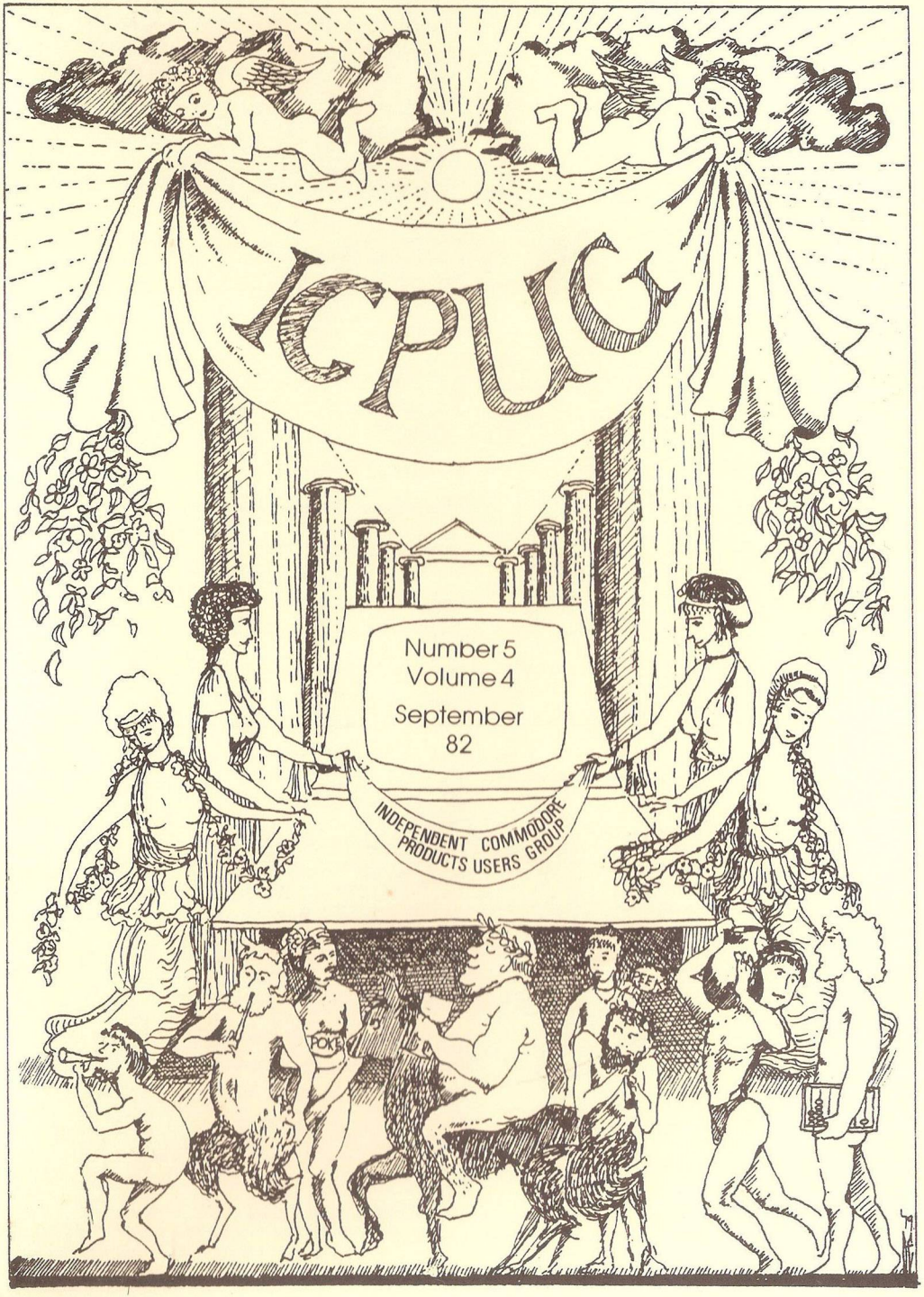

ICPUG
Number 5
Volume 4
September
82
INDEPENDENT COMMODORE
PRODUCTS USERS GROUP
POKE
PEEK

HONORARY NATIONAL OFFICIALS

Chairman:
Wing Cdr. Mick Ryan
164 Chesterfield Drive,
Riverhead,
Sevenoaks, Kent TN13 2EH
Telephone: Sevenoaks (0732) 453530

General Secretary:
Jim Tierney
11 Collison Place,
Tenterden,
Kent TN30 7BU
Telephone: 058-06 2711

Regional Co-ordinator:
Eli Pamphlet
7 Lower Green,
Tewin,
Welwyn Garden City,
Herts AL6 0JX
Telephone: Welwyn (043 871) 7325

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

Sub-Editor:
Mike Todd
27 Nursery Gardens,
Lodgefield,
Welwyn Garden City,
Herts AL7 1SF

Discounts Officer:
Dr. David Annal
142 Windermere Road,
Norbury,
London SW16 5HE
Telephone: 01-764 4043

INDEPENDENT COMMODORE PRODUCTS USERS GROUP

Vol. 4 No. 5 **Newsletter** Sept 1982

Europe's first independent magazine for PET users

Page	Contents

EDITOR'S NOTEBOOK

At this time of year we have our Annual General Meeting at which you have the opportunity of having a hand in the running of the User Group, either to share the work-load, or undertake some new venture on behalf of the Group. Several of the existing officials will not be seeking re-election, myself among them. I have been Editor of the magazine since the Group was formed some four years ago and would like a change. I will give my successor a hand with the next issue to show him/her the way, and will continue to contribute my regular features. The job is not without its perks, but the prime activity is to collate members' articles, correct spelling and grammar, cut out the naughty bits and convert the text to the format of the magazine. Software exists to aid the process and if necessary I will continue to print out the manuscripts ready for reproduction. Remember, no editor, no magazine. For my part, I shall not be having a rest, but will have more time to devote to a number of publications akin to the Compendium and its ill-fated sequel, the ROM Gazetteer.

R.D.G.

--oOo--

MEMBER'S PRIVATE SALES & WANTS

Sold Vic to buy PET - have following Vic accessories for sale. All 'as new'. Stack ROM switchboard (£ 25). Stack RS232 interface (£ 18). Greenwich Instruments GA20 Development board - will hold a total of 35Kbytes of RAM/ROM/EPROM/INSTANT ROM and memory expansion cartridges (£ 35). Three GA8 adapters - permits two 4K devices to occupy an 8K socket, or two 2K to occupy a 4K socket with automatic selection (£ 12 each). All have full instructions. Tel: Brian Atherton, Doncaster (0302) 539142.

--oOo--

MICROCHESS ON ICE

By Walter Green.

The modifications to MICROCHESS presented here are to enable a more flexible program which allows among other things (i) two players to play chess rather than player vs computer, (ii) saving of such a game which can be returned to at a later date and (iii) the making of a set position which can be returned to at will, so allowing alternative lines of attack to be tried.

The work was carried out with an old-ROM PET with expanded memory and the original MICROCHESS program. Modifications to cope with new ROM and the later versions of MICROCHESS are included (courtesy Brian Grainger).

First of all we need to prepare the program. In addition to MICROCHESS you will need a copy of the relocatable Supermon available from the ICPUG software library (SUPERMON1.REL for old ROMs, SUPERMON2.REL for new ROMs). Then carry out the following:-

1) Lower top of memory to $2400 by POKE134,0:POKE135,36 (new ROMs POKE52,0:POKE53,36). Type NEW and LOAD the relocatable Supermon and RUN. PET will break irto monitor automatically so save the Supermon machine code by :-

```
.S"SUPERMON",01,1B71,2400
```

This copes with either Supermon version although the new ROM code starts from 1E33 not 1B71.

2) You need to prepare a modified version of MICROCHESS. Although it loads to $2000 in memory it only uses up to $17DF ($17FC in the later version). To prepare the modified version reset the PET (this may not be necessary) and LOAD"MICROCHESS".
If you have the original MICROCHESS [PEEK(6128)=36] then do the following:-

```
POKE1043,1:POKE1044,0:POKE1049,2:POKE1050,0
POKE1362,1:POKE1363,0:POKE1368,2:POKE1369,0
POKE2294,0:POKE2295,0
```

If you have the later MICROCHESS version [PEEK(6128)=4] then do this instead:-

```
POKE1043,1:POKE1044,0:POKE1049,2:POKE1050,0
POKE1059,1:POKE1060,0:POKE1064,2:POKE1065,0
POKE1388,1:POKE1389,0:POKE1396,2:POKE1397,0
POKE2094,214:POKE2095,122:POKE2096,234:POKE2269,0
POKE2270,0
```

3) Now LOAD"SUPERMON" saved in 1) and if old ROM do a SYS7025 (new ROM, SYS7731). You will now be in monitor again. Transfer zero page by the command:

```
.T 0000 00FF 1A00
```

4) We are going to modify MICROCHESS so that pressing '-' will return to BASIC after transferring zero page to $1900-19FF & restoring zero page to the values from $1A00. Carry out this mod. by monitor commands to display and change memory as follows:-

```
.05B9 4C 00 18 (original MICROCHESS)
.05D7 4C 00 18 (later version)
.1800 A2 00 B5 00 9D 00 19 E8
.1808 D0 F8 BD 00 1A 95 00 E8
.1810 D0 F8 78 AD FE 03 8D 19
.1818 02 AD FF 03 8D 1A 02 4C
.1820 05 1D (old ROM)
.1820 07 FF (new ROM)
```

5) We now wish to modify MICROCHESS so it can set up a position that we specify at the start of a game. Do this by displaying and modifying memory as follows:-

```
.04A2 BD 2D 18 (original MICROCHESS)
.04BC BD 2D 18 (later version)
.055D 4C 50 18 (original MICROCHESS)
.057B 4C 50 18 (later version)
.1850 A2 21 BD 1D 14 9D 2D 18 (original MICROCHESS)
.1858 CA 10 F7 4C 0F 04 (original MICROCHESS)
.1850 A2 21 BD 07 14 9D 2D 18 (later version)
.1858 CA 10 F7 4C 0F 04 (later version)
```

It is suggested that you set up the standard board start position by the following:-

```
.T 141D 143E 182D (original MICROCHESS)
.T 1407 1428 182D (later version)
```

Finally, if you have the original MICROCHESS set $082C to 00 and $0830 to $24.

6) Return to BASIC (.X) and save the modified MICROCHESS by POKE124,0:POKE125,36 (new ROM POKE42,0:POKE43,36) and SAVE"MICROCHESSMOD". The original version of MICROCHESS can be saved more simply by SYS2062.

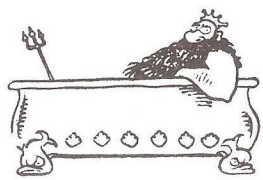
With the modified program you can call SUPERMON at any time with SYS7025 (new ROM SYS7731). Typing RUN and pressing 'return' will start MICROCHESS from the set position stored at $182D-184E (see later). If MICROCHESS is reset by the 'R' command the set position will be permanently replaced with the preset start position (standard board unless modified by the user). While playing chess you can break into BASIC by pressing '-' key. (Note that moves should be given without the '-'. e.g. E2E4).

Enter Supermon and you can review the zero page of MICROCHESS which has been transferred to $1900 upwards. In the original version memory from $1920-19B0 gives a record of moves to date.

We need to show how MICROCHESS stores pieces and positions so that any position can be set up. Look at the table below.

	BQR	BQN	BQB	BQ	BK	BKB	BKN	BKR	B Promoted Pawn
1	94	98	96	92	90	95	97	93	91
2	CB	CF	CD	C9	C7	CC	CE	CA	C8
3	1432	1436	1434	1430	142E	1433	1435	1431	142F
4	141C	1420	141E	141A	1418	141D	141F	141B	1419
5	1842	1846	1844	1840	183E	1843	1845	1841	183F

	BP	BP	BP	BP	BP	BP	BP	BP
1	9A	9C	9E	9F	A0	9D	9B	99
2	D1	D3	D5	D6	D7	D4	D2	D0
3	1438	143A	143C	143D	143E	143B	1439	1437
4	1422	1424	1426	1427	1428	1425	1423	1421
5	1848	184A	184C	184D	184E	184B	1849	1847



	WP	WP	WP	WP	WP	WP	WP	WP
1	AB	AD	AF	B0	B1	AE	AC	AA
2	E2	E4	E6	E7	E8	E5	E3	E1
3	1427	1429	142B	142C	142D	142A	1428	1426
4	1411	1413	1415	1416	1417	1414	1412	1410
5	1837	1839	183B	183C	183D	183A	1838	1836

	WQR	WQN	WQB	WQ	WK	WKB	WKN	WKR	W Promoted Pawn
1	A5	A9	A7	A3	A1	A6	A8	A4	A2
2	DC	E0	DE	DA	D8	DD	DF	DB	D9
3	1421	1425	1423	141F	141D	1422	1424	1420	141E
4	140B	140F	140D	1409	1407	140C	140E	140A	1408
5	1831	1835	1833	182F	182D	1832	1834	1830	182E

Lines 1,2 give the zero-page locations where the current positions of the relevant pieces are. Line 1 is for the original MICROCHESS line 2 for the later version. Each square of the board can be regarded as a two-digit hex number row,column. Top left is 00 and bottom right 77. The position of the piece is given by storage in the zero page location of the appropriate code for the square. Lines 3,4 give the preset storage positions taken up when a reset is performed. Line 3 is for the original MICROCHESS and line 4 for the later version. In this case however the board value is EOR'd with $77. (Alternatively one can view the board as being numbered with 00 as bottom right and 77 as top left!). Line 5 gives the storage position of positions taken when a RUN is executed. Again values are EOR'd with $77. If a piece is captured, it is denoted by a value of $CC in zero page, $BB in any other location.

To give some examples the white king is preset on square 74. Thus the preset position, $141D original version, shows $74 EOR $77, that is $03. As another example on loading the program POKE the RUN time position of the White queen, $182F(=6191)) with $BB(=187) and RUN. The board will come up with the white queen captured. Exit to BASIC via '-' and PEEK the copy of zero page location, $19A3 (original MICROCHESS). The reply is 172(=$CC).

By this means any position can be set up, e.g. a book problem. Two players can play a game by each in turn exiting to BASIC, poking the appropriate RUN time memory (remembering to identify captured pieces), and typing RUN to display new position. Such a game can be adjourned and SAVEd at any time.

Perhaps future modifications can be made to allow a game against the computer to be saved until resuming later.

Anybody finding further interesting points on the old ROM MICROCHESS may wish to contact me at 151, The Hatherley, Basildon, Essex.

--o0o--

ROUND THE REGIONS

On August 9th the Watford group celebrated their first anniversary with a demonstration of the Beeb 'B' computer to see how the other half live, plus a chance to play with the new 8250 disk drive. Future meetings will be September 6th, October 6th and November 10th, the latter two being a shift from Monday to Wednesday evenings.

New regional groups are being formed. At Petersfield there is a local self-help group of about 24 PET owners who meet on the second Tuesday of every month in the Computing Room of Bedales School, Petersfield, Hants. A Vic subgroup is to start at the Kinloss Komputer Klub, contact Mr. C.J.Smith, 49, Trenchard Crescent, RAF Kinloss, Ferres, Moray, Scotland.

Mrs. P.Dawsom, 189, Cannock Road, Westcroft, Wolverhampton, would like to start a Vic group in the Wolverhampton area. Anyone interest should contact her on 0902 721330.

Carol Taylor, 101, Courtland Avenue, Cranbrook, Ilford, Essex, has recently had her first group meeting, but would like further support for the group.

--o0o--

A LESSON TO BE LEARNT?

Attending and helping at our exhibition stands can be very tiring, and being tired can have rather terrifying consequences. ICPUG's chairman, Mick Ryan, went to the PET show ready for all contingencies with almost every disk (a hundred of them) he owned under his arm. After the show he took his box of disks and headed for home by train. To keep the disks away from any possible magnetic fields, Mick placed them safely in the luggage rack and settled down for the journey home.

He woke out of a doze suddenly to find himself at his home station and dived off the train - only realising some time later that he had left his disks on the luggage rack! Although very helpful, British Rail were unable to track the disks down despite a great deal of telexing and telephoning and soon Mick resigned himself to the awful truth - he had lost just about every disk he owned.

Mick spent the last two months trying to rebuild his collection, but many will know, it is impossible to recreate your own programs and replacing commercial programs can be very expensive. But with the help of his many contacts, he was at least able to replace a few of them.

Around the middle of August, two months after the event, he received a letter from a lad in Hastings (where the train had ended up) who had found the disks on the platform at one o'clock in the morning and had taken them home.

Unfortunately they had been tucked under a bed and forgotten about until one day, while cleaning the bedroom, the disks re-emerged. The lad probably had little understanding of their value, but nevertheless wrote to Mick who gladly zoomed off to Hastings and picked them up - with a suitable reward for their finder of course!

Mick's disks were all backed up, but on the reverse side of other disks. They were protected against the disk drive

going bananas, but nothing else. It's easy to be wise after the event, but I'm sure Mick will be making proper back up copies in the future! A sobering lesson to us all!

M.R.T.

--oOo--

DISK FILE

By Mike Todd.

It probably hasn't gone unnoticed that there was no Disk File in the July Newsletter. This was mainly due to a variety of factors, not last of which was that I was having to spend a great deal of time on my proper, full time job! The time spent on the Vic side of ICPUG was also taking up a great deal of time and so the Disk File had to be dropped.

Regrettably, something very similar has happened this time, except holidays have also intervened to make life even more hectic - so I have to apologise for there being no disk file in this issue either.

The fact that there are now several DOS versions around means that, whatever is included in the column, which is based on DOS2.1, may not be valid for DOS1.2, DOS2.2 or the DOS used on 8050, DOS2.5, or the single disk drives and so I am beginning to wonder if the series is really worthwhile. I'd appreciate any comments on this.

The fact that my own 3040 disk unit has failed hasn't helped! After operating for a couple of minutes after switch on, resetting the unit using the "U:" command I get the front LEDs flashing 7 times. Referring to page 39 in the January 1982 Newsletter, this indicates a RAM failure between $2000-$23FF and I now have to replace the RAM chip(s) in question.

What will happen next issue is in the lap of the Gods.

--oOo--

VIC MATTERS

By Mike Todd

When I offered an errata for Vic Revealed, I had little idea of the response that I would get and as I am writing this, I have a pile of stamped and addressed envelopes waiting for the errata. Hopefully, most of you should have your copy by now - if not, then it won't be far away.

VR v PRG

My criticisms of the book are echoed by most who wrote, although some disagree with my conclusion that the Commodore Programmers Reference Guide (PRG) was a better publication. Certainly the Commodore book has many errors too, but Nick Hampshire has cribbed much of "Vic Revealed" (VC) from it (or at least the same source), transferring Commodore's errors as well as adding a hefty supply of his own.

Note that the PRG I am referring to is NOT the early A4, ringbound folder that cost about 20 pounds. This was a very poor value document. The glossy cover, spiral bound, 285 page, A5 book has been available for some time now, and it is this one which I am referring to.

I can't really go into their differences and similarities much more, and would only add that about 90% of what is in VR is in PRG and about 60% of what is in PRG is in VR; PRG is easier to read, and much better presented - it has far fewer typing errors and about a third of the factual errors that VR has.

In terms of value for money I would give VR about 55% and PRG about 85%, although it is true to say that VR was (and still is in some places) the only reference guide available and so it is inevitably worth that much more. My advice? Avoid Vic Revealed like the plague and try to obtain the Commodore Programmers Reference Guide!!

HELP?

I would like to thank all of you who took the time to add in your letters such kind comments about ICPUG, the Newsletter and my Vic column. It certainly makes the many hours of work which go to make up the Vic column (and the Newsletter as a whole) worthwhile.

It is often very difficult to judge what you want to know about, although your letters are often a good guide, so if there's anything you want explaining, drop me a line. If I can't get round to writing about it, I will pass it on to one of the volunteers that are beginning to emerge. It may not be possible to answer specific queries personally - but we will do what we can, and at least include a mention in the Newsletter.

Also, if you think you have some expertise to offer, don't hesitate to contact me, giving details of your field of interest and equipment. The larger the pool of knowledge that we can draw on, the better we can serve those who are struggling. In fact, I would like to see a broadsheet published with the names and addresses of these volunteers, together with details of the help they can offer.

MORE DOWNGRADING

My downgrading technique described last time seems to have been very useful to many, and several have asked if it is possible to perform the downgrade without disconnecting the Super-Expander cartridge. Instead of SYS64824, use SYS41031 and the Expander will be available for use.

I also mentioned a program for doing all the work for you - well I'm including the listing for a program that will do just that. It contains all the essential "meat", but is not very user friendly. If you want to adapt it yourself and include grandiose graphics to show what is happening then that is certainly possible. If anyone does do anything exciting I'd like to have a look at the results.

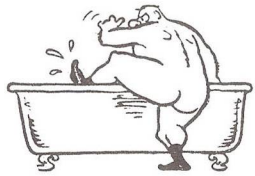
The program asks for the configuration you require (just press a number 0-6) and then asks if you want the expander in or out (press I or O). To make life easy, the guts of the program is in the routines at 1000 and 2000.

The subroutine at 1000 uses the variable C to set up the POKE and SYS parameters, where C is the configuration required. If C=0 then the configuration is unchanged.

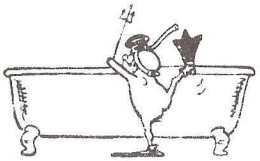
The actual POKEs and SYSs are done after line 2000 and these perform the reconfiguration.

Remember that running this program will reconfigure the Vic as if you had just switched on and so will erase the reconfiguration program itself (auto-cannibalism?). Also be aware that, if you try to set up a configuration which has RAM missing (e.g. to +24K with only +16K RAM expansion) the program will do it for you, but you may be in trouble when trying to do anything with the Vic, especially when using strings.

```
10  REM **** RECON - PROGRAM FOR RECONFIGURING THE VIC ****

100 PRINT "<clear screen>
110 PRINT "PRESS TO CONFIGURE AS
120 PRINT "----- ---------------
130 PRINT "  0   EXISTING
140 PRINT "  1   UNEXPANDED VIC
150 PRINT "  2   + 3K ONLY
160 PRINT "  3   + 0K
170 PRINT "  4   + 8K
180 PRINT "  5   +16K
190 PRINT "  6   +24K

200 GET T$:IF T$<"0" OR T$ >"6" THEN 200
210 C = VAL(T$) :  REM *** C IS THE CONFIGURATION NUMBER
220 GOSUB 1000  :  REM *** SET UP CONFIGURATION VARIABLES
230 PRINT:PRINT"CONFIGURING AS A VIC"
240 PRINT"WITH ";T$
```



```
300 PRINT
310 PRINT "DO YOU WANT SUPER-EXPANDER
320 PRINT "IN OR OUT (PRESS I OR O)
330 GET T$:IF T$<>"I" AND T$<>"O" THEN 330
340 GOTO 2000    :  REM *** DO THE CONFIGURING

1000 REM *** SET UP POKE VALUES (X,Y & Z)
1010 REM *** AND SYS VALUE (Q)

1020 FOR I = 0 TO C     : REM *** READ THE APPROPRIATE VALUES
1030 : READ X,Y,Z,T$
1040 NEXT I
1050 RETURN

1100 DATA   0,  0,  0, NO CHANGE
1110 DATA  16, 30, 30, NO EXPANSION
1120 DATA   4, 30, 30, 3K EXPANSION ONLY
1130 DATA  18, 32, 16, + 0K EXPANSION
1140 DATA  18, 64, 16, + 8K EXPANSION
1150 DATA  18, 96, 16, +16K EXPANSION
1160 DATA  18,128, 16, +24K EXPANSION

2000 REM **** DO THE CONFIGURATION

2010 REM **** IF T$="I" - SET UP TO INITIALISE VIA EXPANDER
2020 IF T$="I" THEN Q=41031

2020 REM **** IF T$="O" - SET UP TO INITIALISE NORMALLY AND
2030 REM ****              DISCONNECT EXPANDER
2040 IF T$="O" THEN Q=64824 : SYS64850

2100 IF C=0 THEN 2230     :REM ** DON'T SET NEW CONFIGURATION

2200 POKE641,0:POKE642,X :REM ** SET NEW CONFIGURATION
2210 POKE643,0:POKE644,Y
2220 POKE648,Z
2230 SYS(Q)               :REM ** RE-INITIALISE THE VIC
```



RS232 USERS BEWARE!

As anyone who uses input and output other than the screen and keyboard will know, it is essential to CLOSE a file when finished. This is especially true with cassette and disk write operations as data could be lost if the file is not properly closed, but CLOSE used with device number 2 (the RS232 port) has some problems associated with it.

When you OPEN the RS232 channel, the Vic grabs 512 bytes at the top of memory for input and output buffers and at the same time clears all variables (just like CLR). This is done deliberately, as strings that existed at the upper limit of memory will now be corrupt. Therefore OPENing an RS232 channel should be done at the very start of a program.

The CLOSE command grabs these bytes back again and, just like OPEN, performs a CLR operation! Consequently, the RS232 CLOSE command should only be used at the very end of a program.

Because of a bug in the routine which transfers the RS232 status byte to the ST variable, ST cannot be used to check for RS232 errors. Instead, use PEEK(663). The bug actually clears the RS232 status byte instead of collecting it!

Although the 3-line handshaking works OK, there are bugs in the multi-line handshaking which can cause the Vic to crash. For those who understand machine code, the bug appears to be a typing error in the source code. At locations $EFF4 and $F512 an attempt is made to check the DSR and CTS lines. Unfortunately, instead of checking location $9110 where these lines appear, the code checks location $9120!

If you are using RS232, especially for receiving data, it is recommended that you use the GET# command as INPUT# may hang if no carriage return is received. Also, you need to GET# characters as quickly as possible otherwise you are in danger of allowing the RS232 buffer to overflow.

CQ CQ CQ DE G8AZY!

As a bit of an aside, I would like to mention that we have a lot of radio amateurs amongst our numbers. I myself am licensed as G8AZY and can be sometimes heard on the 2-metre band. I'm always happy to talk about computing so if you hear me around, give me a call.

A popular use for personal computers is the sending and receiving of morse code (known as CW to radio amateurs) and teleprinter signals (RTTY). It may come as a surprise to many that radio amateurs use teleprinter signals for communication. As well as sending messages from one place to another, the technique can also be used for swapping computer programs - a job which is not too difficult on the Vic with the appropriate software.

Receiving RTTY and CW is a relatively simple task, although hand sent CW is much more difficult due to the variations in speed that occur. With a decoder, it is possible to eavesdrop on fascinating conversations between radio amateurs all over the world. It is also possible to eavesdrop on all sorts of other RTTY and CW from the mundane to the exotic (and may even include diplomatic communications and some of the major world news agencies).

As well as the appropriate software, a receiver covering the appropriate short wave bands and a suitable interface to convert the incoming tones to a digital form for the computer are needed.

There are a couple of such hard/soft packages currently available (see the mags for details) and I hope that we will soon be publishing circuit diagrams and listings for a Vic RTTY/CW package in the not too distant future.

As well as using computers for RTTY & CW, radio amateurs use them to record their contacts (although by law they must also keep a handwritten log), and even to pinpoint the amateur radio satellites.

THE FUTURE

The new range of Commodore products seems to be worrying many members. With the VIC-10 (aka MAX and ULTIMAX) at about 100 pounds, and the VIC-40 (aka Commodore 64) at about 240 pounds appearing in the next few months, some are having doubts as to the viability of the VIC-20.

A VIC-20 should cost in the region of 170 pounds (less if you shop around) and Commodore should be able to offer a hundred pound upgrade from VIC-20 to VIC-40. So VIC-20 owners should at least have a chance to upgrade relatively cheaply.

The VIC-10 and VIC-40 have similar basic facilities, although the exteriors are very different. These internal structures are not fully compatible with the VIC-20 which may mean some software could become unusable on switching over.

There is no doubt that the VIC-40 is a superb machine. I had one at home for a few days in May and was able to discover that the ROMs are virtually identical to the VIC-20, but the video chip and sound generator chip bear no relationship whatsoever to the VIC-20 chips.

In fact the 6567 video controller chip has 46 control registers and the 6581 sound chip has 28 - so the possibilities for PEEKing and POKEing are enormous.

Software writers, especially games authors, will delight from the ability to define large shapes on the screen and move them around very easily. The chip detects when these shapes (known as "sprites") bump into each other and can have one shape going "in front of" another. Normally, there can be up to eight on the screen at a time, but it is possible to define almost as many as you need and swap them on and off the screen with ease.

There is also a greater choice of colours than on the VIC-20, and high resolution graphics are made easier by selecting a bit-map mode which uses an 8K area of memory to

display 320x200 individual dots. This should be much easier to use than the VIC-20 method of high resolution graphics.

The sound chip is really an audio synthesizer. Remember the out-of-tune notes emanating from a VIC-20? Not on the VIC-40, since the frequency of each of the three voices (which can be triangular, sawtooth, pulse or noise) is controlled with 16 bits instead of only 7 on the VIC-20. Each voice has its attack rate, decay rate, sustain level and release rate definable and each is indivdually triggerable. The voices can be programmed to interact (ring modulated) and even fed through a digital filter.

The chip also has two analogue-digital converters for external potentiometers and there is even a random number generator derived from the third voice's oscillator.

The results of programming these two chips could be truly astounding - and remember, even the VIC-10 has these two chips on board!

The interface chips have been replaced by two CIAs (Complex Interface Adapters) which each contain the usual 2x8 bit input/output ports, timers and shift registers. They also contain a 24 hour time of day clock together with an alarm facility. These clocks are not used by the Vic and so should be "get-at-able" by programmers - possibly using PEEKs and POKEs.

And of course, we mustn't forget the 64K of RAM, some of which must occupy the same address space as the ROMs and I/O chips - but I would imagine that there is a way of bank switching these in or out.

The use of $0000-$0400 has changed little, although the USR vector at $00-02 has been moved as there is now an input/output register at $00-$01. The Kernel has been also been retained - so there should be no need to rediscover the inner workings of ROM & RAM leaving all the more time to explore the intricacies of the new chips.

USING CASSETTES

Many newcomers appear confused about the use of cassettes on the Vic. The handbook is far from clear and is actually inaccurate. Many complain that they can't write data to the cassette and read it back again and so I'm devoting the rest of the column to the simple use of cassettes. It is possible to be more adventurous than I am about to describe but this should provide a starting point for experimentation.

Writing to cassette is easy - it's just like printing on the screen except that each character is written on the tape as a series of pulses; reading back is just like typing characters on the keyboard, except that the characters come from the tape.

STARTING OFF - OPEN

Before writing anything to the cassette, a data path to the cassette must be opened using OPEN2,1,1,"TEST DATA" which tells the Vic that we want to use the cassette and it starts the operation by writing the title (in this case "TEST DATA") on the tape. This operation is vital as it marks the start of the data we are going to write.

The OPEN command also ensures that you've pressed the keys on the cassette machine. Be careful though, although the Vic can tell that you've pressed a key on the cassette, it doesn't know the difference between PLAY, RECORD or WIND. Therefore it is up to you to make sure that you press the correct keys when you get the message "PRESS RECORD & PLAY ON TAPE".

The first number after OPEN says how we will refer to the cassette in the rest of the program and could be any arbitrary number between 1 and 127. In this case we use the number 2 and will use PRINT#2 to access the cassette unit. The reason this is needed is that it may be necessary with disks and printers to access several different data channels in the course of a program.

The second number identifies the device we want to use - device number 1 is the cassette unit, and the third number is used to select read (0) or write (1). The interpretation of this third number depends upon the device being used.

Finally, the characters in quotes give the data a title which can be used when reading back to ensure that we are about to read the correct data.

During the OPEN command the cassette runs for a while and then stops. The cassette is then ready to receive data.

WRITING TO TAPE - PRINT#

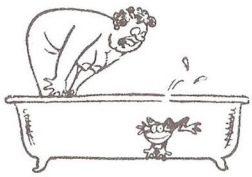
Instead of PRINT we use PRINT#2, which sends the output to the device specified in the OPEN2 command. Note that there is a comma after PRINT#2.

In the early stages of using cassettes, it is best to have only one variable for every PRINT#2 used and make sure that the statement doesn't end in a comma or semicolon. When we read the data back we need a "carriage return" at the end of each item - just as we have to press RETURN when typing data at the keyboard. This will be written to the cassette at the end of the data item provided that we don't suppress it with a comma or semicolon.

Once all data has been written, it is essential to tell the Vic that we've finished with the cassette unit. This is done with the CLOSE command which makes sure that the last few data items are written to tape (it is possible that they may still be in the Vic waiting to be written). In our example we use CLOSE2 - the number used being the same as the first number in the OPEN command. For reliable cassette operation the importance of CLOSE cannot be stressed enough.

Let's imagine we have set up two arrays in memory, each with 20 items in it - NAME$ contains the names of friends and NUMBER contains their telephone number. If we want to record this information on cassette we can do it as follows:

```
1000 OPEN 2,1,1,"PHONE NUMBERS"
1010 FOR I = 1 TO 20
1020 PRINT#2,  NAME$(I)
1030 PRINT#2, NUMBER(I)
1040 NEXT I
1050 CLOSE 2
```



READING IT BACK - INPUT#

Now we have to read the data back. After rewinding the cassette, we use the OPEN command again but in the form OPEN2,1,0,"TEST DATA" where the zero indicates that we want to read the data. We also include the name we gave to the data and the Vic will start reading the cassette, looking for the title "TEST DATA".

If the name is omitted and we use OPEN2,1,0 or even OPEN2 (since the Vic assumes we mean a read on the cassette unit if these parameters are omitted) the Vic will find the first title on the cassette and assume it's the one we want.

Once found, the data is ready to be read. We simply use INPUT#2 in place of INPUT and make sure that we read the data back in the same order as it was written, although the actual variable names don't have to be the same.

To read our telephone list back we can use:

```
1000 OPEN2,1,0,"PHONE NUMBERS"
1010 FOR I = 1 TO 20
1020 INPUT#2, A$(I)
1030 INPUT#2, B(I)
1040 NEXT I
1050 CLOSE 2
```

Using several variables with each PRINT# in the same way as is done on the screen can cause problems since there is no way for the cassette to identify the end of one item and the start of the next. Commas should be printed between these data items and the INPUT# must match the PRINT# in the number

of items it requests. Until totally familiar with these problems I would suggest avoiding having more than one variable in each PRINT# statement.

Also be careful not to read past the last item written. If necessary use an invalid item to indicate the end of the data (such as -999) and stop as soon as it is read. Reading beyond the last item written can result in some odd things happening and you'll more than likely get a "STRING TOO LONG" error message - even if you were looking for a number.

If you are in doubt as to what is actually on the cassette, it is possible to retrieve the data, character by character and print it on the screen as follows:

```
100 OPEN2
110 GET#2,A$
120 PRINT A$;
130 GOTO110
```

This will loop round getting individual characters from tape and writing them to the screen. When all the data has been read, a continuous block of spaces will be printed on the screen - when this happens, press the STOP key otherwise the routine will continue to try to read data from the tape.

A TRICK?

Debugging programs using cassettes is sometimes quite difficult and there is a trick to allow you to put the data on the screen instead of cassette without converting PRINT#2 to PRINT. If you change the OPEN from device 1 to device 3 (the screen) - in other words OPEN 2,3,1,"TEST DATA" - all data will now go to the screen instead of the cassette.

It is also possible to mimic the cassette read using device 0 (keyboard) in the OPEN command and simulate cassette input by typing the data yourself at the keyboard but be aware that you won't have the normal "?" prompt - instead you will simply get a flashing cursor.

FILES

The collection of data written to tape between the OPEN and CLOSE is known as a FILE and the first number in the OPEN and CLOSE commands is known as the "logical file number".

There can be several FILEs on a cassette, each with a different title or FILE NAME. If the Vic fails to find the file specified in the read OPEN command it will continue reading the tape until it finds the file it is looking for. This can be a problem if you have only one or two files on a tape and the Vic procedes to search the whole of a C90 for a non existent file.

To prevent reading beyond the last file, it is possible to label it as being the last on the tape. To do this we use OPEN2,1,2,"TEST DATA" - in other words we replace the 1 indicating that we want to write data with a 2 indicating that we want to write data and that this is the last data that will appear on the cassette.

If the cassette is searched and the file we are looking for is not found we will then get a "FILE NOT FOUND" error and the search will stop.

Because of the way the cassette records data, it is not possible to overwrite individual items of data to update them. Instead, the entire file must be read into the Vic, ammended and then re-written it in its entireity. This can be very tedious but is the only way.

LISTENING TO THE DATA

Because the cassettes used are identical to standard audio cassettes, it is possible to listen to a cassette on an audio cassette player. The first thing you will hear is about 9 seconds of pure tone - this is "leader" tone used to synchronise the read operation. This is followed by 3 seconds of buzzing - this will be the title block.

The data is recorded in blocks of 192 characters (which is why the cassette unit stops and starts while in use) and each is preceded by a burst of the pure tone followed by about 3 seconds of buzzing which is the data. There is a brief "hiccup" in the middle of each data block where the second (backup) copy of the data starts.

A program has the same leader tone and title block, but is recorded as one long block whose length depends on the length of the program - there will still be the "hiccup" in the middle where the second copy of the program starts.

PROBLEMS WITH CASSETTES

If the cassette is fully rewound, there is a fair chance that there will be up to 10 seconds of transparent plastic tape at the start. If this tape is less than 7 or 8 seconds long there should be no problem but if it is too long the leader tone, or even the title block, may not be written on the actual recording tape.

This is probably the single most likely cause of cassette read/write failures since the read operation cannot procede without this leader tone and title block being read. The solution is to ensure that the cassette is wound to the end of the transparent leader tape before recording or to use so called "leaderless" cassettes.

While there should be no problems in reading cassettes you've written on your own machine, there are sometimes problems when reading back tapes produced on other machines. This includes program tapes that you may have bought.

This is usually caused by the azimuth of the playback head in your cassette machine being different from the one that recorded the program or data.

The playback and record heads have a very tiny vertical gap in them, across which the magnetism is generated and transferred to the tape. It can be thought of as putting

vertical stripes of magnetism on the tape, and it is important that the head that reads the information off the tape must match these stripes exactly.

This is no problem if you only use the one machine since its record and playback heads are one and the same and any tilt in recording will be matched by the tilt in playback.

But if the heads are on different machines then problems can occur as a slight tilt on playback can result in the data being read incorrectly and errors will occur.

It is usual for heads to be aligned so that they are at right angles to the tape, and this is what is normally done at the factory. Transport, misuse, or just careless handling can upset this alignment and that's when the errors occur.

There are some cassette machines around which exhibit this problem and some which develop an azimuth error after a period of use. If you are finding difficulty in loading programs or data from other machines then it may be worth having your machine checked by your dealer.

FINALLY

There are a couple of points arising from the memory map in the July Newsletter that I'd like to clear up.

Firstly, these lists are based on ROM numbers 90148-01 and 901486-07 and these numbers were omitted from the bottom of pages 196-198.

All numbers in the lists are in hex with the exception of the line number range of location 014-015, the 60 times a second of 0A0-0A2, 192 in the tape buffer (0A6) and the number of characters per line in location 0D5.

Next time I hope to have a commentary on some of the more useful locations.

--oOo--

REVIEW

BASIC Editor Expandor (BEE) Supersoft

BEE is an acronym for BASIC Editor Expandor and is the name of a Swedish firmware chip recently available through Supersoft (where else ?). The package adds to the BASIC operating system not only the extensions that older hands have by now become familiar, such as the TOOLKIT range AUTO, DELETE, DUMP, FIND, TRACE, etc., but a selection of less common utility commands. I will discuss these briefly in turn.

ASSIST is similar to the Toolkit HELP command. After a run-time error, ASSIST will display the line in error with the error to line end displayed in reverse video.

AUTO has the same format as the CBM assembler editor. The command format is AUTO n, where n sets the line number increment. The command is disabled with AUTO on its own.

BOTTOM y,x defines the bottom right corner of a screen window. The default values are 24 and 79 from which one deduces that the chip is only available for 8000-series machines. TOP complements BOTTOM with default to 0,0.

BREAK simply enters the machine-code monitor, but saves one remembering SYS54386, or is it SYS54395 ?

CHANGE is a search-&-replace facility. The format is CHANGE:X$:Y$:,250-1000 or CHANGE"text""string",n1-n2 where the line number parameters are as in the LIST command. I found the delimiters took some getting used to compared to similar facilities elsewhere, but the newcomer should have no problem. FIND has the same format as CHANGE, but changes nothing, simply listing the line.

CLINK has nothing to do with 'go to jail', but is a cassette append command, not a merge. Consequently it reads a program from cassette (#1 or #2) and links it to the end of the one in memory, irrespective of line numbering. The responsibilty for the resulting line number sequence rests with the user.

CONVERT will do number conversion between decimal and hexadecimal. Hex quantities must be four-digit, e.g. CONVERT$00A9, and decimal 0 to 65535.

DELETE has the same format as LIST, but instead of listing the lines, it removes them, if present, from the program.

I became used to DOS Support before the arrival of BASIC4 and I still use Universal Wedge in preference to the BASIC4 disk commands especially when applied to drive #1. I was pleased therefore to see the inclusion of the DOS command, although not so much its syntax. DOS"string" sends the string to the disk command channel, but tends to lose out with DOS"$0" against DIr for drive zero directory. Its behaviour with CMD to output a directory to a printer is better than with DOS Support.

DUMP will list the simple variables created during a program run in the order that they were created. Scrolling can be prevented with the 'O' key and resumed with any other (this facility is available with the CHANGE command also).

A program on disk may be loaded and then RUN by using EXECUTE"progname",D1 ON U9 where drive number and device number are optional and default to 0 and 8, respectively. The unit (device) number must be in the range 4 to 31.

FLIST and PLIST respectively list to the screen a sequential and a program file to the screen without disturbing a program in memory. The format is as for EXECUTE and scrolling controlled as with DUMP.

GRAPHIC, LCASE and UCASE are related. UCASE is analogous to POKE59468,12 and puts the screen into upper case character mode. GRAPHIC corresponds to ?CHR$(142) and produces upper case text, but with line-spacing closed up ready for graphics. LCASE is equivalent to POKE59468,14 selecting the normal character set.

KILL is included to de-activate the chip, but can be re-activated with the normal SYS call.

MERGE is a true merge command and will take a program from disk and merge it with the one in memory. The syntax is the same as for EXECUTE.

NUMBER is perhaps one of the most useful in the set. It has the facility to renumber part of a program, thereby enabling one to improve the documentation of the program modules (assuming of course your program has some recognisable structure). The full syntax is NUMBERn1,n2,n3-n4 where n1 is the new number, n2 is the increment and n3 to n4 the line range to be renumbered. The variables n1 to n4 are optional and default to the entire program being renumbered from 100 in increments of 10.

SIZE has the same syntax as EXECUTE and displays the number of bytes your program would occupy. Can't think of a use for it, unless you have suddenly lost a great chunk of your available memory, or never had it in the first place.

START is similar to SIZE, but displays the start (load) address.

TRACE is similar, but not the same as the Toolkit trace. When active it displays the line number and first BASIC statement of the five previous lines executed, plus the next line to be executed. The speed of execution is set by TRACEn where 0<n<128. TRACE on its own turns the trace off.

UNITn1TOn2 changes the disk device number from n1 to n2 and is useful in using two disk drive units on line simultaneously.

All the commands are 'tokenised' and can thus be abbrieviated, eg Co for CONVERT, or Nu for NUMBER. My current documentation comprises seven pages of explanatory notes produced on a matrix printer, but knowing Peter Calver, Supersoft will no doubt produce a presentable reference book. At the time of writing, a retail price had not been announced, but Supersoft are at 1st Floor, 10-14, Canning Road, Wealdstone, Harrow, HA3 7SJ. Tel: 01-861 1166.

R.D.G.

--o0o--

COMAL CORNER

By Brian Grainger

First of all many thanks to all correspondents, in particular to Nick Higham again, Jim BacBrayne and Ui Cheah. They have all contributed to the latest COMAL news in some way. The Commodore show and subsequent new ICPUG members have also increased the frequency of my postman's visits. Welcome to all those newcomers to COMAL. Lets go through the points of note discovered since the last Newsletter. They all refer to version 0.11 but most are also relevant to 0.12

1) Pressing the 'STOP' key when executing LOAD, SAVE, LIST or ENTER will cause a hangup.

2) In COMAL there are no problems with the use of ',', ';' and ':' in DATA statements. Quote marks (") can also be used if they are written TWICE. e.g. 12"" LONG would be taken as 12" LONG.

3) A bug has been found where DEL with no line numbers deletes the whole program !

4) When using DIV and MOD with negative numbers funny things can happen. It is unlikely negative numbers will be used but if you have an application you are advised to test the results very carefully. They may not be what you expect.

5) Along with the version 0.12 the 8096 version 1.01 allows procedures to be executed by direct command AFTER they have been prepassed.

6) The word OUTPUT as in SELECT OUTPUT is an optional keyword. If omitted the interpreter will add it automatically.

7) There is a bug in the RENUM command such that if the renumbered lines go above 9999 (The maximum number in COMAL) the lines become unlistable. As the line numbers have no meaning in the COMAL language and are only used by the Editor the resulting program will RUN OK. For those who

think this would produce a good unlist facility think again. The process is reversible so a RENUM 10 command will restore the listing.

8) I have not made it clear that a FOR loop structure in COMAL is tested at the beginning of the loop unlike BASIC. Thus nothing will be printed if the following COMAL line is executed: FOR I=2 TO 1 DO PRINT I
The equivalent BASIC command would print the value '2' which is usually not wanted.

9) I am told that a shifted space will cause an error in COMAL.

10) It is clear that spaces cannot be used in variable names. However it is sometimes useful to give variable names which are more than one English word. There is an unwritten convention rapidly growing that a ' is used:
e.g. FILE'NO, DRAW'DISK, FORM'BAR are all valid names.

One other point of note with variable names. One cannot use the same identity for different types of variables. E.g. two separate variables A and A# cannot appear in the same program. This non duplicity of names also applies to procedure or function names as well as statement labels. This is because COMAL holds all its name references in one long table which are not separated by pointers to variable name start, strings start, etc.

11) Another shortform I forgot to mention. '!' can be used instead of REM or //. I am told that COMAL was developed from Data Geneneral's Extended BASIC which accepted this abbreviation.

12) An interesting word of caution to those with 8032 computers. You are advised to type = and NOT := for the assignment code. The reason is that if in error a shifted = is sent, you will have ':-' which means something entirely different.

Well that's all the new points for this Newsletter. What does the future hold ? Well I now have a copy of Len Lindsay's manuscript for his forthcoming COMAL reference manual. My initial impression is that it does for CBM COMAL what Raeto West did for the CBM Product Manuals. I hope to do a review next time. The COMAL board to allow COMAL users access to all CBM RAM space is to be distributed in this country by Ellis Horwood. See Newsletter Vol.4, No.3, P.167 for address. As for those who say COMAL is a flash in the pan & BASIC is God's gift to programmers, it is interesting to note that COMAL is the standard language on the Galaxy 1 computer as well as the Piccolo. It is also available for Gemini multiboard systems and the Newbrain computer. In addition COMAL-type languages exist for Research Machines computers (SBS) and CROMEMCO systems (Cromemco Structured BASIC). Whether it is called COMAL or structured BASIC, the language is here to stay.

--o0o--

INTERVAL MUSIC

By Nick Bailey.

Here is a small amusing subroutine which is useful in situations when reading large amounts of data, etc., where the delay would otherwise be excessive. The routine called INTMUSIC, is interrupt driven and so does not disturb other processing while in operation. The sound may be heard on a CB2 soundbox. Later 12" screen machines have this built in, but for BASIC4, the interrupt routine address $E62E must be changed to $E455. The POKEs to enable CB2 sound, viz POKE59467,16:POKE59466,15:POKE59464,0 should be given before calling the routine. Data for the music can start anywhere in memory and should be stored in the format: duration (jiffies), POKE value, duration, POKE value, etc, and should end: 01,00,00,00. The (start-of-table)-1 should be POKEd into locations 1099/1100 (lo/hi) and called with SYS1030. At the end, the interrupts are restored, but the tune can be aborted with SYS1041:POKE59466,0.

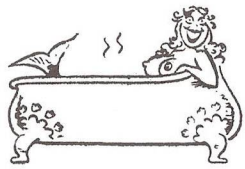
Enter the machine code dump first, then add the BASIC. As shown, the program inputs the tune from cassette data and dumps it to high memory. That dump is also included.

```
.:  0400 00 57 04 00 00 8F 78 A9    For BASIC4 to BASIC2
.:  0408 04 85 91 A9 23 85 90 58          change
.:  0410 60 78 A9 E4 85 91 A9 55     E4 to E6, 55 to 2E
.:  0418 85 90 58 60 EA EA EA EA
.:  0420 EA EA 01 CE 22 04 F0 03
.:  0428 4C 55 E4 20 42 04 D0 06      55 E4 to 2E E6
.:  0430 20 11 04 4C 4E 04 8D 22
.:  0438 04 20 42 04 8D 48 E8 4C
.:  0440 55 E4 EE 4B 04 D0 03 EE          ditto
.:  0448 4C 04 AD CE 7F 60 A9 01
.:  0450 8D 22 04 4C 55 E4 00 00          ditto
.:  0458 00 AA AA AA AA AA AA AA
```

```
PROGRAM NAME: INTMUSIC
   100 POKE53,PEEK(53)-1:CLR
   110 DT=PEEK(53)*256+PEEK(52)
   130 OPEN1,2,0,"THE ENTERTAINER"
   140 INPUT#1,P:POKEDT+I,P:I=I+1:IFST<>64GOTO140
   150 POKE59467,16:POKE59466,15:POKE59464,0
   160 POKE1100,INT((DT-1)/256)
   170 POKE1099,(DT-1)-256*PEEK(1100)
   180 SYS1030
   190 END
```

```
'The Entertainer'

.:  7F00 0A 19 0A 16 0A 1C 14 22
.:  7F08 0A 1E 14 26 0A 33 0A 2D
.:  7F10 0A 3A 14 45 0A 3D 14 4E
.:  7F18 0A 68 0A 5D 0A 75 14 8C
.:  7F20 0A 7D 0A 8C 0A 94 14 9D
.:  7F28 14 00 14 4E 0A D3 0A C7
.:  7F30 0A BC 14 75 0A BC 14 75
.:  7F38 0A BC 30 75 01 00 0A 75
.:  7F40 0A 68 0A 62 0A 5D 0A 75
.:  7F48 0A 68 14 5D 0A 7D 14 68
.:  7F50 30 75 0A D3 0A C7 0A BC
.:  7F58 14 75 0A BC 14 75 0A BC
.:  7F60 30 75 0A 8C 0A 9D 0A A7
.:  7F68 0A 8C 0A 75 14 5D 0A 68
.:  7F70 0A 75 0A 8C 30 68 0A D3
.:  7F78 0A C7 0A BC 14 75 0A BC
.:  7F80 14 75 0A BC 30 75 01 00
```

```
.:  7F88 0A 75 0A 68 0A 62 0A 5D
.:  7F90 0A 75 0A 68 14 5D 0A 7D
.:  7F98 14 68 30 75 01 00 0A 75
.:  7FA0 0A 68 0A 5D 0A 75 0A 68
.:  7FA8 14 5D 0A 75 0A 68 0A 75
.:  7FB0 0A 5D 0A 75 0A 68 14 5D
.:  7FB8 0A 75 0A 68 0A 75 0A 5D
.:  7FC0 0A 75 0A 68 14 5D 0A 7D
.:  7FC8 14 68 3C 75 01 00 00 00
.:  7FD0 AA AA AA AA AA AA AA AA
```



---oOo--

REVIEW

MATH HURDLER / MONSTER MAZE
£ 7.99 Creative Software, California

These two programs, marketted by Audiogenic as VIC Pack VP045, are to be recommended. The first is an educational game, and the second, a compulsive, 100% machine code beat-the-computer game.

In Math Hurdler, you choose from a difficulty level of 1 to 3, a speed of 1 to 5, and one of +, -, *, /, as the operator. A hurdler then sets off at the appropriate speed to jump ten hurdles. Whether he clears them or not depends on you supplying the correct answer to the sum. You score according to how many hurdles he clears and get more points per hurdle the more difficult the sums and the faster he is running. Teachers who have tried it like the idea but complain that they would like a more gradual increase in difficulty. In fact, difficulty level three is impossible at medium or fast speeds, even for addition. Difficulty level one at fast speed is a typing speed rather than a maths test. Nevertheless, overall it is worthwhile.

Monster Maze is in the 'monsters homing in on you trying to escape' catagory. The variation in this case is that the maze only appears around you as you explore it. Having dodged the first one, then two, then three monsters the amount of maze visible to you as you explore it, is reduced. Again, you compete against one, then two, then three. If you manage that your visibility range again

reduces and so on. A certain amount of tactics is involved, along with the normal reaction time versus luck element. It may not be a classic, but it's certainly worth its price. Both fit a basic VIC.

Brian Jones - Slough Region

--oOo--

REVIEW

SKI RUN

Not very impressive! The program, as one might expect, simulates that well televised Winter Olympic sport - skiing. It comes in two parts, instructions and game, which allows it to fit into the standard 3K VIC. With the exception of the sideways skier (a user defined character) there is nothing of special merit. The use of sound is minimal (wind and crash noises) and apart from green trees colour is no benefit.

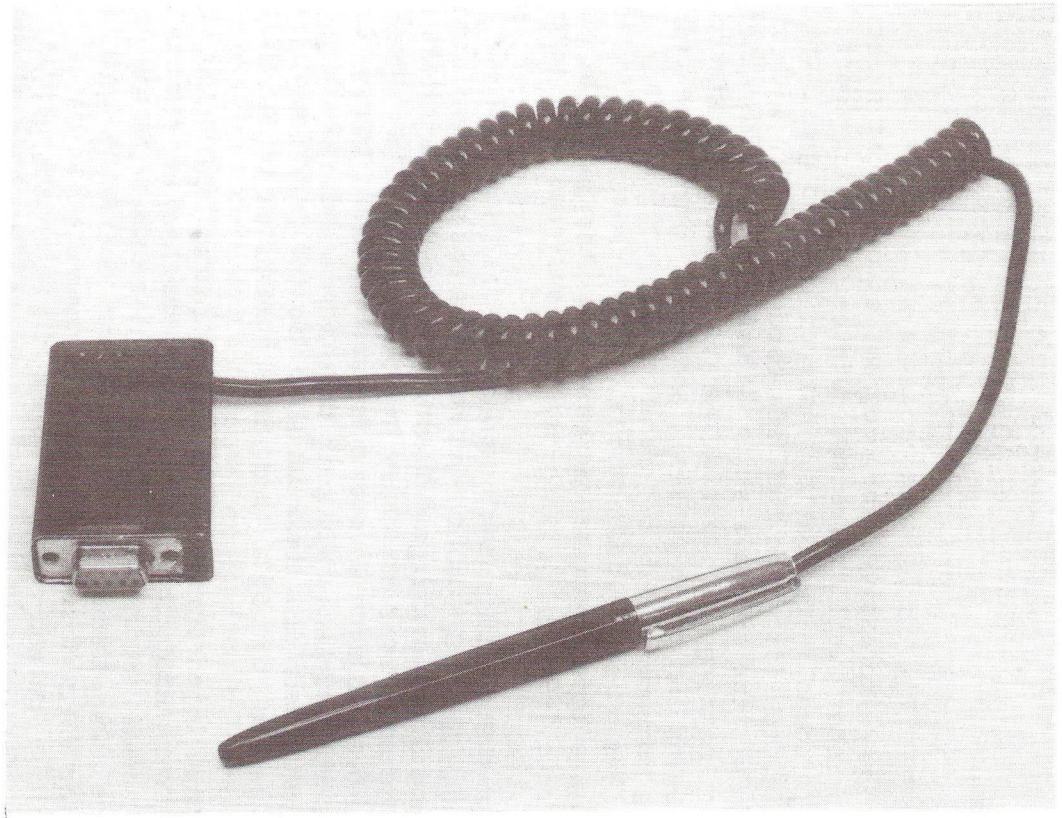
You are offered three varieties of the game, all at skill levels 1-9:

1. Slalom.
2. Giant Slalom.
3. Downhill. I could find no difference between Giant Slalom and Slalom and after twenty minutes play skill level 9 was the only one to cause me problems. (I'm no games expert, over 1500 on Invaders is rare for me). In fact the higher skill levels merely put down more random obstacles, but didn't speed up the movement, so the 'new best time' feature was a bit of a joke since it applied to all skill levels.

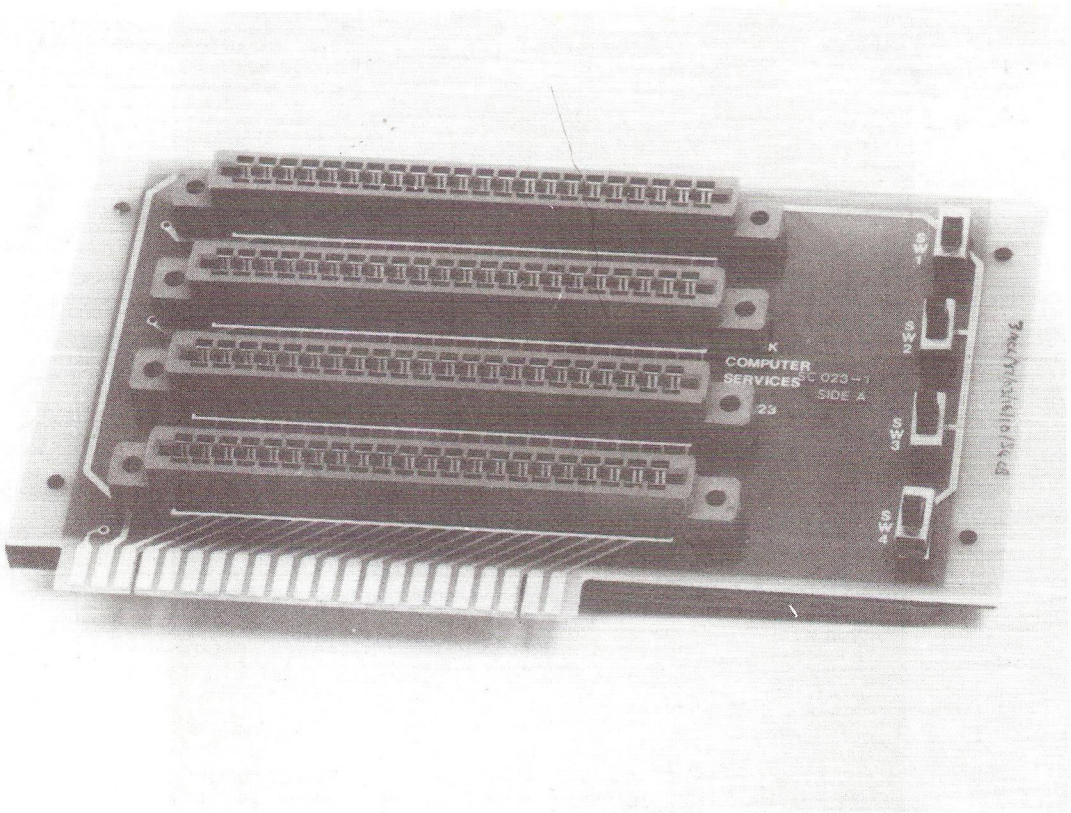
When I tried this game it was priced at £10 + VAT. On writing to RABBIT I received a letter saying they were closing down their VIC operations. However, at the Commodore show there was RABBIT and SKI RUN priced at £ 4.99 inc VAT. I still wouldn't recommend it as value for money.

Brian Jones

--oOo--

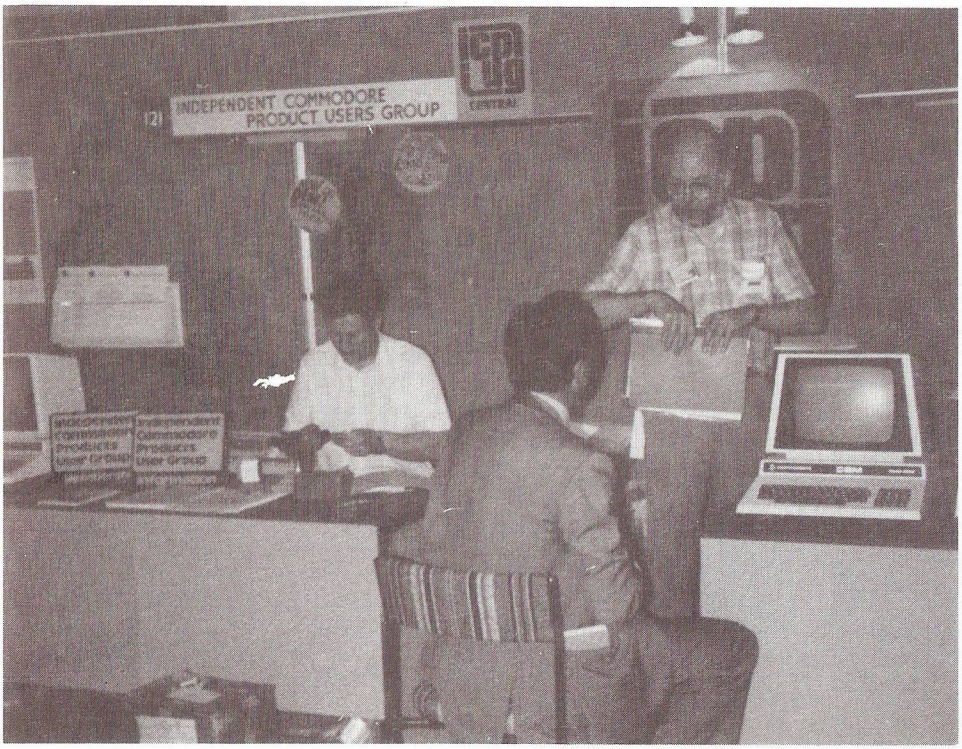


Vic light pen from stock

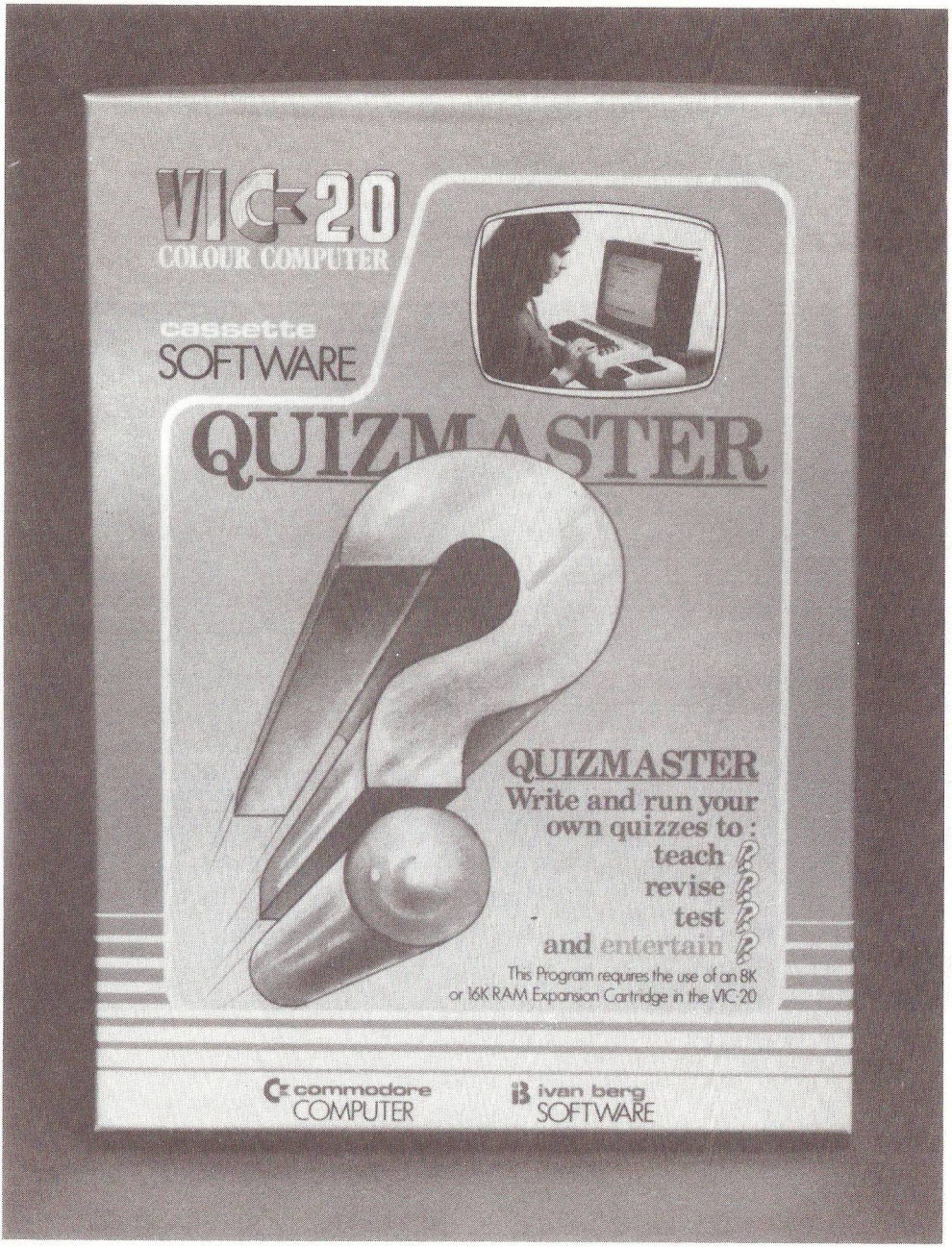


Expansion Board from stock

Jim Butterfield and Chairman Mick Ryan
at the Commodore Show

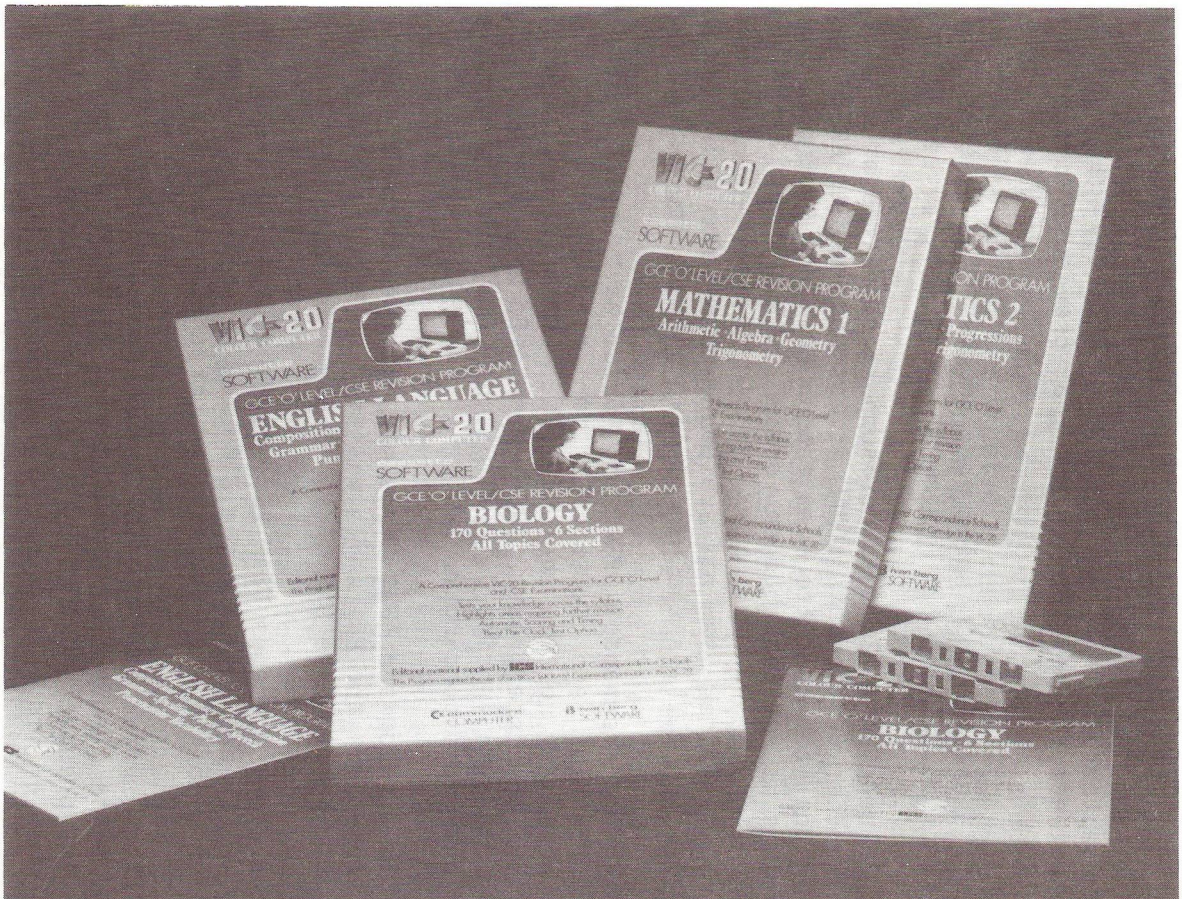


Part of the ICPUG stand at the Commodore Show



The VIC-20 Quizmaster program at £9.99

The new range of CSE Revision Programs



COMAL VERSION 0.12

By Brian Grainger.

In COMAL CORNER in the last Newsletter I mentioned that COMAL had now been revised and a version 0.12 was available. This article will identify the differences between the new version and that previously distributed by ICPUG.

1) There is no split version of COMAL as there was with version 0.11.

2) Version 0.12 does not support cassettes at all. On the earlier version of COMAL it was possible to ENTER and LIST files to cassette even though LOAD and SAVE did not work. This is not possible on version 0.12.

3) The final item not available on version 0.12 that was on version 0.11 is the use of ':+' with string variables.

4) The commands OPEN and CLOSE have been extended to OPEN FILE and CLOSE FILE for readability. It is not necessary to type the word FILE. It is added automatically by the interpreter. Programs SAVEd with Version 0.11 will LOAD into version 0.12 with the OPEN and CLOSE statements changed accordingly.

5) The command 'BASIC' is now implemented which does a warm reset of the PET which will set up BASIC. On version 0.11 one had to type SYS 64790 to have the same effect.

6) The commands TRAP ESC+ and TRAP ESC- are added which enable and disable the STOP key respectively. COMAL comes up with the STOP key enabled until commanded otherwise.

7) A system variable ESC has been added which, if the STOP key is disabled, has a value of TRUE (1) when the STOP key is pressed and FALSE (0) otherwise. When the STOP key is enabled its value is always FALSE.

8) The INPUT FILE and PRINT FILE commands have been extended to cope with input and print to relative access

files. I have not tested this facility but see no reason why it should not be the case.

9) As well as STATUS command causing the disk status to be displayed there is now a system variable STATUS$ which can be used like any other string variable.

10) It is possible to suppress the calling of error messages from disk during the input of code by the command SETMSG-. Messages can be restored by the command SETMSG+. The default case is with messages given. Run time messages cannot be suppressed.

11) While in version 0.11 it was essential to call a procedure by the command EXEC PROCNAME it is now no longer necessary. If a PROCNAME is found by itself it is assumed to be a procedure. This makes program listings more readable. If you wish the EXEC to be specifically listed one can send the command SETEXEC+. If one wishes to return to the default of no EXEC words then send the command SETEXEC-.

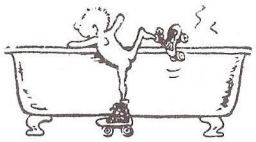
12) One could not make a direct call of a procedure in version 0.11 COMAL. In version 0.12, provided the procedure has been prepassed a direct command is acceptable. This also applies to functions.

13) A fundamental difference with version 0.12 is that procedures CANNOT be used as functions. There is now a separate FUNC header statement. The associated ENDFUNC also exists. The syntax for FUNC is identical for that of the procedure used as a function in version 0.11. The method of returning the function value is different however. The following example, which adds two numbers, will illustrate.

```
0010 FUNC ADD(I,J)
0020   RETURN I+J
0030 ENDFUNC
```

14) It is possible to return from a procedure at any point, as well as via the ENDPROC statement, by use of the RETURN statement. The following example, which prints a message dependent on the sign of a variable, will illustrate:

```
0010 PROC SIGN(I)
0020   IF I<0 THEN
0030     PRINT"THE VALUE IS NEGATIVE"
0040     RETURN
0050   ELSE
0060     PRINT"THE VALUE IS POSITIVE"
0070   ENDIF
0080 ENDPROC
```



15) The second fundamental difference in the new COMAL is the definition of substrings. Version 0.11 was very similar to MICROSOFT BASIC where A$(4:3) denoted the substring starting at the 4th character and length 3 characters. Version 0.12 COMAL defines a substring by its start and end characters. The above example becomes A$(4:6). Secondly where in version 0.11 one referenced an array substring by e.g A$(1,4:3), the equivalent in version 0.12 COMAL is A$(1)(4:6). While this incompatibility between COMAL 0.12 and its earlier version will no doubt cause some inconvenience when transferring programs, I do feel the current standard is more sensible.

16) It is now possible to send disk commands via the PASS command. The following example illustrates:
PASS"S0:MYFILE" will delete MYFILE from drive 0 of the disk unit. The disk commands used in the PASS command are identical to the BASIC2 commands.

17) There is now a NULL command which does precisely nothing ! An obvious use is to create a delay.

```
FOR I=1 TO 775*SECONDS DO NULL
```

If anybody can find another USEFUL example of the need for this command I would like to hear from you !

18) The statement ZONE=5 to set the print zone length is now a command ZONE 5. In addition there exists a system variable ZONE which holds the current value of the print zone length. PRINT ZONE is therefore a valid statement.

19) Finally, I am told the latest version of COMAL supports variable names up to 79 characters long ! I have not tested this facility and I cannot see much value in it !

As you can see COMAL version 0.12 is much more flexible than its predecessor and removes much of my earlier criticism of version 0.11. Apart from two omissions which I hope will be corrected, it supports the latest definition of COMAL. This is the COMAL KERNAL of MAY 1982 which has replaced the earlier NUCLEUS definition.

Copies of the new version are available through me if a disk and return postage is sent to 73, Minehead Way, Stevenage, Herts. SG1 2HZ. Disks will be formatted to 4040 standard.

--oOo--

SHOP WINDOW

A number of speech output devices have been mentioned in these columns, but the model 9816 from Time Electronics Ltd. has an IEEE-488 interface. The unit comes in a 3U 19" rack-mounting system and uses a Digitalker speech chip. The standard ROM holds 280 words and numbers. The output can be single words, numbers or complete phrases programmed from the IEEE-488 bus. A standard volume control determines the sound level from a speaker or headphones. Price is £ 550 from Time Electronics Ltd., Botany Industrial Estate, Tonbridge, Kent, TN9 1RS. Tel: (0732) 355993.

Using the PET as a dedicated processor ? Why not have your program auto-booted up at power-on ? The Progstore PPMS-01 enables 2K to 28K of EPROM-stored program to be 'permanently' available and called with either a SYS command, or booted up at power-on according to switch selection. Programs can be in BASIC, machine-code, compiled, or a combination. The system is ideal in hostile environments where disk or cassette storage would be unsuitable, or possibly to illiminate keyboard use/abuse (eg oily or heavily gloved hands). Priced at £ 345, details are available from Microscience Ltd., P.O. Box 14, Bramhall, Stockport, Cheshire, SK7 2QS. Contact D.L.Mawdsley or A.S.Blears on 061-477 3888.

Yorkshire Microcomputers (formerly Kingston Computers), famous for their Netkit, have now superseded it with the Netkit II. The main differences are additional commands and features, 4K EPROM and guest ROM socket on board, application software available, a better manual, and a smaller, easier to install, circuit board without the former steel case. Users of Netkit I will still be supported, or assisted with a possible upgrade. Yorkshire Microcomputers Ltd., are at 28, Ramshill Road, Scarborough, North Yorkshire, YO11 2QF. Tel: (0723) 78136.

Data acquisition products for the PET abound, so I will just mention contacts. Products are generally analogue I/O, digital I/O and relay drivers, IEEE-488 interfaced. DMS 550 comes from DI-AN Microsystems Ltd., Mersey House, Battersea Road, Heaton Mersey Industrial Estate, Stockport, Cheshire. SK4 3EA. Tel: 061-442 9768. Plant Interface Peripheral (PIP) comes from MC Computers Ltd., Park St., Newbury, Berks, RG13 1EA. Tel: (0635) 44967. Interact System from Tastronic Controls Ltd., Unit 16J, Gloucester Trading Estate, Hucclecote, Gloucester, GL3 4AA. Tel: (0452) 64244. FO Series boards from Cytel - these are not IEEE-488 but daisy-chain on the 'Cybus', an extension of the memory expansion port. Available from Cytel Instruments Ltd., 61, Woodburn Road, Carrickfergus, BT38 8HQ. Tel: (09603) 62494.

An IEEE-488 to Centronics 2K buffered interface with user programmable code conversion comes from Tastronic Controls Ltd., address as above.

Simplex Cash Book is a program to handle the book-keeping for small traders and is marketed by Micro-Simplex Ltd., Moss House, High Street, Mosborough, Sheffield, S19 5AE. Tel: 0742 484466, or 8, Charlotte Street West, Macclesfield, Cheshire, SK11 6EF. Tel: 0625 615000.

R.D.G.

--oOo--

PET REVAS

Club Software Review By Brian Grainger.

In this the second of my articles on club software I want to turn the spotlight on Dave Prentice and Tom Cranstoun of ICPUG-SE. When microcomputers had just started PCW ran a series of articles on REVAS, a reverse assembler. Like a disassembler this recreates assembler code from a binary machine code file. Unlike a disassembler however it attaches labels where necessary. I rather fancy these articles were a bit before their time as most hobbyists had yet to get to grips with BASIC let alone machine code. Now however, it is a different story. What Dave and Tom have done is to create a PET REVAS.

From a binary file PET REVAS will recreate assembler code directly compatible with the Commodore assembler system. Labels can be created for any calls to ROM routines, calls to locations within the code by 3-byte op-codes, branch instructions or any combination of the above. PET REVAS not only creates labels however, it has 2 other very useful facilities. A label editor exists so that label names can be chosen to have meaning or the addresses themselves can be changed if required. Also a table of labels for ROM routines is held with BASIC2 and BASIC4 address references. By careful use one can REVAS some BASIC2 code, tell the program to create label declarations for BASIC4 instead of BASIC2 and then the resulting assembler source can be reassembled by the Commodore assembler. I do not know of an easier way to convert from BASIC2 to BASIC4.

The functions available from PET REVAS are:

Create labels for code in memory or on disk. Labels can be created for JMPs+JSRs, branch instructions etc.

Disassemble from memory or disk to screen/disk/printer using the labels already created.

Store and retrieve standard labels and created labels to screen/disk/printer.

Edit the labels.

Send any label declarations to a device as an assembler source file.

Read an ASCII file to screen/printer. This function is useful if only to read the program instructions !

Exit to BASIC.

To give an example of the usefulness of this program I wanted to take a BASIC2 version of the RABBIT, convert it to BASIC4 and relocate it to RAM. I carried this out in the following steps:
1) Create labels to ROM routines assuming BASIC2 code.
2) Create labels to the code itself for all instructions including branches.
3) Disassemble the code using the labels created and send the source to disk.
4) Edit the labels which address the code to cater for the required relocation.
5) Send label declarations to a disk file BUT sending declarations for ROM routines as BASIC4.
6) Exit to BASIC.
7) Once in BASIC I used the source file editor to link the label declaration file to the source file created in 3). The resulting files were then input to the Commodore assembler and my code was created. It all sounds complicated but I'm sure machine code enthusiasts will realise the value of the program.

In conjunction with PET REVAS, Tom Cranstoun has also created an improved Editor for the Commodore Assembler system. As well as the normal facilities it includes the following:
1) Simpler PUT and GET commands which automatically read the disk error channel.
2) A DO command which executes the code in the first line of the source after the ';'. If one has a PUT instruction in the first line one can resave some edited source file simply by typing DO.

3) HILOAD inbuilt into the Editor.
4) GOLOAD will create a binary file from an object file (TIM format).
5) Display next page, display previous page, display same page, display a page from current cursor position.
6) Bidirectional scrolling.
7) Automatic repeat key which is turned off automatically when using file commands.

Versions of this editor exist for BASIC2, BASIC4 or 8032.

Both the above programs are available from ICPUG-SE Software Librarian, Tom Cranstoun, Flat 7, 10, Lancaster Rd., London, SE25 4AQ. Subject to the the usual conditions of providing a disk and return postage they represent, at a cost of nothing, excellent value to the machine code enthusiast.

GROUP DISCOUNTS

ICPUG can now arrange at least 20% off new VICs, 17+1/2% off all hardware and even more off some software and consumables - get in touch with me.

'Speakeasy' (reviewed p4 - Ed) now at the special single price of £ 68.85 including VAT + postage. This represents a saving of £ 12 ! Orders through me and cheques payable to me please. Driver programs and demos also from me, or ICPUG SE region - send disk and return postage.

Stack Computers will give 10% off list price to ICPUG members.

Contact Dr. David Annal, ICPUG Discount Organiser, 142, Windermere Road, London, SW16 5HE. Tel: 01-764 4043.

--oOo--

STRICTLY FOR BEGINNERS

By Ray Davies.

This is by way of an apology for some errors which have crept in to some of the programs previously printed under the above heading. Three readers have written to say that these programs did not work. Unfortunately this is absolutely true. The corrections to the relevant programs are printed here. All the page numbers relate to volume 4.

Page 40.

```
100 DIM A(20),A$(20):FOR I = 0 TO 19:INPUT A(I) :
    INPUT A$(I) : IF A(I) = 0 THEN 120
110 NEXT I
120 FOR I = 0 TO 19 : ? A(I) A$(I) : NEXT I
130 INPUT " WHICH NUMBER DO YOU WANT " ; N
140 FOR I = 0 TO 19:IF A(I) = N THEN ?"THE NUMBER YOU
    WANT IS " ; A(I) ; "AND THE NAME IS "; A$(I)
150 NEXT I
```

The fault with this program was the semi-colon (;) in line 100. BASIC does not permit this after an INPUT statement.

Page 81.

The next-to-last paragraph, last sentence, states 'This can also be done with cursors up, right, left or home.' As the previous part of the paragraph was discussing printing blank lines on screen, this does not make sense. What I meant was that all the cursor control keys can be incorporated into program statements, and will then have the effect that they have when used in direct mode. e.g. A cursor up <cu> in a program will cause the cursor to move up one line when it occurs. So if it was HERE the word 'the' after HERE would appear with it's first letter (t) in the line above the line HERE was in, and the other words would follow on that line, overwriting what was already there. So that is an occasion when you would not use a cursor up!

Page 83.

Line 230 of the second program on this page should read:

```
230 GET Q$ : IF Q$ = "" THEN 230
```

Incidentally, printers such as those that print our Newsletter do not have zeros with the diagonal line, so all those apparent O's in my programs must not be confused. They are all zeros, if not in words.
Page 121.

Several alterations are required in this program, so I'll just give you the corrected versions of the appropriate lines:

```
 90 PRINT"<dn>IS THIS A NEW PROGRAM?":GOSUB60
290 FORI=1TOK:IFLEFT$(A$(I),L)=N$THEN350
350 PRINT"<dn>NAME"TAB(17)"QUANT"TAB(25)"PRICE"TAB(32)
    "VALUE
360 PRINT"<dn>"A$(I)TAB(18)B(I)TAB(25)C(I)TAB(32)D(I)
```

Page 123.

```
50 DEFFNA(X)=INT(X*100+.5)/100:GOTO170
```

Explanation: Line 60 states that Z=FNA(Y). The computer does not know what FNA means without the extra statement in line 50 above. That statement is a user defined function. Running line 50 defines the function, which is then referred to in the rest of the program by the use of the command Z (or any other variable label) = FN (the name of the function) followed by any legal variable name. (In this case A).

The remainder of the function (after the equals sign) means that I want the number in question (X in the definition is what is called a 'dummy' variable) to be rounded to a two decimal place number. This number is then padded out with trailing zeros as necessary by the rest of the subroutine. This is very useful for financial programs, in which we do not want to see amounts of money such as £ 12.30021 or worse still £ 9.99999978E-03. This is the result if you subtract 4.99 from 5. The result using the FNA as above is .01, as one would expect. The rest of the routine also enables us to line up the decimal points.

That is the end of the corrections. I have since then 'revamped' the program on page 121, and anyone wanting a working copy should send me a suitable disk or cassette tape and I will SAVE it for them. (If you take me up on this, please enclose return postage). And my equipment is a

PET 4032 with 4040 disk drive, 4022 printer and a seldom-used cassette unit.

Finally, may I say once again that I am still available for any beginners requiring assistance with programs. My address is 105 Normanton Road, Derby, DE1 2GG. I promise you a reply within 48 hours, unless you choose a holiday period or a Postal strike!

--o0o--

DISK FILENAME TIP

By John Stout.

Concerning the naming of disk files I'd like to offer the thought that, contrary to a lot of systems around, the idea of putting an extension to a filename BEFORE the rest of the filename makes a lot of sense with CBM disk drives. For example using BAS. for BASIC programs, ASM. for assembler files, OBJ. for object code files and MCD. for machine code files allows one to use pattern matching to find all assembler files (ASM.*), all files to do with renumber (???.RENUMBER) and so on.

--o0o--

REVIEW

TRIOS EPROM

£ 20 for 12" models
£ 30 for 9" models

TRIOS is the third and latest EPROM from ICPUG's SE Region and adds commands to BASIC with code occupying the spare address space in the 'E'-ROM slot. This results in extra facilities without sacrificing 'spare' socket space, although these could no doubt be occupied by the other two EPROMs from the same authors. TRIOS will automatically link into these other two, BASMON and PLUSDOS, and will also coexist with the TOOLKIT.

TRIOS is activated by SYS59700 and enables a number of features, perhaps the the most useful of which is the bi-directional scrolling, a feature which first appeared in the POWER chip. NOSCROLL turns this feature off and SCROLL will reinstate it. The feature is only available in BASIC and attempts to use it when in monitor simply cause a listing of any BASIC program present to reappear. I would have liked to have had the facility available in the machine code monitor, as in MICROMON, so that one could scroll through the memory dump, or even a BASMON disassembly ! However, I do appreciate that 'E' address space is limited and possibly accounts for what I suspect may be just 'padding' by the inclusion of COLD (same as SYS64790 - resets PET), OLD which restores NEW, and DO, which will ignore line number and REM in the first line of BASIC and execute it as a direct command. The latter appears to work by listing the part line to the screen and inputting it to the BASIC input buffer.

As far as I could tell, the chip has its own routine to scroll down and does not use PET's routine entry at $E021. This may be for compatibility of coding with 9" screen machines which do not contain the CRT controller chip.

Paging is another feature of TRIOS and a listing can be paged from the cursor line, or one can select previous, same or next pages. These are selected using a number in conjuction with either shift or for 8032, ESCape.

BASIC2/4 machines with 9" screens need a hardware extra to further decode the 'E' slot, but will then with TRIOS give many 8000-series features such as delete to end of line, Bell, and will give greater compatibility with new software. The chip comes with a printed instruction sheet which explains in detail the individual commands. Since the chip is a plug-in replacement for an existing chip, apart from money, you lose nothing by having one. Orders should be sent to Kevin Viney, 95, Crofton Road, Orpington, Kent. Tel: (0689) 22443 (Cheques/POs payable to K. Viney).

R.D.G.

--oOo--

DATA APPENDING

By Philip Mortiboy.

PROGRAM NAME: DATALINE INST

```
 10 PRINT"<clr><rvs>DATA LINE WRITER<off>:- CONVERTS A
    SEQUENTIAL FILE ON TAPE INTO DATA";
 20 PRINT"LINES WHICH CAN BE APPENDED TO A BASIC PROGRAM.
    IT WORKSON";
 30 PRINT" BASIC 2 OR 4.":PRINT
 40 PRINT"STRINGS ARE INPUT, CARRIAGE RETURNS     ACTING
    AS SEPARATORS, AND";
 50 PRINT"CONVERTED INTO A PROGRAM LINE. (LINES ARE BUILT
    UP TO 80 CHARACTERS)."
 60 PRINT:PRINT"<rvs>TO USE<off>:LOAD DATA LINE WRITER.":
    PRINT
 70 PRINT"SET UP FILE TO BE READ AND TYPE RUN.":PRINT
 80 PRINT"DATA LINE WRITER WILL READ THE FILE,    CONVERT
    IT, AND POKE IT INTO";
 90 PRINT"HIGHER      MEMORY. WHEN AN END OF FILE MARK IS
      REACHED A";
100 PRINT" MACHINE CODE ROUTINE IS AUTO- MATICALLY RUN ON
    THE SCREEN TO MOVE";
110 PRINT"THE DATA STATEMENTS TO THE BASIC PROGRAM   AREA
    ":PRINT
120 GOSUB1000
130 PRINT"<clr>THE BASIC POINTERS ARE THEN RESET."
140 PRINT"WHEN THE PET RETURNS WITH THE READY     MESSAGE
    CLEAR THE SCREEN";
150 PRINT" YOU CAN THEN   LIST AND SAVE THE PROGRAM."
160 PRINT"(N.B. STRINGS OF 69 TO 248 CHARACTERS   WILL BE
    WRITTEN AS ONE ";
170 PRINT"LINE. CARRIAGE   RETURNS THROUGH THESE LINES
    WILL LOSE   THE EXCESS";
180 PRINT" CHARACTERS!)":PRINT
190 PRINT"ON APPENDING TO YOUR BASIC PROGRAM      CONVERT
    INPUT# AND GET# ";
200 PRINT"ROUTINES TO     APPROPRIATE READ STATEMENTS.":
    PRINT
210 PRINT"IF DATA LINE WRITER FINDS INSUFFICIENT  MEMORY
    OR A STRING LONGER";
220 PRINT"THAN 248     CHARACTERS THE PROGRAM TERMINATES.
    ":PRINT:PRINT
230 GOSUB1000:PRINT:PRINT"LOAD DATA LINE WRITER WHEN
    READY.":END
```

```
1000 PRINT"<rvs>              PRESS ANY KEY                    "
1005 GETA$:IFA$<>""THEN1005
1010 GETA$:IFA$=""THEN1010
1015 RETURN
```

PROGRAM NAME: DATLIN WRTER

```
 100 PRINTCHR$(147);"DATA LINE WRITER:BY:-P.N.MORTIBOY"
 110 LL=68:I=10:LN=10000:OP=1025:CP=3524:MS=PEEK(52)+256
     *PEEK(53):LC=128:QU=2
 120 POKE53,13:POKE52,196:POKE49,13:POKE48,196:OPEN1,1,0:
     GOSUB310
 130 GET#1,A$:IFA$=""THENA$=CHR$(0)
 140 IFASC(A$)>LCORA$=CHR$(44)THENQ=QU
 150 IFA$<>CHR$(13)THENB$=B$+A$:GOTO130
 160 IFQ=QUTHENB$=CHR$(34)+B$+CHR$(34)
 170 IFLEN(B$)>248THENPRINT"STRING TOO LONG":GOTO290
 180 IFLEN(C$)+LEN(B$)>LLTHENGOSUB220:GOSUB300:C$=""
 190 C$=C$+B$+",":K=1:B$=""
 200 IFST=64THENGOSUB220:GOTO250
 210 Q=0:GOTO130
 220 L=LEN(C$):IFCP+T+L>MSTHENPRINT"INSUFFICIENT MEMORY":
     GOTO290
 230 FORJ=1TOL-K:X=ASC(MID$(C$,J,1)):GOSUB340:NEXT:K=0
 240 PRINTLN;"DATA ";LEFT$(C$,L-1):LN=LN+I:RETURN
 250 X=0:GOSUB340:GOSUB360:GOSUB340:GOSUB340
 260 PRINTCHR$(147);:FORI=33008TO33061:READX:POKEI,X:
     NEXT
 270 NU=CP+T+1:GOSUB350:POKE33062,LO:POKE33063,HI
 280 PRINT"SYS33008:CLR":POKE623,19:POKE624,13:POKE158,2
 290 CLOSE1:NU=MS:GOSUB350:POKE53,HI:POKE49,HI:POKE48,LO:
     POKE52,LO:END
 300 X=0:GOSUB340:GOSUB360
 310 NL=CP+T:T=T+QU:NU=LN:GOSUB350
 320 X=LO:GOSUB340:X=HI:GOSUB340
 330 X=131:GOSUB340:X=32:GOSUB340:RETURN
 340 POKECP+T,X:T=T+1:RETURN
 350 HI=INT(NU/256):LO=NU-HI*256:RETURN
 360 NU=OP+T:GOSUB350:POKENL,LO:POKENL+1,HI:RETURN
 370 DATA173,196,13,141,1,4,238,241,128,208,3,238,242,128,
     173,39,129,205,242,128
 380 DATA 208,10,173,38,129,205,241,128,208,2,240,11,238,
     244,128,208,3,238,245
 390 DATA128,76,240,128,173,244,128,133,42,173,245,128,133
     ,43,96
```

--o0o--

DATA MAKER AND LOADER

By John Stout

The main object of this article is to present yet another machine code data maker and loader. They differ in that they include some attempt at error checking using checksums and data counts. The general idea is for every line of data to consist of four sections:
1. an identifier for the line of data concerned, in the programs this is assumed to be the same as the BASIC line number.
2. a count of the number of data items on the line.
3. the data items for that line.
4. a checksum, simply a number formed by adding up all the data items on the line.

The terminating line of the data has -1 as an identifier. The data handled need not be machine code or even numbers, although some other way of producing a checksum would be needed (add up the ASCII value of each character perhaps).

The data maker program is basically that published in the January 1981 ICPUG magazine by Jim MacBrayne without the section to delete the datamaker program.

The data reading program proceeds in two stages, first of all it reads through all the lines of data, checking for checksum errors. It then does a restore to start again (this means that as written the data statements must be the first in the program) and then treats it as machine code data, with a first line of data containing just the start and end addresses of the machine code. When reading it checks for two errors which can be missed by the first section of the program:
1. an error in the address header, e.g. missing altogether
2. a missing or duplicated line which results in an incorrect number of data items.

PROGRAM NAME: DATAMAKER

```
    10 INPUT"<clr>START ADDRESS<2rt>.<3lft>";HEX$:GOSUB 1000
       :SA=HEX
    20 INPUT"FINISH ADDRESS <2rt>.<3lft>";HEX$:GOSUB 1000:FA
       =HEX
    30 LN=60000:KQ=158:KB=623
    40 PRINT "<clr><3dn>";LN;"DATA";LN;"<lft>, 2,";SA;"
       <lft>,";FA;"<lft>,";SA+FA
    50 PRINT "KQ=";KQ;":KB=";KB;":I=";SA;":LN=";LN;":SA=";SA
       ;":FA=";FA;":GOTO 70<hcsr>"
    60 POKEKQ,2:POKEKB,13:POKEKB+1,13:END
    70 LN=LN+10
    80 PRINT "<clr><3dn>";LN;"DATA";LN;
    90 DT=I+9: IF(DT>FA) THEN DT=FA : F=1
    95 PRINT "<lft>,";DT-I+1;",";
   100 CH=0:FORX=I TO DT
   110 DT$=MID$(STR$(PEEK(X)),2):PRINTDT$;",";
   120 CH=CH+PEEK(X)
   130 NEXTX:PRINTCH;:IF (F=1)THEN PRINT ", -1";
   140 PRINT:PRINT "KQ=";KQ;":KB=";KB;":I=";I+10;":LN=";LN;"
       :SA=";SA;":FA=";FA;
   150 IF (F=1) THEN PRINT ":GOTO 180<hcsr>":GOTO 170
   160 PRINT ":GOTO70<hcsr>":
   170 POKEKQ,2:POKEKB,13:POKEKB+1,13:END
   180 END
  1000 REM CONVERT HEX$ TO HEX
  1010 HEX=0
  1020 FORH=1TOLEN(HEX$)
  1030 DI=ASC(MID$(HEX$,H,1))-48
  1040 IF (DI>9) THEN DI=DI-7
  1050 HEX=HEX*16+DI
  1060 NEXT H
  1070 RETURN
```

PROGRAM NAME: DATALOADER.00

```
   100 GOSUB59000:STOP
 59000 REM READ IN A MACHINE CODE ROUTINE WITH CHECKS
 59010 READ LN: IF (LN=-1) THEN 59080
 59020 READ ND: CH=0
 59030 FOR I=1 TO ND
 59040 READ DI: CH=CH+1
 59050 NEXT I
 59060 READ TC: IF (TC<>CH) THEN PRINT "ERROR IN LINE:";LN
```

```
59070 GOTO 59010
59080 RESTORE
59090 READ LN,ND,SA,FA,CH
59100 AD=SA:IF (ND<>2) THEN PRINT"ERROR IN ADDRESS HEADER"
      :STOP
59110 READ LN : IF (LN=-1) THEN 59170
59120 READ ND
59130 FOR I=1 TO ND
59140 READ DI: POKE AD,DI:AD=AD+1
59150 NEXTI
59160 READ CH: GOTO 59110
59170 IF (AD<>(FA+1)) THEN PRINT"ERROR IN NUMBER OF DATA
      ITEMS";AD;FA:STOP
59180 STOP
```

--o0o--

MATTERS ARISING...

Two typing slips occurred in the STOP-key disable article, pp212/213, firstly the line to update the clock an extra time should read BEQ CLK1. Secondly line 13 of the program should have the figure 144 inserted after the first occurrence of 133. Those of you that used the hex dump for comparision may have spotted the difference.

--o0o--

VISICALC AND COMMODORE 2022/2023 PRINTERS

By Brian Grainger

This article derived from the fact that I had a version of VISICALC which I wished to run with an early Commodore printer (2022/2023). These printers have the unfortunate problem that upper and lower case characters are reversed from most other printers. VISICALC needed to be modified to cope with this fact. The changes identified here work with VISICALC Version 1.70A. It may, with minor mods, work with later VISICALC versions but not having seen any I cannot guarantee this.

1) LOAD"0:VISICALC40",8 (or VISICALC80 if you use an 8032 machine).

2) Enter MLM by SYS1024

3) Change locations from 5565 as follows:

```
from .5565  OD 80 E6 47 DO CA 4C 6E 3D (original VISICALC40)
     .5565  OD 80 E6 54 DO CA 4C 6E 3D (original VISICALC80)

to   .5565  OD 11 80 E6 47 DO C9 EA EA (VISICALC40)
     .5565  OD 11 80 E6 54 DO C9 EA EA (VISICALC80)
```

4) Change locations from 551E as follows:

```
 from  .551E  A9 23 20 77 23 22 85 3D 2B BB
 to    .551E  4C BB 3D A9 23 20 77 23 2B BA
```

5) This step is optional. It alters the screen display to ensure a SETUP option is not given to the print command. Change locations from 5E70 as follows:

```
 from  .5E70  22 73 65 74 75 70 2C
 to    .5E70  20 20 20 20 20 20 20
```

6) Change locations from 55B9 as follows:

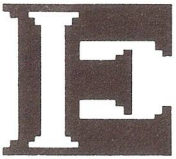
```
 from  .55B9  4C D3 96 A9 0A 20 EA 22
       .55C1  20 4B 21 0D 80 4C 1E 3D

 to    .55B9  18 60 A9 0A 20 EA 22 20
       .55C1  4B 21 0D 11 80 4C 21 3D
```

7) Save your modified machine code by the appropriate machine code save:

```
       .S"@0:VISICALC40",08,1A79,6B89
 or    .S"@0:VISICALC80",08,1A79,6B89
```

In using VISICALC with certain printers (e.g. 8027), I have found that a line feed character needs to be sent at the end of each line. Similar modifications to those above can be made to solve this problem. Should anybody using Version 1.70A require the mods for this problem send an SAE to me at 73, Minehead Way, Stevenage, Herts. SG1 2HZ and I'll send them by return.

--o0o--



TELESOFTWARE - The Realities!

By Bob Denton
Electronic Insight *800#

I have read so many articles on this subject recently but none have captured the essence of what as been happening. The current benefits of Telesoftware are in the fast and efficient distribution of non-commercial software. By non-commercial I mean the programs designed by teachers in individual schools that are not intended to be money-earning but need speedy and broad distribution to all schools. Also by non-commercial I mean that vast storehouse of programs each computer user club amasses that relies on members dedicated time to duplicate and distribute to other members.

Any writer of software who is wanting kudos for a program need look no further than Prestel Telesoftware as a medium.

The Background.

For over a year Prestel have been trying to set up a full service for Telesoftware - this does not just mean the creation of a format or the production of relevant hardware or the development of necessary software but the complete service to include all of these plus a good database of telesoftware programs.

IPC [ref 1] have had the word Telesoftware displayed on their free node [ref 2] *357# [ref 3] for over a year but still have no down-loadable programs, simply giving program listings that the Prestel users must note and then transcribe onto their computers.

The CET, Council for Educational Technology, *211#, have been actually offering down-loadable programs for some time but because of their concentration on the educational area and the Research Machines 380Z computer their impact has been very selective.

It was only when Tandata issued their MicroTantel that things started to move. For the first time a low cost means of connecting low cost microcomputers to Prestel was available. However it did not solve all of the problems. Theoretically the MicroTantel was able to interface with any computer but in fact it proved very simple for the Apple II but almost impossible with the PET.

The Hardware Emerges.

tantel

Late in 1981 all interested parties gathered to discuss how we could get the service underway and the reality of Telesoftware has its origins at that meeting.

The first decision was made to establish the CET format as the standard and the other contenders gracefully withdrew their systems. Although the format question is left dormant for now, it was made clear that if a better system should emerge then it would be reconsidered.

The MicroTantel shortcomings (and short delivery!) were discussed and the development of the TelesoftTantel created by Prestel ordering 250 units from Tandata. This unit has an RS232, or industry standard, interface that makes connections easier.

For the Apple II a simple connection from the existing MicroTantel's DIN cassette port to the Apple's twin mini-jack cassette port achieves the hardware solution. For the PET the TelesoftTantel is needed and the connecting lead requires a small printed circuit board built into it. For the TRS-80 again the TelesoftTantel is necessary and a 25-pin to DIN lead.

This product-by-product appraisal is all-important because no one-off solution is practicable. The largest computer user base is, of course, the Sinclair ZX80/1 and it is felt necessary to have a low-cost hardware solution to suit this market.

Prestel announced a contest with a £1000 prize which excited much comment but only two entries. Lion Viewdata produced a low-cost unit but the software would not

function on judgement day. Matochoice produced a Mullard Lucy-based unit (the heart of the Tantel too!) which had good embellishments and worked well but was not really a low price solution. The prize was split between the two and we now await their plans to reach production.

The Software Activity.

All the real accolades are deserved by the various individuals who were commissioned to write the necessary software packages. David Boulton and John Sharp for Apple, Stephen Rabagliati and Rod Eva for PET/VIC and Roger Hamlet for TRS-80.

For each micro two mirror-image programs were needed - the uploader and the downloader. The CET format determines the shape of the program but the uploader has to take a normal program listing, format it and then deliver it into the Prestel database. The downloader, of course, has to handle the reverse task and check the program has been correctly received. To date we have concentrated on the large volume micros but the task must continue through all of the other worthwhile systems.

To Become a Telesoftware User:

Any user of one of the previously mentioned micros may soon use telesoftware. This will require the user to obtain a Prestel jackplug and become a Prestel user and the purchase of the necessary package. The package consists of the relevant Prestel adaptor, the downloading software, the connecting lead and full instructions - see page *80061#.

The Database.

Once the service started to come together there was a flurry of IP [ref 4] activity to become involved. However, in order to ensure a focal point and to make sure that problems and queries could be effectively centralised, Prestel themselves have established a database called Aladdin's Cave *700#. On Aladdins Cave a broad range of public domain software has been established to reflect each micro as its package is available. Electronic Insight has

been involved in acquiring these programs and maintains a full subject index on *80066# with almost one hundred programs available now! Further, many computer user groups who have contributed this software have a news and comments section on *8008#.

The Future.

To permit commercial software to utilise these techniques requires one of a variety of methods. As most of you are aware the 50p maximum charge per frame is to be increased, but it is likely to come about by the use of a 'Gateway-ed' third-party database or by 'dongle'-ising micros (the 'dongle' is a component that is hired to decode the program received - without it, the program would be meaningless).

For a detailed catalogue and brochure please write to: 'Electronic Insight', Bushfield House, Orton Centre, Peterborough, PE2 0UW. Further, if you require a kit for the Apple II, PET 3000, 4000, or 8000 please specify which and ask for a quotation, or: Phone 0733 237111 business hours, 0234 721088 out of hours.

References.

[1] IPC - Publishing House and Prestel Information Provider.

[2] NODE - page number under which an IP holds his information.

[3] *nn# - reference to a Prestel page.

[4] I.P. - Prestel information provider.

--o0o--

THOUGHT FOR THE MONTH

Remember, anything that begins well ends badly; anything that begins badly ends worse.

--o0o--

COMAL TOWERS OF HANOI

By Brian Grainger

I recently had a letter from Jim MacBrayne (Region T organiser) who was amazed at the speed of COMAL. He was running the Towers of Hanoi program which comes with the latest COMAL disks and likened its speed to that of machine code. This program was written for 80-column machines so I set about modifying it for 40-column machines. It is slightly slower than the original since the 40-column machine does not have a screen window capability. It is however extremely fast and serves to introduce a powerful COMAL feature which helps in this speed. I am talking of recursive procedures where a procedure (subroutine for BASIC fans) can call itself. The usual example given for this idea is that of calculating the factorial of a positive number. However the Towers problem is a classic example. The problem is simple. You have a pile of discs of varying sizes piled on a peg so that a smaller disc always lies on top of a larger disc. The problem is to move the pile from the peg on which they are set to one of the two other empty pegs, by moving the discs one at a time so that at all times a larger disc is NEVER placed on a smaller one. The problem is easy with a small pile (3-5 disks). You may like to try it with seven, but if you want a real problem try it with twelve ! With a bit of thought the solution is simple enough:-

1) Move all but the bottom disc of a pile to a peg
2) Move the bottom disc to the other spare peg
3) Bring back the discs moved in 1) on top of the disc moved in 2).

Unfortunately all we have succeeded in doing is defining the solution in terms of another problem - how to move a pile of discs one less than the original pile. However if we repeat this procedure we eventually define the problem in terms of moving one disc and that is easy- just move it! If we write a computer program to solve this problem we need a routine to move a pile of 'X' discs say. This can be programmed by calling the same routine with X-1 discs, calling the routine with one disc and finally calling the

routine with X-1 discs again. As we keep breaking the program down, the routine will call itself over and over again, nesting itself to a level dependent on the number of discs. It is possible to program the problem in BASIC but because of the limited stack space available to subroutines and the problem of keeping track of the parameters being passed at each stage it is not easy. COMAL with its dynamic stack limited only by memory and recursive procedures is ideal for the program. Here is a COMAL program to solve the problem. It runs in all COMAL versions and its simplicity does much to speed it up. I challenge anybody to write such a speedy program in BASIC. Remember the COMAL program is about 2K long and variables and stack perhaps use as much again.

```
0010 //
0020 dim pile(19,3), top(3)
0040 dim space$ of 12, filler$ of 12, bar$ of 14
0050 //
0060 space$(1:12):="" // fill with spaces
0080 //
0090 exec initialize
0100 exec hanoi(max,1,2,3)
0110 end
0120 //
0130 proc initialize
0140  exec clear
0150  exec read'no'of'disk
0160  while 3>max or max>12 do
0170   print "at least 3 and no more than 12 disks!"
0180   exec read'no'of'disk
0190  endwhile
0200  exec clear
0210  top(1):=max
0220  for i:=1 to max do
0230   pile(i,1):=max-i+1
0240   exec draw'disk(i,1,pile(i,1))
0250  next i
0260  exec cursor(23,1)
0270  print tab(5),"pile 1",tab(18),"pile 2",
      tab(31),"pile 3"
```

```
0280  print
0290  print tab(7),"tower of hanoi with";max;"disks.",
0300 endproc initialize
0310 //
0320 proc read'no'of'disk
0330  input "<home,5dn>how many disks:    "+chr$(157)+
      chr$(157)+chr$(157): max
0340  max:=int(max)
0350 endproc read'no'of'disk
0360 //
0370 proc draw'disk(no,pile,disk)
0380  exec cursor(21-no,13*pile-11)
0385  exec form'bar(disk)
0390  print bar$
0400 endproc draw'disk
0410 //
0420 proc move'disk(from'no,from'pile,to'no,to'pile,disk)
0425  exec form'bar(disk)
0430  // raise disk:
0440  j:=from'pile*13-11
0450  for i:=21-from'no to 20-max step -1 do
0452   exec cursor(i,j)
0455   print space$(1:12)
0460   exec cursor(i-1,j)
0465   print bar$
0470  next
0480  exec cursor(19-max,13*from'pile-11)
0490  if from'pile<to'pile then
0500   // push to the right:
0510   for i:=1 to 13*(to'pile-from'pile) do print<space>
       chr$(148),
0520  else
0530   // push to the left:
0540   for i:=1 to 13*(from'pile-to'pile) do print<space>
       chr$(20),
0550  endif
0560  print
0570  // lower disk:
0580  j:=to'pile*13-11
0590  for i:=19-max to 20-to'no do
```

```
0592    exec cursor(i,j)
0595    print space$(1:12)
0600    exec cursor(i+1,j)
0605    print bar$
0610   next
0620  endproc move'disk
0630  //
0640  //- main routine, is called recursively -//
0650  //
0660  proc hanoi(no'of'disks,source,via,dest)
0670   if no'of'disks>1 then
0680    exec hanoi(no'of'disks-1,source,dest,via)
0690    exec hanoi(1,source,via,dest)
0700    exec hanoi(no'of'disks-1,via,source,dest)
0710   else
0720    top(dest):+1
0730    pile(top(dest),dest):=pile(top(source),source)
0740    exec move'disk(top(source),source,top(dest),dest,
        pile(top(source),source))
0750    top(source):-1
0760   endif
0770  endproc hanoi
0780  //
0790  proc form'bar(disk)
0800   if disk mod 2 then
0810    bar$:=chr$(18)+chr$(161)+space$(1:disk-1)+
        chr$(146)+chr$(161)
0820   else
0830    bar$:=chr$(18)+space$(1:disk)+chr$(146)
0840   endif
0850   filler$:=space$(1:(12-disk) div 2)
0860   bar$:=filler$+bar$+filler$
0870  endproc
0880  //
0890  proc cursor(row,col)
0900   poke 216,row-1
0910   row:=32768+(row-1)*40 // start of new line
0920   poke 196,row mod 256
0930   poke 197,row div 256
0940   poke 198,col-1
0950  endproc cursor
0960  //
```

```
0970 proc clear
0980  print chr$(147),chr$(142),
0990  print "the tower of hanoi"
1000 endproc clear
1010 //
```

--o0o--

MEN & WOMEN

```
  100 REM MEN & WOMEN-FOR COMMODORE 8032
  110 REM BY PETER PETTS.................
  115 REM FOR BUSINESS USERS WHO HAVE NEVER PLAYED WITH
      GRAPHICS TAKE CARE TO
  120 REM DISTINGUISH BETWEEN FIGURE 1 AND THE LETTER I.
  130 PRINT CHR$(142)"<clr>";
  140 M$="<4rt>"+CHR$(213)+CHR$(201)+"<dn><2lft>"+CHR$(202)
      +CHR$(203)+"<dn><4lft>"
  150 M$=M$+CHR$(164)+CHR$(206)+"<rvs>  <off>"+CHR$(205)
  155 M$=M$+CHR$(164)+"<dn><4lft>"+CHR$(223)
  160 M$=M$+CHR$(223)+"<dn><2lft>"+CHR$(233)+CHR$(223)+
      "<dn><2lft>"
  165 M$=M$+CHR$(165)+CHR$(167)
  170 W$="<4rt>"+CHR$(213)+CHR$(201)+"<dn><2lft>"+CHR$(202)
      +CHR$(203)+"<dn><4lft>"
  180 W$=W$+CHR$(164)+CHR$(206)+CHR$(218)+CHR$(218)+CHR$
      (205)+CHR$(164)+"<dn><4lft>"
  190 W$=W$+CHR$(223)+CHR$(233)+"<dn><2lft>"+CHR$(233)+
      CHR$(223)+"<dn><2lft>"+CHR$(165)
  200 W$=W$+CHR$(167)
  210 FORJ=1TO3:IFJ=2 OR J=3 THEN PRINT"<8rt>";
  220 FORI=1TO6
  230 PRINTM$;"<5up>";W$;"<5up>";
  240 NEXTI:PRINT"<6dn>";
  250 NEXTJ
  260 PRINT"<2dn><36rt>PRESS SPACE TO EXIT"
  270 GETA$:IFA$<>" "THEN270
  280 PRINT"<clr>"CHR$(14)
  300 REM PETER PETTS, BRAMLEY HALE, WRETTON, KING'S LYNN,
      NORFOLK PE33 9QS
```

COMMODORE COLUMN

If by now you haven't ordered a copy of Superscript from ICPUG's South East Region, should you want a copy it is now marketed by Precision Software worldwide via the Commodore dealer network. It will thus compete directly with Dataview's Wordcraft and Wordpro, now distributed by Wego Computers. The price is substantially greater - more anon.

A new machine at the Hanover Fair was the BX256, a CBM 256 with an Intel 8088 16-bit processor chip supporting CP/M-86 alongside the normal 6509 processor. An optional Z80 card is offered for the 8-bit enthusiast.

The monthly sales of the VIC-20 are currently around 6,700 per month and although Sinclair's ZX series go at 20,000/month, Commodore leads with 2.6 million pounds in the financial charts.

--o0o--

ATARI VERSUS COMMODORE

Just as the Newsletter closed for press, it was announced that Atari are wielding the legal big stick at Commodore over copyright claims. Atari claim that the VIC-20 game called Jelly Monsters is a copyright violation of their Pac-Man game. In court were a VIC-20 and an Atari 800 set up to demonstrate both games to the judge....

--o0o--

DISK TO TAPE SAVE (ARROW FORMAT)

By Brian Grainger.

As promised in the last Newsletter I have modified my disk to tape routine to work with the ARROW chip. It needs an ARROW chip or relocated code to work. With this program a complete disk of programs can be saved in less than 13 minutes! I used relocated ARROW code for the version below but I have included instructions on how to change it for the various chips.

I repeat my warning in the last issue that it is up to the user to ensure the disk does not contain programs that are too long for the available RAM space.

First some changes to the original program have to be made as follows:

change line 155 to:

```
155 PRINT"<10spaces>IN ARROW FORMAT."
```

change line 470 to

```
470 RESTORE:FORA=33768-IT033767:READB:POKEA,B:NEXT
```

change line 2740 to:

```
2740 IFLS=0ANDHS=4THENLS=1: REM NEEDED FOR ARROW
```

finally change line 30270 to:

```
30270 DATA 208,202      :REM BNE LAB1B
```

Now replace lines 19980-20100 in the original program with the lines below:

```
19980 :
19981 REM **********************
19982 REM * ARROW SPECIFIC M/C *
19983 REM **********************
19984 :
19990 REM ********************************************
      ******************
19991 REM *      FOR USE WITH 'ARROW' IN A BASIC2 OR
      BASIC4 MACHINE     *
19992 REM ********************************************
      ******************
19993 REM * HUNT YOUR 'ARROW' FOR THE BYTES A5 B6 8D B4
      02. LOOK AT THE *
```

```
19994 REM * 3 BYTES PRIOR TO THE VALUE GIVEN AND YOU WILL
      SEE JSR $ABCD *
19995 REM * DISASSEMBLE 'ARROW' FROM THE VALUE GIVEN AND
      THE 3 BYTES    *
19996 REM * IMMEDIATELY BEFORE THE FIRST 'RTS' WILL GIVE
      JSR $EFGH. USE *
19997 REM * JSR $ABCD IN LINE 20240 AND JSR $EFGH IN LINE
      20420          *
19998 REM ***********************************************
      ******************
19999 :
20000 DATA 165,209      :REM LDA $D1       (START)
20010 DATA 201,16       :REM CMP #$10
20020 DATA 208,2        :REM BNE LAB1
20030 DATA 169,15       :REM LDA #$0F
20040 DATA 168          :REM TAY           (LAB1)
20050 DATA 169,32       :REM LDA #$20
20060 DATA 153,160,2    :REM STA $02A0,Y
20070 DATA 136          :REM DEY           (LAB1A)
20080 DATA 177,218      :REM LDA ($DA),Y
20090 DATA 153,160,2    :REM STA $02A0,Y
20100 DATA 152          :REM TYA
20110 DATA 208,247      :REM BNE LAB1A
20120 DATA 164,209      :REM LDY $D1
20130 DATA 177,218      :REM LDA ($DA),Y
20140 DATA 141,176,2    :REM STA $02B0
20150 DATA 133,185      :REM STA $B9
20160 DATA 200          :REM INY
20170 DATA 177,218      :REM LDA ($DA),Y
20180 DATA 141,177,2    :REM STA $02B1
20190 DATA 133,186      :REM STA $BA
20200 DATA 165,201      :REM LDA $C9
20210 DATA 141,178,2    :REM STA $02B2
20220 DATA 165,202      :REM LDA $CA
20230 DATA 141,179,2    :REM STA $02B3
20240 DATA 32,14,124    :REM JSR $7C0E
20250 DATA 165,182      :REM LDA $B6
20260 DATA 141,180,2    :REM STA $02B4
20270 DATA 165,183      :REM LDA $B7
20280 DATA 141,181,2    :REM STA $02B5
20290 DATA 169,160      :REM LDA #$A0
20300 DATA 133,136      :REM STA $88
```

```
20310 DATA 169,2        :REM LDA #$02
20320 DATA 133,137      :REM STA $89
20330 DATA 133,139      :REM STA $8B
20340 DATA 169,183      :REM LDA #$B7
20350 DATA 133,138      :REM STA $8A
20360 DATA 169,60       :REM LDA #$3C
20370 DATA 141,182,2    :REM STA $02B6
20380 DATA 169,83       :REM LDA #$53
20390 DATA 133,182      :REM STA $B6
20400 DATA 169,1        :REM LDA #$01
20410 DATA 133,180      :REM STA $B4
20420 DATA 32,87,121    :REM JSR $7957
20430 DATA 76,180,131   :REM JMP $83B4
20440 DATA 208,158      :REM BNE START    (LAB1B)
```

--o0o--

REVIEW

COMPUTE!'s First Book of PET/CBM $12.95
COMPUTE! Books, 625, Fulton St., PO Box5406
Greensboro., NC 27403, USA.

Some of you may from time to time have seen the magazine 'COMPUTE!'. Although an excellent magazine, in the UK the price can be rather high, especially as the 6502 version of the magazine is about 40% Apple-related, 40% PET/CBM-related and the remainder Atari, Ohio, et al. The Commodore interest articles come largely from Jim Butterfield, and other contributors supplying material to an advanced standard. It is this material which has been collated and published as 'COMPUTE!'s First Book of PET/CBM'.

The book comprises about 250 pages spiral bound in an odd size, 8" X 6". The articles selected for reprinting have been collated under six chapter headings. Following the introduction, chapter one is entitled 'Getting Started' and for the new-comer includes a potted history of Commodore along with some basic principles such as explaining the use of Microsoft BASIC tokens.

Chapter two is 'Programmer's Corner' and covers articles that provide help, hints and tips. Examples are sort routines, uncrashing from machine code errors and using GET. The more advanced stuff is covered in chapter three - Beyond the BASICs. This chapter contains ideas which are useful and useable, but which require more time and knowledge than perhaps one could spare.

Graphics is the heading for chapter four and includes a screen dump to printer article. Chapter five is my favourite and covers the subject of utilities. Included are Jim Butterfield's Cross reference program (now in the public domain and the ICPUG software library), Brett Butler's Trace and an article on Multi-tasking on the PET (!). Chapter six contains two articles on communications from Jim Butterfield.

The book concludes with Appendices (memory maps and ROM routine entry points) and an index. The memory maps are not as detailed as in the 'Compendium' or Ray West's book (p56) but no doubt will come in handy.

I went to some lengths to establish whether any errors which may have been in the original articles had been corrected before reprinting. After much searching of text I did find one correction and on the strength of this, assume it is general. The odd typo still remains, eg Superman instead of Supermon ! Do note, however, the material reproduced comes from COMPUTE! of 1980 so that some of it is a little dated, especially as the US appear to be behind Europe on Commodore matters. For example, references to upgrading generally mean from BASIC1.0 to BASIC2.0 although BASIC4 is covered. I would have liked to have seen with each article the issue in which the article originally appeared, but this is just a personal preference. By implication from the title, subsequent material may well be reproduced at some future date. If you haven't seen COMPUTE!, you don't know what you're missing; what's more, you need this book.

R.D.G.

--oOo--

Printed and distributed by Richardson Printing Ltd., Unit 23, Colville Road Works, Colville Road, Oulton Broad, Lowestoft, Suffolk NR33 9QS. *Telephone: (0502) 67029*