Know your own IQ
10	(21)	Super-Expander
11	(27)	Money Manager
12	(9)	Omega Race
13	(10)	Gorf
14	(17)	SimpliCalc
15	(--)	GCE/CSE Revision - Physics
16	(--)	Cosmic Cruncher
17	(15)	Gortek and the Microchips
18	(34)	Engine Shed
19	(7)	Adventureland
20	(31)	Programmer's Aid



VIC REVEALED

I don't think that I need remind anyone that the original edition of this book was something of a disaster and that I prepared an error list which ran to 17 pages (still available, free of charge to members who send me a large SAE)

Well, the promised second edition has arrived complete with a sticker on the front cover proclaiming that it is a "NEW EDITION - Corrected and revised". Don't be fooled by this into thinking that all the errors have been corrected or that the presentation has been revised, because it hasn't.

True, some errors have been put right, but in fact only about thirty at a rough count and many of the more serious errors still exist. I will therefore try to modify the error list accordingly to make it suitable for both editions.

If you look at the back cover of the new edition, you will see a quote from me! It says "The only reference guide available - Mike Todd in ICPUG".

I'm sorry to say that I probably did say these exact words at some time - but prefixed by the words "Unfortunately, it is ....."

Duckworth (who have now taken over the publication of the book) were shocked when I contacted them about the implied endorsement and will remove it from the third edition when it appears. They may also incorporate some corrections in this edition!

VIC GRAPHICS

The same author has also produced VIC GRAPHICS, and I have to say that this is certainly better than VIC REVEALED.

But, don't let the title mislead you. It doesn't cover graphics in any more detail than the VIC REVEALED, but it does have many programs to allow you to draw graphcs, patterns and shapes and to scale, stretch, rotate and move shapes drawn on the screen. There's even a section on 3D graphics including the facility to rotate or change the viewing position of an object as well as a hidden line removal program.

These programs are given with specific shapes included and with little description of what the program is doing, but there should be adequate scope for further experimentation.

All the programs in the book are on two cassettes (£7.95 the set) available from the publishers.

One thing that is only mentioned on a sticky label on the back cover is that a SUPER EXPANDER is needed to run the programs in the book, as they use the graphics commands - although there are a couple which don't need it.

I think the book is worth considering, but be warned that to adapt some of the programs to your own applications will need a fair bit of work. At least it's not too bad an introduction to some of the techniques involved.

THE HIGH RES SCREEN

Two questions have "crept" into my mailbag recently regarding high resolution graphics on the Vic: (1) how can the normal high resolution techniques be used on a Vic with 8K or more expansion, bearing in mind the fact that the screen memory moves around; (2) what is the biggest high resolution screen that can be used ?

Taking the second question first, there is a limit to the size of the hi-res screen on the Vic, and this is dictated by the amount of RAM available. The VIC chip can only use internal RAM and not RAM expansion which means that the total RAM available is actually 4K (address 4096-8191).

The normal screen requires a 0.5K block allocated to it, which is normally 7680-8191, leaving a total of 3.5K for the necessary re-definition of the character generator used to create the high resolution graphics.

This is a maximum of 3584 bytes, or 28672 bits - each bit being used to "map" to a point on the screen.

The existing screen is 22x23 or 506 characters - each of which is 8x8 dots and this would require 8x8x506 bits to fully represent in high resolution graphics. In other words, 32384 bits (4048 bytes) But, as already stated, the maximum possible is 28672 bits (3584 bytes) - so you can't even fill the existing "window" with graphics! In fact, the practical maximum is about 21x21 (441) characters, or a total of 28224 dots (3528 bytes).

The SUPER EXPANDER actually uses 20x20 (400) character cells of 8x8 and therefore there are 160x160 dots (25600) which comfortably sit in the available memory space.

Of course, those who have actually played around with the high resolution graphics will know that the character cells used are actually 8x16 dots and the vertical screen size becomes 10 characters, but the number of dots remains the same.

Because of the fact that the screen moves when there is 8K or more of expansion, life becomes a bit tricky if you want to use existing high-res techniques as well as getting the most out of the available memory.

The SUPER EXPANDER handles this by swapping everything back again as soon as the first GRAPHIC command is issued to set high or medium resolution. It also has to move any BASIC program out of the way.

It is not really practical for a BASIC program to move itself out of the way, so the easiest way is to reconfigure the Vic before the BASIC program is loaded.

This couldn't be easier, just execute the following short program and all will be well:

```
20 POKE 648,30
30 POKE 642,32 : SYS64824
```

Line 20 tells the operating system that the screen RAM now starts at $1E00 (7680), line 30 tells the operating system that the lowest available RAM byte is at $2000 (8192) and the SYS command resets BASIC with the new values of screen and RAM pointers.

All this must be done before loading or writing any BASIC program as it completely resets the Vic.

Now, to get high-res graphics, you simply use the same methods as you would with no RAM or with only 3K RAM - but don't use any of the POKEs which you would normally use to protect the BASIC program from being overwritten by the graphics. These would normally be POKE 55 or POKE 56 (or in fact anything between 51-56) and such POKEs should be left out as any BASIC program will be quite safe as it is nowhere near the graphics.

SETTING THE SCREEN AND CHARACTER MEMORY

Those of you with copies of the error list for VIC REVEALED will have seen the tables showing all possible values for the screen and character registers in the VIC chip. For those who haven't, here they are again but pay special attention to the notes given as only a few of the combinations are normally valid.

	Register $9005 bits 76543210	Char generator start address		
*	xxxx0000	$8000	32768	ROM (upper case & graphics)
*	xxxx0001	$8400	33792	ROM (RVS upper cas & graphics)
*	xxxx0010	$8800	34816	ROM (lower case & upper case)
*	xxxx0011	$8C00	35840	ROM (RVS lower case & graphics)
	xxxx0100	$9000	36864	note (3)
	xxxx0101	$9400	37888	note (3)
	xxxx0110	$9800	38912	note (3)
	xxxx0111	$9C00	39936	note (3)
	xxxx1000	$0000	0	note (4)
	xxxx1001	$0400	1024	note (5)
	xxxx1010	$0800	2048	note (5)
	xxxx1011	$0C00	3072	note (5)
*	xxxx1100	$1000	4096	
*	xxxx1101	$1400	5120	
*	xxxx1110	$1800	6144	
*	xxxx1111	$1C00	7168	

Register $9005 bits 76543210	$9002 bit 7	Screen RAM start address		Colour RAM start address		
0000xxxx	0	$8000	32768	$9400	37888	note (6)
0000xxxx	1	$8200	33280	$9600	38400	note (6)
0001xxxx	0	$8400	33792	$9400	37888	note (6)
0001xxxx	1	$8600	34304	$9600	38400	note (6)
0010xxxx	0	$8800	34816	$9400	37888	note (6)
0010xxxx	1	$8A00	35328	$9600	38400	note (6)
0011xxxx	0	$8C00	35840	$9400	37888	note (6)
0011xxxx	1	$8E00	36352	$9600	38400	note (6)
0100xxxx	0	$9000	36864	$9400	37888	note (3)
0100xxxx	1	$9200	37376	$9600	38400	note (3)
0101xxxx	0	$9400	37888	$9400	37888	note (3)
0101xxxx	1	$9600	38400	$9600	38400	note (3)
0110xxxx	0	$9800	38912	$9400	37888	note (3)
0110xxxx	1	$9A00	39424	$9600	38400	note (3)
0111xxxx	0	$9C00	39936	$9400	37888	note (3)
0111xxxx	1	$9E00	40448	$9600	38400	note (3)
1000xxxx	0	$0000	0	$9400	37888	note (4)
1000xxxx	1	$0200	512	$9600	38400	note (4)
1001xxxx	0	$0400	1024	$9400	37888	note (5)
1001xxxx	1	$0600	1536	$9600	38400	note (5)
1010xxxx	0	$0800	2048	$9400	37888	note (5)
1010xxxx	1	$0A00	2560	$9600	38400	note (5)
1011xxxx	0	$0C00	3072	$9400	37888	note (5)
1011xxxx	1	$0E00	3584	$9600	38400	note (5)
* 1100xxxx	0	$1000	4096	$9400	37888	
* 1100xxxx	1	$1200	4608	$9600	38400	
* 1101xxxx	0	$1400	5120	$9400	37888	
* 1101xxxx	1	$1600	5632	$9600	38400	
* 1110xxxx	0	$1800	6144	$9400	37888	
* 1110xxxx	1	$1A00	6656	$9600	38400	
* 1111xxxx	0	$1C00	7168	$9400	37888	
* 1111xxxx	1	$1E00	7680	$9600	38400	

note (1): "x" indicates a bit which is not relevant - unless being changed deliberately, these bits should be preserved

note (2): "*" indicates a practical configuration - all others are given for information only

note (3): Locations $9000-$9FFF contain the I/O chips and therefore cannot be usefully employed for the screen

note (4): Locations $0000-$03FF contain operating system variables and should not be used for the screen

note (5): Addressing restrictions on the memory expansion port prevent the use of any RAM expansion for the screen. Locations $0400-$0FFF cannot therefore be used

note (6): Locations $8000-$8FFF contain the character generator and cannot therefore be used as video RAM

SOME OTHER PRODUCTS

Two Vic products are sitting on my desk - the first is a book called "Beginners Assembly Language Programming for the Vic-20" in the "Dr. Watson Computer Learning Series" and written by Dr. P. Holmes. It is published by Glentop, and is available from Honeyfold Software Ltd., Standfast House, Bath place, High Street Barnet, London (01-441-4130).

I've not yet had a good chance to read right through it, but it appears to be a fairly well put together introdution to programming in assembly language (which, for beginers, is a "user friendly" way of writing in machine code - the code that the microprocessor itself understands).

The book is not quite as logically presented as I would have liked, and the assembly "language" used is different to the industry standard. It uses such mnemonics as LDAZ to indicate a zero page address, and LDAIM for immediate addressing. This may be easier to assemble, but are not the same as those used in the Commodore machine code monitor for instance, and which may cause confusion at a later date.

This criticism aside, the book does handle machine code in as straightforward a way as possible and it even contains the listings for two assemblers (one of which contains a very simple machine code monitor) and there's also a couple of tutorial type programs to help you learn binary and hex notation - both of which are essential if you're going to use machine code seriously.

As well as giving many example programs, methods are shown of incorporating machine code into BASIC programs and there's even reference to some of the arithmetic routines within the ROMs of the Vic.

This sort of programming is not easy, and the book seems to make the path through the jungle reasonably easy going. There are a few places in the book which seem to cloud the issues, but overall, at least on first reading, I would recommend it as an introduction.

64K VIC!

The other item on my desk is a 64K RAM expansion board for the Vic. It costs about a hundred pounds, is distributed by Spectrum and should be available through dealers.

Now, before you get too excited, plugging the SUPER-X2 64K RAM expansion pack into the Vic does not give you 64K of RAM to play with - well not directly anyway!

The RAM is divided into 16x8K blocks and, by means of small switches on the board, these blocks can be positioned to suit most needs. You can configure the 64K RAM to occupy memory blocks 1, 2, 3 & 5 (in other words at addresses $2000, $4000, $6000 and $A000) in quite a variety of ways.

Normally, you would have 2x8K occupying blocks 1 & 2 ($2000-$5FFF) which, at switch on gives you 19967 bytes free. The remaining six blocks can occupy block 3, block 5 or both simultaneously, which would give a maximum of 28159 bytes free on switch on.

So, how do we get 6x8K into 8K of memory space? Well, the answer is called "bank-switching". By POKEing location 39000 with a number from 0-5, we can select any one of the 8K memory banks. This is really of practical use only to those who have the know-how to use it and for most Vic users is of little use although a simple 10-line program is given in the instruction leaflet to allow several BASIC programs to sit in the Vic at any one time, and makes it possible to switch between them as required.

Because it is possible to have RAM in block 5 ($A000) where games cartridges are normally situated, it is (at least theoretically) possible to transfer a games cartridge into the RAM, save it to tape and then load it back again as required.

I have found the board extremely useful in experimenting with the Vic, and with some clever software, I've no doubt that someone, somewhere, could make very great use of this amount of RAM. But, for the novice, it is an expensive way of getting a "full" Vic and I would suggest buying the cheaper 16K or 24K RAM - unless you're prepared to learn how to use the extra RAM. At least the instruction leaflet explains the operation of the board fairly well.

Surprisingly, the board is absolutely tiny! It is smaller than most cartridges that I've come across. Unfortunately, the small bank of switches needed to do some of the configuration is situated next to the edge connector so that, with the board plugged straight into the expansion socket, it is impossible to get at. Even on a mother board, these switches are difficult to get to, although it could be argued that you don't really need to touch them once the board is plugged in. But I do!

Overall, the board is quite good value, but only-if you know how to use it!

--oOo--

DS & DS$ AND THE 'OPEN' COMMAND

By Nigel Richman

When using the CBM disk drive it is advisable to test the error channel to ensure that the command was sucessfully completed. With BASIC 2.0 Commodore recommended the following short routine:-

```
10 OPEN 15,8,15
20 INPUT#15,EN,EM$,ET,ES
30 CLOSE 15
```

This was a very tedious process that could only be run within a program. The CLOSE 15 also closed any open file which then needed to be opened again. This effect could be overcome by opening the command channel at the beginning of the program and closing it again when all disk operations are complete.

The DOS SUPPORT program was a great help in the direct mode but not in program mode. Commodore then released BASIC4.0 with DISK commands including the reserved variables DS and DS$ to access the DISK STATUS in both direct and program modes. Having read the error channel the disk drive resets ~~to~~ 'QO, OK, 00, 00'. BASIC4.0 keeps the values of DS and DS$ since the last disk command. This is to allow repeated tests of DS or DS$ without it restoring to 'OK'. When you ask the operating system for DS or DS$ it tests a byte ($0D) in zero-page. If this is zero then the OS OPENs a channel to the disk drive and reads in the error message. If the byte is not zero (normally $28) then the last message read in is printed or used in the test. When used after a DISK command (such as DOPEN, DCLOSE, SCRATCH etc.) DS and DS$ return the latest error message. However if you are modifying BASIC2.0 programs to take advantage of the new reserved variables the following information must be born in mind.

A typical program using OPEN, rather than DOPEN, to access a file is :-

```
100 INPUT "ENTER FILE NAME";NM$
110 OPEN 2,8,2,"0:"+NM$+",S,R"
120 IF DS=0 THEN 170
130 PRINT "<clr><rvs>DISK ERROR"
140 PRINT "<rvs>" DS$:CLOSE2
150 GET A$:IF A$ ="" THEN 150
160 GOTO 100
170 .........REM PROGRAM CONTINUES
```

Now line 160 loops back so that another file name can be entered. But if the error was '62 FILE NOT FOUND ERROR' DS would remain at 62. In fact if DS had been called earlier in the program then line 120 could easily have failed. This is because the OPEN routine does not reset DS and DS$. (It does of course, reset ST). Therefore to be certain that the DS given to us by the OS is the latest we need to reset it before reading it. This can be done in two ways:

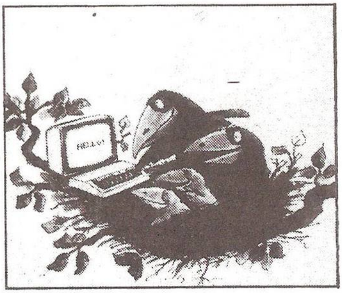
1) Simply POKE 13,0 to reset the string descriptor for DS$. This does not reset the back-pointers at the end of the stored string DS$.

2) SYS 56289 which resets DS, DS$ and ST in addition to the in addition to the back-pointers.

Both methods are effective but the second method is best if harder to remember. The first method may result in GARBAGE COLLECTION problems due to the back-pointers being left.

So the best test of a successfully OPENed (rather than DOPEN) file is:-

```
100 INPUT "ENTER FILE NAME";NM$
110 OPEN 2,8,2,"0:"+NM$+",S,R"
120 SYS 56289: IF DS = 0 THEN 170
130 PRINT "<clr><rvs>DISK ERROR"
140 CLOSE 2:PRINT "<rvs>"DS$
150 GET A$:IF A$="" THEN 150
160 GOTO100
170 continue program
```



So be careful when you use the variables DS and DS$ with the BASIC2.0 OPEN command to disk. It would of course be even better to alter the OPEN commands to DOPEN for all disk commands. Do not access the disk drive with a mixture of OPEN and DOPEN commands. This is because DOPEN allocates a free secondary address to the given logical file number and use of the same secondary address in an OPEN command could cause untold problems.

In case you are not quite sure of the DOPEN command the above program is given in full BASIC 4.0 format.

```
100 INPUT "ENTER FILE NAME";NM$
110 DOPEN#1,(NM$),D0:REM add ',W' for WRITE instead of READ
120 IF DS = 0 THEN 170
130 PRINT "<clr><rvs>DISK ERROR"
140 DCLOSE#1:PRINT "<rvs>"DS$
150 GET A$:IF A$ = "" THEN 150
160 GOTO 100
170 continue program.
```

Have fun with BASIC 4.0 and Commodore's lack of information.

--oOo--

ROUND THE REGIONS

The March meeting of the North Herts Region had the 40/80-column board for the Vic-20 on display. It was quite acceptable in the 40-column mode, but the 80-column display really needed a video monitor.

The Watford Group had the pleasure of Simon Tranmer and Tom Cranstoun demonstrating 'Easy Office', their new database package for the C-64. Following their example the North Hants Region were treated to a similar evening, with Superscript and Superspell thrown in.

--oOo--

C-64 - MACHINE CODE 'OPEN' AND 'CLOSE'

By Brian Grainger

The main purpose of this article is to highlight a potential problem area for those enthusiasts converting machine code programs from a CBM/PET to use on a C-64.

Let me briefly mention one general point about machine code programs on the C-64. It has been mentioned a number of times in the Newsletter that the Vic will automatically load programs to the current start of BASIC memory area. This is equally true of the C-64. If you cannot understand why your machine code routines do not load properly this may be the problem. For example DOS 5.1, the DOS Support program for the C-64 should be loaded into $C000+ but a standard LOAD will load it to $0800+. The solution is identical to that for the Vic. Use a secondary address of 1 with the LOAD command.

e.g. LOAD"DOS 5.1",8,1 if loading from disk.
LOAD"DOS 5.1",1,1 if loading from cassette.

Now to get down to the nitty gritty of this article. When converting some machine code using BASIC4 to use BASIC-64 instead I found some inconsistencies in the C-64 routines for OPEN and CLOSE.

Looking at OPEN first the BASIC4 routine called by the Kernal vector at $FFC0 does three things.

(i) Fetch Parameters (JSR $F50D)
(ii) Loads the accumulator with current logical file number (LDA $D2)
(iii) Performs OPEN

Bearing in mind that the idea of a Kernal vector is to allow machine code to be independent of the machine it runs on what would you expect a call of $FFC0 on the C-64 to do? If you think it will be the same three steps above you are wrong! Step (i) is NOT carried out. It is assumed to be already done. The other 2 steps are OK though. If you wish the equivalent routine to BASIC4 $FFC0 on the C-64 you need to call $E1BE. If you do this step (i), fetch parameters, is done, and then a JMP through $FFC0 is carried out. This will lead to a JMP through a RAM location ($031A) which will load the accumulator with the logical file number ($B8 in C-64) and then perform the OPEN.

If you think the OPEN command is inconsistent just wait till you hear about the CLOSE command. On BASIC4 the procedure is clear.

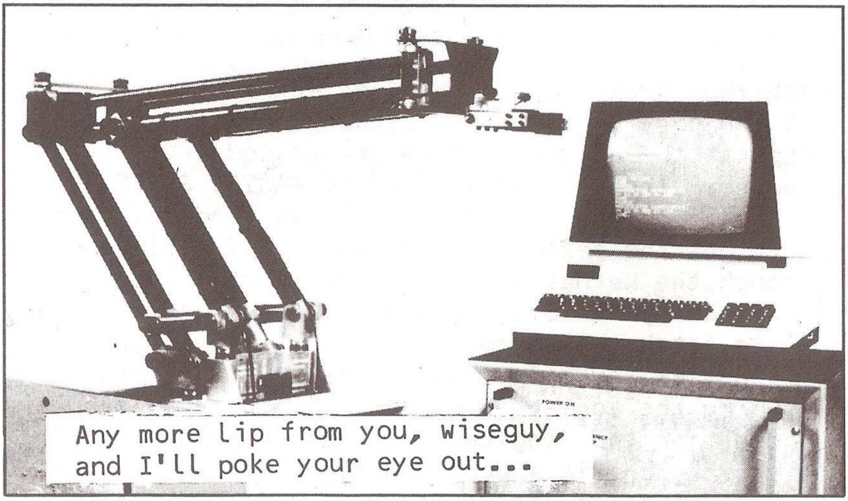
(i) Fetch parameters
(ii) Load accumulator with logical file number
(iii) perform CLOSE

As with the OPEN command the C-64 Kernal vector for CLOSE expects the parameters to be already obtained. However it also expects the accumulator to be loaded with the logical file number! The C-64 equivalent to BASIC4 $FFC3 is $E1C7. If we look at the code from $E1C7 we find a routine is called to get the parameters, then LDA $49 (!!!) and then a JSR through the Kernal vector $FFC3. What is this LDA $49 you ask. Well, I did any way. It so happens that the routine to fetch the parameters as well as storing the file number in $B8, where it should be, will finish with $49 also holding the logical file number. Thus the accumalator is loaded with the logical file number before calling the Kernal vector, but in a very non-standard way. So what, you say ? Well consider the case where some machine code for BASIC4 used to set the parameters manually and then simply called the code for CLOSE skipping the bit that set the parameters (JSR $F2E0). The expected solution for C-64 would be to set the parameters manually and then call JSR $E1CA. However, because of the $49 being used and not the correct $B8, the whole thing gets messed up which entails additional code to store the contents of $B8 in $49 before calling $E1CA. Now for those without a reverse Assembler this extra code is going to cause problems as all the subsequent JMP/JSR references could be incorrect.

Now I fully agree with the concept of Kernal vectors. I could even perhaps accept the fact that the C-64 Kernal vector code is not QUITE the same as the BASIC4 version, IF

IT LEADS TO IMPROVEMENTS. The CBM was only a testbed for the Kernal technique after all. However, picking data from a temporary location rather than its permanent home is just not on and in my opinion is poor programming! My faith in the Kernal vector concept has been severely dented! I do not think much of Commodore's Software Quality Assurance either if such inconsistencies are introduced to what is in effect a developed product (BASIC2 having been with us a long time).

--oOo--



VISICALC - PIRACY & LOOK-UP TABLES

By Brian Grainger

I want to start this article by recounting a little story. When VISICALC first appeared I was asked by a friend to look at making it work without its ROM. I did so and after about a month of work on and off I got a de-ROMed version working. This meant I had got a copy free although I had probably spent my 125 pounds in time although of course it did not go to the originators. Because I felt the program was worth the money and I did not want my version to be freely circulated around ICPUG regions as a freebee I only gave two copies out. Each to well known ICPUG members and both with instructions not to circulate for the reasons given above. I also had made some notes on how the program was deROMed which went with the program copies. On my deROMed version I included my name and former address.

Lo and behold I received a letter in March at that same address requesting some info on VISICALC. The letter, which as far as I am aware came from somebody totally unrelated to where the program copies went, pointed out that a deROMed version was used. I also know that a copy of my notes have been seen in the North of England. There is a lesson to be learned here. With the best will in the world you cannot stop people, no matter how reputable, from distributing illicit copies of programs. The only way to stop it is not to make copies available in the first place. In this case the software writers have probably gained since I know my 'free' version encouraged some additional sales but it does show an aspect of piracy I had not fully appreciated before.

To return to more useful topics, I mentioned in my article on printing the VISICALC formulae (Vol.5, No.1, p.23) that labels in lower case came out strange on the printout. Laurie Faulkner wrote to tell me of the technique he uses with his CBM 4022 printer and the BASMON/PLUSDOS utility chips (available from ICPUG). To display the formulae on the screen he puts the computer into lower case mode (POKE 59468,14) and then does a SEQ"filename". To produce hard copy Laurie sets his printer to upper/lower case mode by:

```
OPEN 7,4,7 : PRINT#7 : CLOSE 7
```

He then spools the VISICALC file to the printer using the PLUSDOS utility 'filename. Apparently this does not cause case problems and he sent me a listing to prove it.

I want to end this piece by discussing the @LOOKUP command. These notes are a combination of two articles I have seen on the subject. One by Mick Ryan in ICPUGSE news and the other by Robert Ramsdell in Byte. I have added some contributions of my own.

The @LOOKUP command, probably the most difficult to understand, enables a user to search a table for the nearest number, NOT GREATER THAN a given number. It then returns a value corresponding to the number found. If the table is given in a row the value returned is given by the entry in the row below opposite the table entry found. If the table is in a column, then the value is in the column to the right of the table.

e.g. If we set a table in column A of the spreadsheet with the values in column B.

	A	B
1	1	1
2	2	2
3	3	9
4	4	16

then @LOOKUP(VALUE,A1.A4) is equivalent to the BASIC:-

```
10 FOR I=1 TO 4
20 IF A(I)>VALUE THEN 40
30 NEXT I
40 RESULT=B(I-1)
```

To give an example of the use of this function suppose we wish to determine what day of the year a given date will be. We could set up the following spreadsheet.

	A	B	C	D	E
1	DAYS	MNTH		DATE	
2	31	1	0	MONTH	5
3	28	2	31	DAY	7
4	31	3	59------------		
5	30	4	90	ANSWR	127
6	31	5	120		
13	31	12	334		

Columns A and B of the above sheet give the number of days (Col.A) in each month (Col.B). Column C gives the number of days in the year immediately before the corresponding month. Thus C3=A2+C2 and C3 can be replicated from C4..C13 using Relative changes for both variables. The month and day of the date are input to E2, E3 respectively. The result will be given in E5. The formulae for E5 will be:-

@LOOKUP(E2,B2.B13)+E3

This works by searching entries B2 to B13 for the nearest value not greater than E2 (5). The nearest value not greater than 5 IS 5 in entry B6. The @LOOKUP thus returns the value of C6 (120) to which E3 (7) is added giving the answer (127).

This particular example could easily be extended to cover the number of days between two dates, (useful for calculating interest payments maybe), or to include years in the date given.

One word of warning when using @LOOKUP. Be aware of the way VISICALC orders its calculations. In the example above it calculates A1,A2..A13, then B1,B2..B13 and so on down each column. Thus to work correctly, the date, month and answer formulae had to be located to the RIGHT of the table columns- in Col.E. If they had been located in Cols A and B it would have given incorrect answers as the results of Column C which are needed for the @LOOKUP function would not be calculated when columns A and B are evaluated.

--oOo--

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.

MATTERS ARISING

In the article on multiple key-press detection (p81 Jan), Brian Grainger points out that the hex to decimal conversion was out by one. References to 256 should read 255 and as a result 251 should read 250.

Colin Pipe suggests that the 'input crash' (p78) can be resolved by typing 'CONT'. The program will continue with variables intact as if nothing has happened. However, I should point out that if one had not noticed the program break out of 'RUN' mode and entered say 20 in response to the anticipated INPUT, this would be seen as an attempt to delete line 20, if any. This would destroy all variables and prevent CONTinue.

One member queried what line 50 did on p35, January issue, well not a lot. Alfred Rose tells me that following the second colon, insert

L=R-256*H:POKE S+2,L:

In addition for BASIC 4, 28,202 in line 60 should be 29,187.

The printer presence check (p174 July and p70 Jan.) still creates interest. Malcolm Friend points out an easier method. The old-ROM program segment following uses an IEEE routine to send an Attention sequence to the bus. Like most solutions it will not work if another device is active on the system. For BASIC2, still use SYS61622, but for BASIC4 use SYS61650. In both cases alter the POKE524,0 to POKE150,0 to clear the status byte.

```
100 SYS61622:IFST=-128THENPRINT"PLEASE SWITCH THE
    PRINTER ON!"
110 IFST=-128THENPOKE524,0:SYS61622:GOTO110
```

The techniques on register exchange presented on pp61 and 215 appear unnecessarily tortuous and a more elegant solution is presented by Paul Barden. A & Y may be simply exchanged by:

```
PHP
STA $0140 ;lowest address of BASIC stack
TYA
LDY $0140
PLP
```

If there is room on the stack for the three PHA's used in Andy Scott's technique, there is certainly room to use the lowest location as a temporary store.

The March issue was larger than most, so it was inevitable that more pages meant more gremlins. At the foot of p180 add the words:
'base are used in the same manner as in standard character mode. The COLOUR MATRIX is used differently.'

The Discounts page (223) went a little odd, but little things can mean a lot. John's request should have read 'Do not telephone UNTIL after 7.30p.m. weekdays. The reference to the 'Compendium' on p238 was misleading, it is not restricted to PET members only (we are all Commodore equipment users), it is still available to all, and in conjunction with the ROM list article on p188, is of use to Vic/64 users also. If you don't have one, there are still some left (£ 2.50 from the Membership Secretary), but don't be surprised if they all go at the Commodore Show.

Finally, with reference to the SYSRES review, the problems that were mentioned on p232 were not due to the SYSRES program but the program that I was using for test purposes. I had unwittingly picked on one of the few I have in BASIC which still use the second cassette buffer and which conflicted with the working space required by SYSRES. Sandra Boden of Solidus informs me that it is possible to use SYSRES on the 8250 disk drive, but the boot disks must have been created on an 8050 drive. Don't forget the members' discount available on SYSRES via John Bickerstaff.

R.D.G.

--oOo--

VICREL - AN INTERFACE FOR THE VIC TO THE OUTSIDE WORLD

Peter J. Pengilly.

I recently purchased a VICREL (Datatronic ab.) add-on for the Vic and thought that my experience could prove useful to ICPUG readers. This device fits to the User Port on the back of the VIC and provides the following facilities:-

1. Six outlet controlled ports max 24 volts, 10 Watts.
2. Two inlet ports.
3. 5-volt supply current.

Connections are made to the device very simply (no soldering) and the 5-volt supply can be used to provide current for the input ports if required.

This means that the Vic can be used for all sorts of control work very easily. Sophisticated burglar alarms can be built, model trains controlled and robot experiments carried out.

If control of mains devices is required this can be done by making up an opto-isolator (a suitable one is marketed by T. K. Electronits of 11, Boston Road, London, W7 3SJ at 2.40 pounds.)

The VICREL was purchased using the Group's discount arrangements and retails at £28.95.

The instructions for programming are very simple and anyone wanting to use the Vic for control is strongly recommended to consider it.

--oOo--

THOUGHT FOR THE MONTH

YEA, from the table of my memory I'll wipe away all trivial fond records. - Hamlet (Act 1, Sc 5 line 98).

--oOo--

VIC/64 CHARACTER SETS

By Mike Todd

There are three ways of getting characters on to the screen of a Vic-20 or CBM-64. You can either print character strings, print the CHR$ values of the characters or POKE the character straight into the screen memory.

There are also two modes of display, the graphics mode and the text mode. The former is how the Vic/64 powers up - capital letters are produced unless the shift key is used, i which case we get graphics characters.

In the text mode, the normal character is the lower case letter and shift produces the capital, just like a typewriter. But, whether you are in text or graphics mode, the characters in the screen memory remain the same, it is purely the character generator that is switched.

Although BASIC normally uses capital letters for keywords and so on, it is the un-shifted letters which must be used - this explains why, in text mode, the capital letters (which are shifted) do not work.

The first two tables that follow show the complete character sets and their screen POKE values - first in graphics mode, and second in text mode. The small numbers under the characters show the actual decimal POKE values, while the column/row numbers could be used to find the hex codes.

The remaining two tables show the same character sets, but as they appear when used with the CHR$ command - thus PRINT CHR$(54) would produce the figure "6".

There are no reverse field equivalents in the CHR$ character set, as these are obtainable using the RVS ON key (CHR$(18)). This character is one of a theoretical maximum of 64 special control characters which, when printed normally, don't generate a character, but instead perform some screen control function.

For instance, CHR$(17) is the same as the cursor down key and PRINT CHR$(17) would have the same effect. All the CBM-64 control characters are listed, including the extra colour codes which the Vic-20 doesn't have (129 and 149-155). When you use GET, and then find the ASC value of a key pressed, you will get the same character values shown which is why the function keys are also listed.

Most of these control codes are self explanatory, although the LOCK and UNLOCK commands may be unfamiliar. They simply lock the computer into its current display mode (graphics or text) such that the user cannot alter them using the CBM/SHIFT key combination, while the program can still alter the mode using the TEXT and GRAF control codes.

When you press a control key on the keyboard, the appropriate control code is generated and the correct action is taken - unless you have typed an odd number of inverted commas. This is to allow the codes to be incorporated into character strings - the so-called "progammed cursor" mode.

To help (?!) identify these codes, the computer generates a reverse field character which, for control codes in columns 0 and 1 are the same characters (but in reverse field) as those in columns 4 and 5. Similarly, for the shifted control characters in columns 8 and 9, they appear as the reverse version of the corresponding characters in columns C and D.

This means that control codes, which can't be generated directly by pressing a key, could be generated in two ways. By using their CHR$ values or by "fooling" the computer into thinking that the character embedded in a string is a control character.

Let's say you're in graphics mode, and want to include a TEXT control character. The simplest way would be to PRINT

CHR$(14), but it could be incorporated in a character string, say A$. First type the command A$="" (note the two quotes!) and then, using the DEL key, delete the second of these quotes. After the first quote the computer goes into programmed cursor mode, the second reverts to direct mode.

Now, select reverse field (RVS-ON) and type the corresponding letter in the 4th column ("N"), cancel the reverse field mode (RVS-OFF) type the closing quotes and press RETURN. Now, the TEXT control character is part of A$, and PRINT A$ should flip the display into TEXT mode!

This technique can be extended to include nearly all control characters but beware trying to use CHR$(0) or CHR$(13) as this could cause problems.

When a program with control codes is LISTed, most will appear in their normal reverse field form, but those which have to be actioned regardless of the quotes mode (such as DEL) will actually be actioned during the LISTing. This means that, by using the above technique of forcing a control character such as DEL (using RVS-"T"), characters on a BASIC line can be deleted during listing. This could be useful for hiding passwords or quiz answers and is worth experimenting with - although, don't place too much reliance on this method of security as it is very easily broken by someone with a bit of inside knowledge.

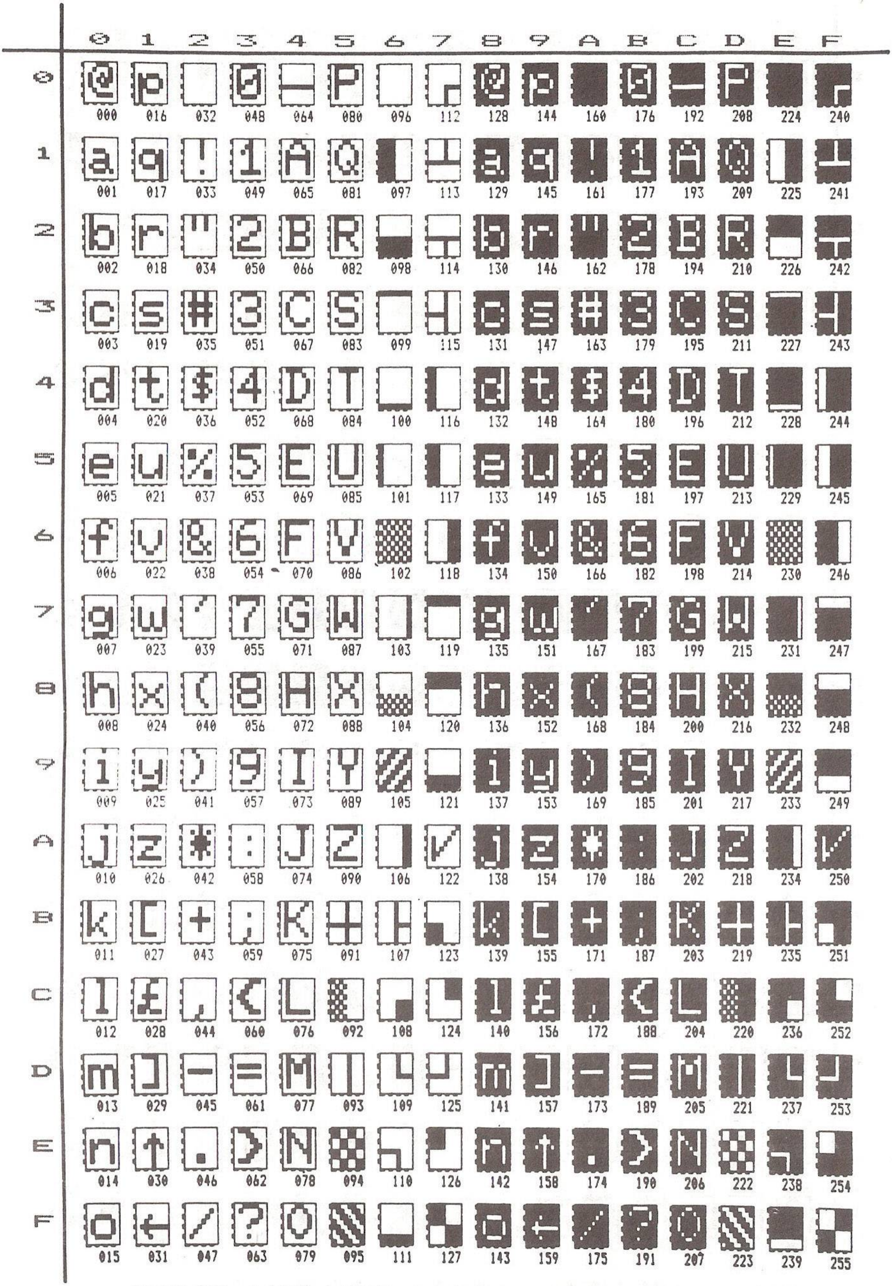
In the CHR$ tables, columns 6 and 7 are a repeat of 2 and 3, similarly E and F are repeats of A and B. This is a quirk of the way the Vic handles characters for output and, although they could be used if required, I would suggest that these repeated character values are avoided.

With the exception of some of the control characters and the pound sign, CHR$(92), these lists also apply equally well to PETs, although I will be producing similar tables for these machines at a later date.

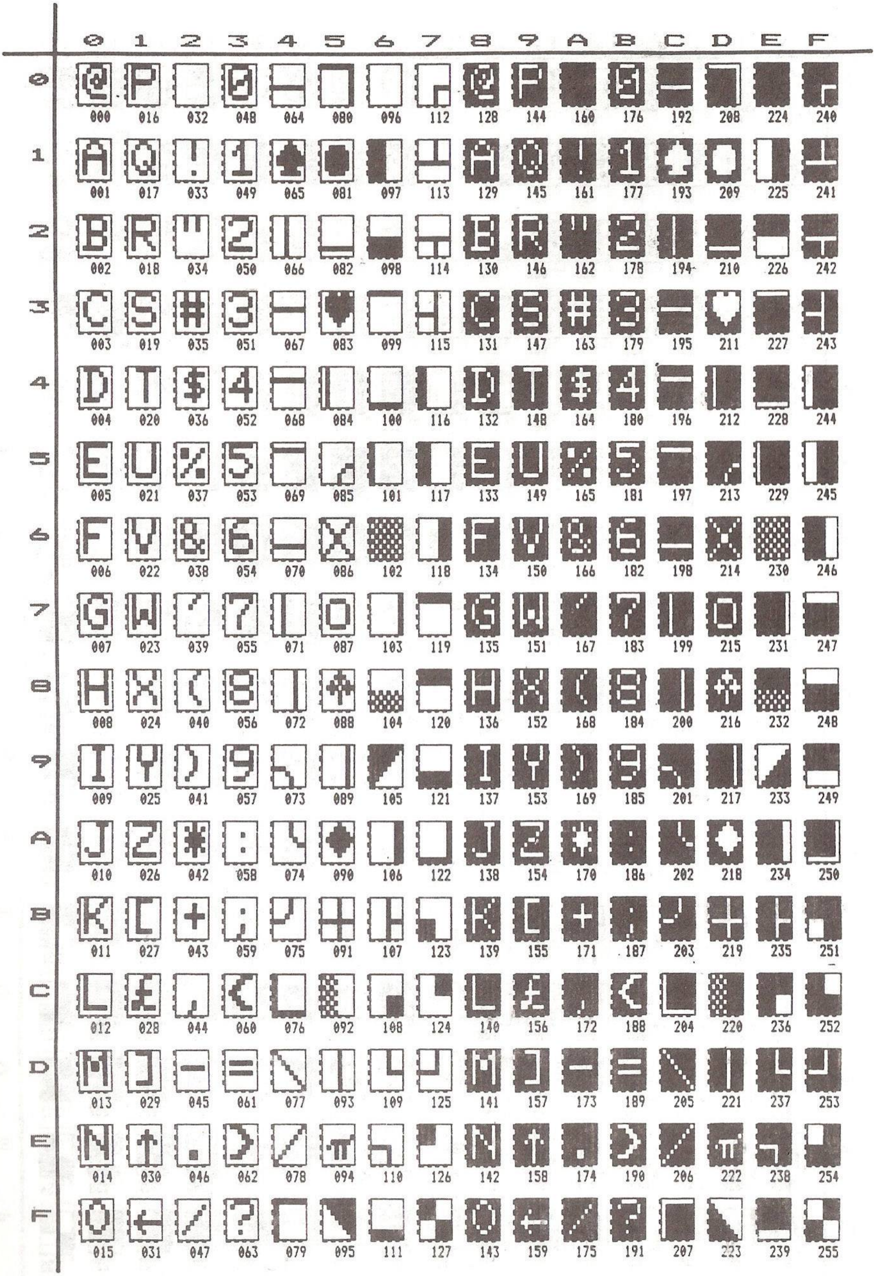
--oOo--

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	000	016	032	0 048	@ 064	P 080	096	0 112	128	BLK 144	160	176	192	208	224	240
1	001	CRSR DOWN 017	! 033	1 049	A 065	Q 081	! 097	1 113	BLK 129	CRSR UP 145	161	177	193	209	225	241
2	002	RVS ON 018	" 034	2 050	B 066	R 082	" 098	2 114	130	RVS OFF 146	162	178	194	210	226	242
3	003	CRSR HOME 019	# 035	3 051	C 067	S 083	# 099	3 115	131	CLR SCRN 147	163	179	195	211	227	243
4	004	DEL 020	$ 036	4 052	D 068	T 084	$ 100	4 116	132	INS 148	164	180	196	212	228	244
5	WHT 005	021	% 037	5 053	E 069	U 085	% 101	5 117	(F1) 133	BRN 149	165	181	197	213	229	245
6	006	022	& 038	6 054	F 070	V 086	& 102	6 118	(F3) 134	LT RED 150	166	182	198	214	230	246
7	007	023	' 039	7 055	G 071	W 087	' 103	7 119	(F5) 135	DARK GREY 151	167	183	199	215	231	247
8	LOCK 008	024	(040	8 056	H 072	X 088	(104	8 120	(F7) 136	MED GREY 152	168	184	200	216	232	248
9	UN LOCK 009	025	) 041	9 057	I 073	Y 089	) 105	9 121	(F2) 137	LT GRN 153	169	185	201	217	233	249
A	010	026	* 042	: 058	J 074	Z 090	* 106	: 122	(F4) 138	LT BLU 154	170	186	202	218	234	250
B	011	027	+ 043	; 059	K 075	[091	+ 107	; 123	(F6) 139	LT GREY 155	171	187	203	219	235	251
C	012	RED 028	, 044	< 060	L 076	£ 092	, 108	< 124	(F8) 140	PURP 156	172	188	204	220	236	252
D	CAR RET 013	CRSR RT 029	- 045	= 061	M 077	] 093	- 109	= 125	SHFT RET 141	CRSR LEFT 157	173	189	205	221	237	253
E	TEXT 014	GRN 030	. 046	> 062	N 078	↑ 094	. 110	> 126	GRAF 142	YEL 158	174	190	206	222	238	254
F	015	BLU 031	/ 047	? 063	O 079	← 095	/ 111	? 127	143	CYAN 159	175	191	207	223	239	255

CHR$ VALUES – GRAPHICS MODE
VIC-20 & CBM-64



POKE VALUES – TEXT MODE
VIC-20 & CBM-64



POKE VALUES – GRAPHICS MODE
VIC-20 & CBM-64

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	000	016	032	0 048	@ 064	p 080	096	0 112	128	BLK 144	160	176	192	P 208	224	240
1	001	CRSR DOWN 017	! 033	1 049	a 065	q 081	! 097	1 113	BLK 129	CRSR UP 145	161	177	A 193	Q 209	225	241
2	002	RVS ON 018	" 034	2 050	b 066	r 082	" 098	2 114	130	RVS OFF 146	162	178	B 194	R 210	226	242
3	003	CRSR HOME 019	# 035	3 051	c 067	s 083	# 099	3 115	131	CLR SCRN 147	163	179	C 195	S 211	227	243
4	004	DEL 020	$ 036	4 052	d 068	t 084	$ 100	4 116	132	INS 148	164	180	D 196	T 212	228	244
5	WHT 005	021	% 037	5 053	e 069	u 085	% 101	5 117	(F1) 133	BRN 149	165	181	E 197	U 213	229	245
6	006	022	& 038	6 054	f 070	v 086	& 102	6 118	(F3) 134	LT RED 150	166	182	F 198	V 214	230	246
7	007	023	' 039	7 055	g 071	w 087	' 103	7 119	(F5) 135	DARK GREY 151	167	183	G 199	W 215	231	247
8	LOCK 008	024	(040	8 056	h 072	x 088	(104	8 120	(F7) 136	MED GREY 152	168	184	H 200	X 216	232	248
9	UN LOCK 009	025	) 041	9 057	i 073	y 089	) 105	9 121	(F2) 137	LT GRN 153	169	185	I 201	Y 217	233	249
A	010	026	* 042	: 058	j 074	z 090	* 106	: 122	(F4) 138	LT BLU 154	170	186	J 202	Z 218	234	250
B	011	027	+ 043	; 059	k 075	[091	+ 107	; 123	(F6) 139	LT GREY 155	171	187	K 203	219	235	251
C	012	RED 028	, 044	< 060	l 076	£ 092	, 108	< 124	(F8) 140	PURP 156	172	188	L 204	220	236	252
D	CAR RET 013	CRSR RT 029	- 045	= 061	m 077	] 093	- 109	= 125	SHFT RET 141	CRSR LEFT 157	173	189	M 205	221	237	253
E	TEXT 014	GRN 030	. 046	> 062	n 078	↑ 094	. 110	> 126	GRAF 142	YEL 158	174	190	N 206	222	238	254
F	015	BLU 031	/ 047	? 063	o 079	← 095	/ 111	? 127	143	CYAN 159	175	191	O 207	223	239	255

CHR$ VALUES - TEXT MODE
VIC-20 & CBM-64

CLUB SOFTWARE NEWS

By Brian Grainger

In this article on club software I want to cover two items. Some programs for numerical analysts and the latest news on the PET REVAS.

Starting with the REVAS, a lot has happened since I reviewed this software last September. To briefly remind you of the purpose of REVAS, it is to create from a machine code binary file, completely source code compatible with the Commodore Assembler. It attaches labels as requested to JSR/JMP instructions and/or Branch instructions and/or 3-byte instructions. The REVAS also came with label tables so that, say, BASIC4 source could be created from a BASIC2 binary file. Well the latest news on this utility is that the JAN'83 version now has label tables for C-64 included. This means that machine code can be translated from any of BASIC2, BASIC4, BASIC-64 to any other. The REVAS itself has been enhanced with a new function (Do the whole lot!) which will attach one set of labels to JMP/JSR references. A different set of labels to branch references and a further set of labels to references of 3-byte op-codes. This greatly simplifies the conversion process combining what used to be six tasks into 1. The instructions to REVAS have also been enhanced. It is also worth mentioning that the enhanced Editor for the Commodore Assembler system is now at version 8 and exists for BASIC2, BASIC4, FAT40 or 8032. All these programs are available from our Assistant Editor, Tom Cranstoun, 107, Dalmally Road, Addiscombe, Croydon.

Those who have read my articles on COMAL will have seen me refer on occasion to Nick Higham. Nick is a mathematician who takes a keen interest in Numerical Analysis. He has recently made available to the software library two programs which will be of particular interest to those with similar interests.

The first called 'Quadrature' will perform numerical integration by a variety of methods. The function is defined as if input into a program line (e.g. f(X)= 2*X+3*X↑2)). The appropriate integration method is chosen, integration limits given and in no time at all the result is displayed. Integration methods covered by the program are Gauss-Legendre, Repeated Simpsons Rule, Gauss-Laguerre and Gauss-Hermite.

The second program from Nick to be added to the software library is to manipulate N by N matrices. Such matrices can be input and edited. The program then allows:-

(i) matrix multiplication to be performed
(ii) solution of Matrix equations of the form Ax=b
(iii) LU factorisation and matrix inversion
(iv) Eigenvector evaluation by the Power method or inverse iteration method.

Both these programs are well written, useful and easy to use. A must for Numerical Analysts or the Maths or engineering faculties of Further Education establishments.

--oOo--

POUND SIGNS WITH 'SIMPLY WRITE' & CBM PRINTER

By Peter J. Pengilly.

I noticed a plea for a method of getting a pound sign with the Simply Write word processor in the computer press recently and send you my solution to the problem.

Insert the following lines of code at the start of the Simply Write program:-

```
DATA 2,18,254,146,194,2
OPEN5,4,5
FOR I=1 TO 6:READ A:A$=A$+CHR$(A):NEXT I
CLOSE 5
```

When you want to use it in the text, use as in the following example:-

```
THE PRICE IS
↑CH=254←
256................←
```

--oOo--

COMMODORE ASSEMBLER EXTENSIONS REVISITED

By Simon Humphrey.

I was interested to read the article on extending the Commodore Assembler (November p325) and since then I have obtained a copy of the program from the ICPUG library. The program as supplied (and as mentioned in the article) is intended for use with the Commodore assembler for BASIC4 (V121579). As I have the assembler for BASIC2 (V112779) I have adjusted the program to work with this assembler. The changes are as follows:

File ASM.REPINPUT

```
        change ISALPH = $1049 to ISALPH = $0FC5
        change BBASIC = $1F8D to BBASIC = $1EF3
        line 268 change JMP $061A to JMP $0619
```

Change the following to be compatible with the standard assembler:

```
        line 411 CMP #'"↑    to    CMP #'↑
        line 420 CMP #'"←    to    CMP #'←
        line 433 CMP #'"\    to    CMP #'\
        line 460 CMP #'"↑    to    CMP #'↑
        line 463 CMP #'"←    to    CMP #'←
        line 466 CMP #'"\    to    CMP #'\
```

Change patching the assembler (instructions at the end of this file).

```
        line 490 change $FFCF to $F1E1 [not essential - Ed].
        line 491 change $1F0E to $1E79
        line 493 change $061A to $0619
        line 494 change $05E2 to $05E1
```

File ASM.EXTNCNTRL

```
        change NXTSTR LDA $0348,Y to NXTSTR LDA $032C,Y
 line 51 change JMP $0F72 TO JMP $0EEE
```

in NAMES add 32 more spaces after .BYTE 'LEA' (there are only 16 spaces in the Library copy.)

in STRCAD change $0F72 to $0EEE

File ASM.IFEXTN

```
        change SYMBOV = $143D   to   SYMBOV = $13B9
        change READCH = $0FA2   to   READCH = $0F1E
        change UNDFSY = $0F72   to   UNDFSY = $0EEE
        change IGNRLN = $0F95   to   IGNRLN = $0F11
        change RANOFF = $0F72   to   RANOFF = $0EEE
        change OUTMES = $063A   to   OUTMES = $0639
```

in comment on line 328 change \$02C0 to \$02C2

In the instructions for patching the assembler, change as follows:

Location	From	To
\$042D	A0 20	A0 28
\$042F	A2 D3	A2 39
\$043D	20 19 06	20 37 22
\$05E1	20 19 06	20 37 22
\$0908	4C EE 0E	4C 2D 23
\$1E79	20 E1 F1	20 B3 22

The instructions on the disk omit to mention how to create the new object file. Save each file (under the original names) after the changes have been made, then with the disk containing the files in drive 1, assemble the source file ASM.ASSEXTNS (this calls the files in the correct order) creating an object file on disk. This new object file can be used in place of the file OBJ.EXTNS to create the extended assembler as described in the instructions on the ICPUG library disk.

--oOo--

It was simply that 152 pages cost too much...

DISK FILE - SECTOR 6

By Mike Todd

Well, I'm afraid this DISK FILE, like the last, is going to have to consist of bits and pieces owing to time and space restrictions!

8250/8050 COMPATIBILITY

There are many differences between the 8050 and 8250. The obvious one is that the 8250 is a double sided system, whereas the 8050 uses only one side of the disk.

The relative file structure on the 8250 is also different and care is required in using them. The main feature is that the 8250 can, at least in theory, use the entire disk to store relative files, whilst the 8050 is limited to 182K.

While it is possible to use disks written on an 8050, there are several precautions that need to be taken. Remember, though, not all disks suitable for use on 8050 drives can be used on the 8250, only those that are high quality, double sided, double density and 77 track.

The first attempt to access an 8050 disk on an 8250 will result in a disk error but future accesses will work correctly. It is not quite so simple when using relative files, where the 8250 must be set up to handle unexpanded relative files as follows:

```
OPEN 15,8,15
PRINT#15, "M-W";CHR$(164)CHR$(67)CHR$(1)CHR$(255)
CLOSE15
```

Copying relative files from 8050 to 8250 is really quite simple. Enter and run the above code to disable expanded relative files, then with the 8050 disk in drive 0 and a formatted 8250 disk in drive 1, copy the relative file in the normal way (using COPY D0,"source" TO D1,"destination").

If you get a disk error on the first attempt (which you will if the disk in drive 0 has not yet been accessed) then simply repeat the command.

Once the copy is complete you will need to reset the 8250 (by switching off and on again) then put the Demo Disk (supplied with the 8250) into drive 0 and LOAD and RUN the program "EXPAND RELATIVE".

There have been reports of difficulties in loading Visicalc and Silicon Office, as well as accessing the directories from Wordcraft.

Commodore recommend that any software existing on 8050 disks should be copied to 8250 format. This works fine with Visicalc and Wordcraft, but Silicon Office is copy protected and users should contact Bristol Software Factory for assistance.

Copying ordinary files from 8050 to 8250 is also straightforward, as long as you remember to access the 8050 disk first to avoid disk errors.

With the original 8050 disk in drive 0 and an 8250 disk in drive 1, format the 8250 disk in the usual way (using HEADER command and, ideally, using the same ID and name as the original disk).

Perform a CATALOG D0 (twice if necessary to avoid the disk error occuring) and simply copy the entire disk using COPY D0 TO D1.

One final difference between the two disk systems is the error code returned in DS$ or through the error channel. On the 8250 it consists of five items instead of the four on the 8050. The format is now:

Error number, message, track, sector, drive

ANTI-COPYING

Over the last year I have had many letters asking how to protect disks against copying and so I thought I'd explain some of the problems involved.

First of all, there are actually only four ways of copying programs from one disk to another.

There's the inbuilt BACKUP or DUPLICATE command which is actually part of the disk operating system and which makes an identical, block-for-block copy of the original.

If during this copy process there are any errors in the original disk, the copy process will abort. Therefore to prevent this type of copying, it is simply a matter of corrupting a section of the disk.

In its most crude form, a magnet can be placed near the outer edge of the disk, erasing sections of the first few tracks. Alternatively, the disk can be formatted omitting the first couple of tracks. This is easily done by starting the formatting process on a second disk, and after the head has stepped through, for instance, two or three tracks, the disk to be formatted is put back in the drive and the formatting will continue.

The second method of copying is to use the COPY command in the DOS which reads the original file character by character and writes it back to the duplicate disk.

The third method involves using the DOS U1 and U2 commands to read and then write every block on the disk. This will also correct any checksum errors in the data and is a useful method of recovering some forms of disk corruption.

The fourth method is to read the program into the computer and then write it back again to a new disk. Of course, any program which runs automatically on loading and which disables the STOP key or takes other precautions to stop you getting at the loaded program will prevent this type of copying.

Copy protection involves a variety of complex techniques to corrupt the disk, to write the program in unusual formats, to encode the program or to place uncopyable "identifiers" on the disk, which the program must constantly check.

Of course, all combinations of these are often used, but the disadvantage of any copy-protection system is that you need to be able to get the program into the computer in a machine readable form. This is most often done by employing a loader program, which itself must be loadable as a normal program and therefore can be "got at" by anyone with only a little experience of disk systems.

The programs themselves will often be "hidden" on the disk so that they don't appear in the disk directory and they will be encoded in some unique way such that they can only be read back using the loader program.

This loader may also keep a check on the disk to make sure any "identifiers" placed on the disk are still there.

Of course, it is possible to modify the loader to remove these checks and to determine the encoding system used and to determine where on the disk the programs are saved and, with this done, the program can be copied with no problem.

Some programs use the "dongle" security device which assumes that a program can be copied, but can only run with this special hardware version of a password. Even so, it is possible to "de-dongle" a program by hunting through the program looking for the "dongle-checks" and removing them. Of course, the smart programmer will also incorporate checks (often in hideously obtuse code) to make sure that the "dongle-checks" are still intact.

I've said it before, it is impossible to fully protect any program from copying. As long as it is necessary to read the program into the computer in executable form then it can be disassembled and examined and the method of protection determined.

Of course, the more sophisticated the protection, the deeper the knowledge and the more time it will take to break. It is this which will deter most copiers.

I might add that I've yet to get hold of a disk that I have not been able to break within a couple of hours - and some of these have used some of the most sophisticated techniques currently available!

FILENAMES et al

When you OPEN, LOAD or SAVE a disk file, the first thing that is sent to the disk drive is the filename and in all three cases it is sent in exactly the same way. The only exception is using OPEN without a filename, when nothing is sent to the disk drive at all.

LOAD and SAVE start off like OPEN, except a specfic secondary address is used - 0 for LOAD and 1 for SAVE.

The first thing that happens with OPEN is that ATN is set to indicate that a command is being sent and a LISTEN command goes out onto the bus. This is in the form 001x xxxx where x is the device number which can theoretically be between 0 and 31 but 0-3 are reserved on Commodore systems and 31 is used as the universal UNLISTEN command.

With the disk drive now listening, the secondary address is sent (still with ATN set) and this is in the form 1111 xxxx where x is the secondary address. The 1111 indicates to the disk drive that this is an OPEN command and that a filename follows.

The ATN line is then cleared and the filename sent, the last character is sent with EOI set. Finally, the universal UNLISTEN command 0011 1111 is sent with ATN set.

Once this has been done, the disk drive performs the necessary housekeeping according to the type of file.

It waits to receive or send data when requested. For instance, a PRINT# command starts by sending the device listen command as before, and sends the secondary address in the form 0110 xxxx where x is the secondary address. This activates the appropriate channel within the disk drive which then procedes to take characters off the bus until the EOI is set (with the last character). The universal UNLISTEN is then sent by the computer.

SAVE works in exactly the same way, except that all the data is sent in one go and the secondary address is 1 which forces the disk drive to default to a PRG,WRITE file.

CLOSE is a matter of sending LISTEN again followed by the secondary address, but this time in the form 1110 xxxx.

INPUT and LOAD work in similar ways but, after the file-name is sent and the disk is ready to send data, the sequence starts with the TALK command 010x xxxx, where x is the secondary address. 0101 1111 is the universal UNTALK command.

The exact action following the OPEN (or indeed the SAVE or LOAD) depends upon what is contained in the filename and the secondary address used.

The filename is parsed to identify the drive number, the filename, the file type and the file mode. If the file mode is omitted, it will default to READ, and if both file mode and file type are omitted, the default is the correct file type for the file being accessed.

If the secondary address is 0 (LOAD) they default to PRG,READ and if 1 (SAVE) to PRG,WRITE.

This means that:

```
SAVE "0:FILENAME,SEQ,WRITE"
```

is a valid instruction and the program file will not be written to disk as a PRG file, but as a SEQ file. This means that it must be loaded back again using:

LOAD "0:FILENAME,SEQ,READ"

In fact, the file type can be SEQ or USR which leads to interesting possibilities in confusing those who read the directory ! But it doesn't work for DEL or REL files.

READING THE DIRECTORY

It is often important for a program to be able to read the disk directory and there are two ways in which this can be done.

First, a file can be OPENed as a PRG file (using the command OPEN 2,8,0,"$0") and then characters fetched from the directory using GET. This will return the directory in its program format, with link addresses and so on and with the file types spelt out.

This can be used to determine the number of blocks free by simply using OPEN 2,8,0,"$0:%&%" since this will return the disk name and ID (again in program format) and then it will list all files called "%&%", of which there should be none. The next line received will be the BLOCKS FREE message, from which the capacity of the disk can be read.

However, it is also possible to OPEN the directory as a SEQ file in the normal way (OPEN 2,8,2,"$0") in which case GET will return single bytes straight from the directory. In fact the first block of bytes will be the BAM, followed by the disk name and then the file entries, complete with their starting pointers.

This has many uses and I will go into these, and offer some useful programs in the next DISK FILE - that's a promise!

THE DISK REVEALED

Just as I was finishing material for the newsletter, I received a copy of DISK REVEALED for review. There's not time for a complete review in this issue - but next time....

So, what is DISK REVEALED? Well, many of you will be relieved to hear that it is not a book written by N*** H********! In fact, it's not a book at all. It's a program to allow you to examine any disk, make modifications to sectors and, to some degree at least, reconstruct a corrupted disk. It is written in machine code by Nigel Richman (an ICPUG member) and is available from SUPERSOFT (01-861-1166) - I'm sorry, but I don't know the price.

The program will work with 2031, 2040, 3040, 4040 and 8050 disk drives, and with BASIC 2 or 4 PETs (40 or 80 column) although there are problems at the moment when using the 2031 (owing to a "quirk" of the 2031), but this is being corrected and all should be well in later versions.

The program allows you to get a hex and ASCII dump onto the screen of the contents of any block on the disk, to modify and re-write the block, "chain" through files, search files for character sequences and to generally "roam" about the data on a disk. There's no doubt that you need a good understanding of the disk's structure to make full use of the program but with such a knowledge the program could be extremely useful.

The program works out which disks and which machine it is being run on and a version which will be able to run on a 64 with 1541 disk drive is being devloped.

It seems to be a very fast and useful tool for those who need to get at the disk and it appears to work without any problems. By the next issue I'll have had a chance to put it through its paces, and I'll give more impressions then.

--oOo--

FORTH COLUMN

By Ron Geere.

Subsequent to my review of 'Forth for PET/CBM' (p185) I have discovered a bug in the DUMP utility. Each iteration of the address loop leaves the address on the stack. If one attempts to DUMP too many addresses the resulting stack overflow causes a system crash.

The fix is to delete one of the two consecutive DUP instructions, however, apart from 16 addresses to a line, the facility as it stands has little value. One could simply enter the monitor with MON and achieve virtually the same effect. If now we insert OVER OVER prior to the second DO loop and

```
    3 SPACES DO I C@ CHR$ LOOP
```

after it and define CHR$ prior to DUMP as follows:

```
 : CHR$ ( b --- )
                 DUP 96 AND IF EMIT ELSE 46 EMIT DROP THEN ;
```

then we not only have a hex dump, but the ASCII equivalents where applicable. An 80-column screen gives better results.

U< bug.

Some implementations of Forth have a bug in the U< function which has its roots in the original Fig-Forth 6502 definition. Frank Chambers sends this fix from fullFORTH+.

```
 CODE U<    2 X LDA,  SEC,  0 X SBC,  3 X LDA,  1 X SBC,
  3 X STY,  CS NOT IF,  INY,  ENDIF,  2 X STY,  POP JMP, C;
```

Forth on a Eurocard.

There may be some interest in the all-CMOS circuit board with built-in Forth on a 6301 microprocessor. This £ 185 Eurocard incorporates 10K of RAM, 16K of EPROM, 10 lines of i/o, a timer and serial interface. It has a full screen editor. The unit, known as the TDS900, comes from Triangle Digital Services Ltd., 23, Campus Road, London, E17 8PG. Tel: 01-520 0442.

Forth & Chips.

I was particularly intrigued to hear of a single-chip microprocessor from Rockwell that contains a Forth operating system in its internal ROM. Holding 128 run-time Forth functions suitable for a dedicated system. This includes kernel level operations such as stack manipulation, run-time portions of control structures, i/o and formatting. An external ROM completes the remainder of the operating system, such as dictionary building operations. This can later be replaced by the application program which is initially developed in RAM. The chips are part of the 6500-series.

Astute Forth devotees will have realised that the CALC RESULT package in the Kobra catalogue mailed with the March Newsletter, was written in Forth by Swedish company Datatronic AB. Next issue I shall be reviewing this version of Forth, along with Huang's book 'And so Forth'.

--oOo--

BASIC 4 LOWER CASE LISTER

By Jim MacBrayne.

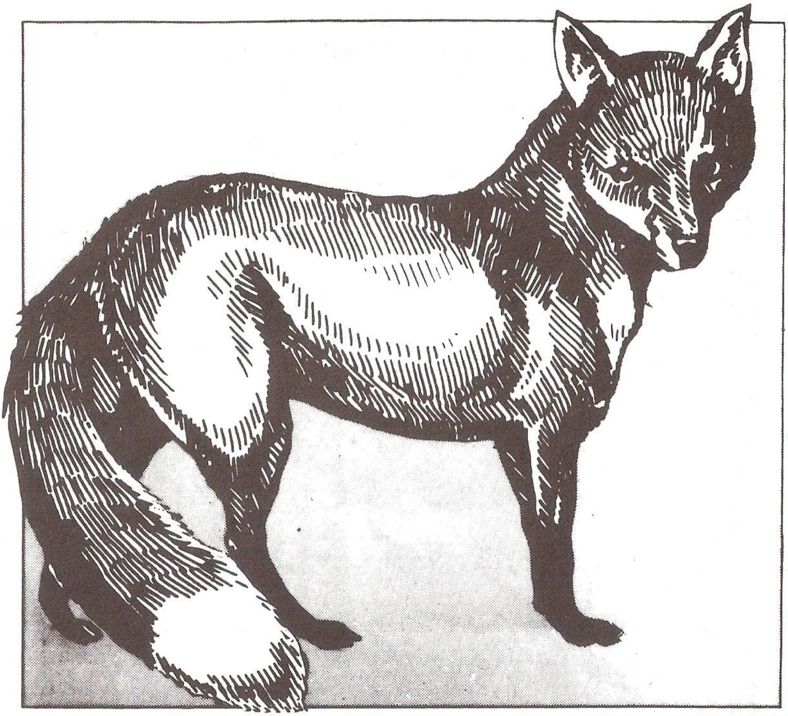
As the owner of an 8032 with a 3022 printer I've constantly been irritated by the fact that I couldn't list programs in lower case. I was thus delighted when I found that the ICPUG software library contained a 'lower case lister'. When I got my copy, I decided to transfer it from $033A to $027A, thus removing it from an area where it could be corrupted by the BASIC 4 disk commands. When I tried to run it, however, I found that it would not list any of the new commands associated with BASIC 4. A quick glance showed why this was so - The keyword table at $B0B2 in BASIC 4 occupies more than one page, and thus can't be accessed by a single register. A reference to the LIST routine in ROM allowed me to make the required changes which make the routine operate correctly with BASIC 4.

[Ed's note: Some printer ROM sets will accept secondary address 7. OPEN2,4,7:PRINT#2:CLOSE2 then OPEN4,4:CMD4:LIST then PRINT#4:CLOSE4.]

```
100 print"<CLR>This program corrects  a  bug
110 print"in the Commodore 3022 printer
120 print"which prevents it listing  in
130 print"lower case. To use it type:-
140 print"<DN>OPEN 4,4 : CMD 4
150 print"<DN>SYS 634 (the program lists)
160 print"<DN>PRINT#4 : CLOSE4
170 print"<DN>The program lives in the 1st
180 print"cassette buffer, and remains
190 print"there until  overwritten, or
200 print"the machine is switched off.
210 readl,h:fori=ltoh:readdt:pokei,dt:next
220 data 634,799
230 data 169,0,133,17,133,18,32,163,181,104,104,160,1,132
    ,9,177
240 data 92,240,70,32,225,255,32,223,186,200,177,92,170,
    200,177,92
250 data 201,255,208,4,224,255,240,49,132,70,32,131,207,
    169,17,32
260 data 70,187,169,32,164,70,41,255,32,13,3,201,34,208,
    6,165
270 data 9,73,255,133,9,200,240,17,177,92,208,16,168,177,
    92,170
280 data 200,177,92,134,92,133,93,208,178,76,255,179,16,
    218,201,255
290 data 240,214,36,9,48,210,170,132,70,160,178,132,31,
    160,176,132
300 data 32,160,0,10,240,16,202,16,12,230,31,208,2,230,
    32,177
310 data 31,16,246,48,241,200,177,31,48,5,32,70,187,208,
    246,73
320 data 128,208,161,72,9,192,201,219,144,8,201,224,16,4,
    104,73
330 data 128,72,104,76,70,187
```

--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

STRICTLY FOR BEGINNERS - 7

By Ray Davies.

Note: As far as I am concerned, DISK means disk drive; DISKETTE means the actual floppy thing; CASSETTE means the Cassette recorder itself; TAPE means the cassette tape. I have been asked by one of my (two) readers to try to explain File Handling. He had tried to adapt one of the programs I published previously in this column to his business, and had the problem of not being able to read the data he had recorded in a file back into the computer. He actually came from Sheffield to Derby one day to demonstrate the problem to me. The answer was simply a space in the write file title, and no space in the read file title.

I propose to start right from the very beginning. I will illustrate the differences between tape and diskette as I go on. If you have read my column before, you will know that this gave me the utmost difficulty and frustration when I first bought my computer two and a half years ago. So here goes.

File handling consists of first setting up a file, which means creating the data for the file, and recording this data onto tape or diskette. The process of setting up the file data usually involves the use of INPUT statements in your program. Here is a very small program to demonstrate this:

```
 10 INPUT"MAXIMUM NUMBER OF NAMES IN FILE";N
 20 PRINT"ENTER 'ZZZ' FOR SURNAME TO END"
 30 DIM SN$(N),FN$(N),B$(N),T$(N)
 40 FOR I=1TON:INPUT"SURNAME";SN$(I):
              IFSN$="ZZZ"THENI=N:GOTO70
 50 INPUT"FORENAME(S)";FN$(I)
 60 INPUT"BIRTHDAY";B$(I):INPUT"TELEPHONE NO.";T$(I)
 70 NEXT:NN=I-1
```

N.B. NN is the actual number of items in the data file, N is the Maximum Number. This program has now created four arrays entitled respectively SN$, FN$, B$, T$ in the computer's memory. The reason for these names is, I hope, obvious. If not, think of SurName, Fore Name, Birthday, Telephone. If you switch off now, the information is all lost. So we want to record the information. Let us do it on tape first. This entails OPENing a file on the tape, using the cassette, which must be plugged in to the appropriate socket at the back of the PET first (i.e. before switching the PET on). The tape is inserted into the cassette, and rewound. PET does not start recording for something like 10 to 14 seconds, so you needn't worry about Leader Tapes. Just ignore them. [Ed's note of caution: Cassette leaders vary from 2 or 3 seconds to as much as 19 seconds. The Vic-20 runs some 10% faster and thus waits 10% less. The preamble is used on Play to determine the average tape speed and should not be missed completely].

We now need more program to do this. Here it is:

```
100 OPEN 1,1,1,"RD"     (NOTES The space
110 PRINT#1,NN          (after OPEN is
120 FOR I=1 TO NN       (not necessary,
130 PRINT#1,SN$(I)      (but all the
140 PRINT#1,FN$(I)      (other punctuat-
150 PRINT#1,B$(I)       (ion IS.
160 PRINT#1,T$(I)
170 NEXT
180 CLOSE1
```

For disk users, the program would read (BASIC 4.0):

```
100 CR$=CHR$(13):OPEN1,8,2,"RD,S,W"
110 PRINT#1,N
120 FORI=1TON
130 PRINT#1,SN$(I);CR$;FN$(I);CR$;B$(I);CR$;T$(I)
140 NEXT
150 CLOSE1
```

Line 100 asks the computer to tell the cassette or disk to get ready to receive data from the computer, and to put it in a file called RD. For disk users, each item of data must be separated from the next item, so I use a carriage return. This is understood by the PET and disk as CHR$(13). To save typing CHR$(13) over and over again, I set another (smaller) variable (CR$) to equal CHR$(13). Any variable would do, so long as you remember that it is alphanumeric, and therefore must have the $ sign after it. And the disk file will be a sequential file (S) and we are going to write to the file (W).

Line 110 writes NN to the file. Don't forget to type PRINT in full. The question mark (?) abbreviation for PRINT will NOT be enough in this case.

Line 120 starts a FOR/NEXT loop to write the data to the file, one item at a time, until they are all entered.

After this it is ESSENTIAL to CLOSE the file. Otherwise the disk or cassette will remain 'listening' for more data. This could lead to your data being corrupted. All that is needed is CLOSE followed by the file number. Incidentally, this file number is nothing special. Any number will do from 1 to 255. The second number in the file opening sequence is the number assigned by Commodore to the various PET peripherals. 1 is a cassette; 3 is the PET screen; 4 is the printer; 8 is the disk. The third number is known as the Secondary Address. This usually tells the peripheral unit what to do with, or how to treat, the information it receives. For instance, our statement in line 100 - OPEN 1,1,1,"RD". This signals PET to open a line numbered 1. This line must be to the cassette (1). And the final 1 means that the data is to be written to tape, in a file named RD. If the final number, the secondary address, had been 0, then the cassette would have known that it was to READ data from tape back into PET. We can have up to 10 files open at once. Hence we could open1,1,1 to write data to file 1, open 2,1,1 to write data to file 2, and open 5,1,0 to read data from file 5. Of course only one file can be used at a time, but they can all remain open (listening). They only come into action when their own file number is used. In this example, by the way, the files would be closed as follows: CLOSE1,2,5. And of course, as stated previously, this is absolutely vital to ensure the safety of your data.

Now to retrieve the data from wherever it is stored. For the cassette user, the only changes to the program above starting at line 100 are: OPEN1,1,0,"RD" and INPUT on each line in place of PRINT. Here is an example:

```
 200 OPEN 1,1,0,"RD"
 210 INPUT#1,NN
 220 FORI=1TONN
 230 INPUT#1,SN$(I)
 240 INPUT#1,FN$(I)
 250 INPUT#1,B$(I)
 260 INPUT#1,T$(I)
 270 NEXT
 280 CLOSE1
```

For disk users:

```
 200 OPEN1,8,2,"RD,S,R"
 210 INPUT#1,NN
 220 FORI=1TONN
 230 INPUT#1,SN$(I),FN$(I),B$(I),T$(I):NEXT:CLOSE1
```

Then one needs a bit more programming to see what you have retrieved from the diskette or tape, such as:

```
 300 FORI=1TONN:?"<clr>"
 310 ?SN$(I):?FN$(I):?B$(I),T$(I)
 320 ?"<dn>PRESS A KEY"
 330 GETQ$:IFQ$=""THEN330
 340 NEXT
```

This will print each surname, forenames, birthday and telephone number on screen, one at a time, having the Press A Key message between each screenful.

If you wish to see them scroll up the screen one at a time, but not be erased immediately, then leave out the 'CLEAR SCREEN' in line 300, or replace it with a 'CURSOR DOWN'.

The other important things to remember are that the PRINT# and INPUT# statements must match precisely. i.e. If you have lines 130 to 160 as above in the cassette program, you cannot have, for example, in lines 230 to 260;

```
 230 INPUT#1,SN$(I)
 240 INPUT#1,FN(I)
```

etc., because this leads the PET to expect a numerical amount, whereas the data actually stored is alphanumeric. This will give a TYPE MISMATCH ERROR. Similarly, the file title must be precisely identical in both writing and reading operations. E.g. we have OPEN 1,1,1,"RD" above to write to the tape. If we OPEN 1,1,0,"R D" to read it, we will not read it. The computer (not just the PET, but any computer) insists on absolute accuracy o descriptions. That extra space in "R D" is read by the PET and forms part of the title. The reason for this is that the PET does not read the title as letters and space, but as their equivalent ASCII characters.

Dual disk users must also remember that a file name without a disk drive number will only attempt to use the last drive used. [If you still use DOS1.0 (2040), then expect trouble - DOS1 puts the filename to one drive and the data to the other, so in this case the drive number cannot be omitted. - Ed]. If you want the data to go to, or from, drive 1, you must specify this in the file title, e.g. OPEN1,8,2,"1:RD,S,R" will read from drive 1 only. When writing to files, it is possible to overwrite the previous data, i.e. erase it and replace it with new data, by using the '@' symbol in the file title. e.g. OPEN 5,8,5,"@0:RD,S,W" I do not wish to take up the whole Newsletter with this, so I will end there. This has only covered Sequential files, which are those which place the data piece by piece on the tape or diskette in order. Hence they can only be read in the same way, which takes a long time on tape, but not too long on diskette unless it is a very large file. Which reminds me, tape users must always rewind the tape between writing and reading, otherwise they may be attempting to read blank tape! I can see no reason why any of the foregoing should not be used on the Vic-20. If this is not so, I am sure someone will correct me. I am still open to answer queries on this or other PET problems. My address is 105, Normanton Road, Derby DE1 2GG. Next time I will write about Relative Files, or files which allow the user random access.

--oOo--

LITTLE KNOWN FACTS #1327

Commodore International are probably the largest single manufacturer of tape cassette decks in the world. Sales of CBM computers, worldwide, already exceed 2 million units and the cassette deck has been an integral part of Commodore systems since the launch of the PET in 1977.

--oOo--

REVIEW - DELPH CONVERTER BOARD

By Brian Grainger

Delph Electronics Ltd.,
4, Deeping Rd., Baston, Peterborough
Tel. 07786 535

I must at the outset of this review declare an interest. have been slightly involved in the testing of the product and wrote the instructions so I will limit this review to an indication of what the product will do.

For some time now there have been companies willing to convert FAT40 CBM machines to 8032 machines. Now Delph Electronics have provided the means to convert a 'thin' 40 (9" screen) into an 8032. The hardware consists of a neat PCB which fits above the ROMs on the existing 4032 (or 3032 upgraded to BASIC4) motherboard. The board plugs into the existing 'E' ROM socket and other connections go to the memory expansion connectors and the character generator. All connectors are provided and are simple to make. The character generator from the CBM is placed in the board. The software is a new 'E' ROM which is provided with the board.

Having connected the board your 'thin' 40 will behave exactly like an 8032 with all its functions and more. Because of differences between the CBM and business keyboards the latters new keys, such as 'Repeat' and 'Escape' are accessed in a different way. However the software has been written to ensure that all the graphic characters of the CBM are still accessible from the keyboard, something that the 8032 cannot do.

All the functions of the 8032 are made available. The 80 column display is very crisp. LISTings can be paused and restarted. Sreen windows can be set. Tabs can be set on the screen. Automatic indication of line end will be given via the bell (if user port sound is provided). All the additional screen editing commands are available. i.e. INST/DEL Line, ERASE Begin/End, SCROLL Up/Down, SET

Top/Bottom, Text/Graphic mode, Set/Reset Tab. A repeat key toggle is provided so that 'repeat' operates either on cursor control and space keys only, or on all keys.

One nice touch is that by use of the technique of multiple key press detection (See Vol.5 No.1 p.81) all the additional screen editing commands are available from the keyboard. Something that is not completely possible on the 8032.

The basic board is fitted with the capability of replacing the character generator with a 4K version. A software switch between either of the 2K character sets is provided. The 'E' ROM of the basic board is fully decoded allowing a 4K 'E' ROM to be used provided locations reserved for I/O ports are not used.

A number of versions of the board exist. Apart from the basic board described above a version exists which includes a 4K 'E' ROM, 4K character generator and RAM for an additional screen display (located at $8800-$8FFF). The extra 2K of character generator is used to store a mid-resolution character set which allows a plotting density of 160 x 200. Simple POKEs are used to switch between either the normal character generator or the mid-resolution characters. The additional screen memory can be used to store a second screen display or a mid-resolution plot. Part of the extra 2K in the 'E' ROM has been used to provide a 'flash' routine which very quickly alternates between the two screen displays with or without change of character set as well. This allows for example the possibility of a graph plot and annotated text to be mixed on screen.

The cost of the basic board is £ 149 plus VAT, the enhanced version £ 160 plus VAT. Contact John Bickerstaff, Discounts Officer, for special ICPUG prices.

--oOo--

HOME PROGRAMMERS WANTED

Commodore need a small number of home programmers, with experience of programming in both 6502 assembler and BASIC, to write programs for use on the Commodore 64 in primary and secondary schools.

This is for a new scheme under which Commodore is promoting the creation of educational software. Teams of teachers analyse where and how within the curriculum of a subject it is appropriate to use a microcomputer as a teaching aid. They then write a specification for each point at which they have decided a microcomputer can help in the classroom. In consultation with the teachers, Commodore turns this document into a formal specification that can be worked from by a programmer. The final stage is for the program to be checked by the teachers.

It is expected that all programs produced under the scheme will need to be partly written in assembler to exploit fully the graphics capability of the 6569 VIC chip.

The programmers will, of course, be paid.

Anyone interested in becoming a Commodore home programmer should write to John Collins, Technical Services Department, Commodore Business Machines (UK) Ltd., 675, Ajax Avenue, Trading Estate, Slough, Berks SL1 4BG enclosing some evidence of their programming competence.

--oOo--

LATE NEWS

The 500 (alias P128) seems now unlikely to be produced, Commodore concentrating its efforts on the 64 & 700-series. Possibly a new machine will appear, known as the TED, a cheap 40-col, 16K job without 6569 VIC and 6581 SID chips of the 64, ultimately to replace the Vic-20. The Corby factory is to start Vics & 64s in July. The 8250 to have Matsushita half-height drives restyled to match 700.

The SX100 is now to be known as the SX64 (that's the new portable C-64). So far only the single-disk drive prototype version exists. The 'Teacher's PET' is to be replaced by a 64, repackaged in a 4032 case (with colour) to sell in Sept. under £ 400. This is a UK-origin product.

--o0o--

Printed and distributed by COMPUPRINT COMPUTERS LTD.
4 Sands Road, Swalwell, Tyne and Wear. Tel 0632 888936