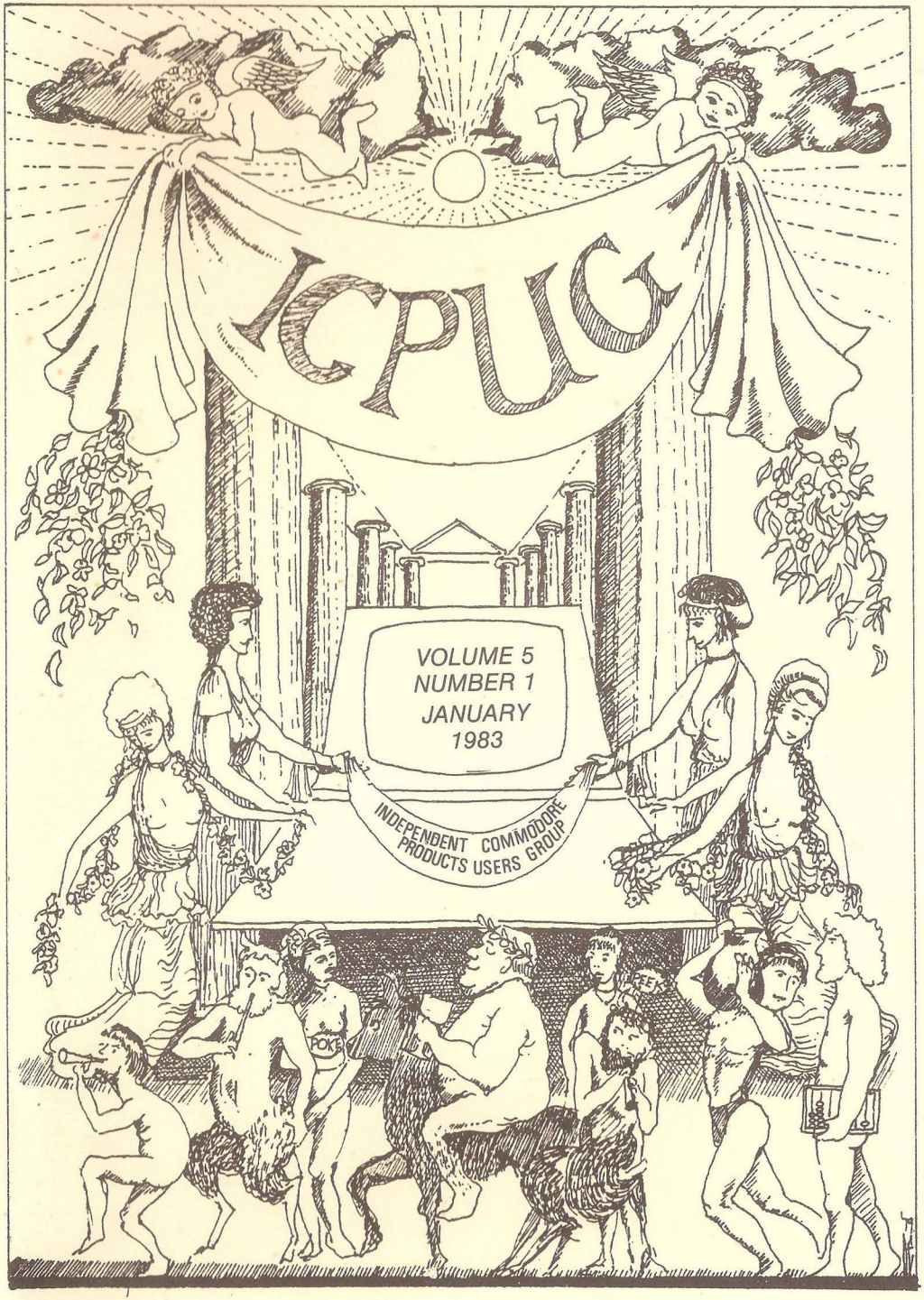

ICPUG
VOLUME 5
NUMBER 1
JANUARY
1983
INDEPENDENT COMMODORE
PRODUCTS USERS GROUP
POKE
PEEK

HONORARY NATIONAL OFFICIALS

Chairman:	Wing Cdr. Mick Ryan 164 Chesterfield Drive, Riverhead, Sevenoaks, Kent TN13 2EH Telephone: Sevenoaks (0732) 453530
Technical Queries Secretary:	Jim Tierney 11 Collison Place, Tenterden, Kent TN30 7BU Telephone: 058-06 2711
Regional Co-ordinator:	Terry Devereux 32 Windmill Lane, Southall, Middlesex UB2 4ND
Treasurer:	Joseph Gabbott
Software Librarian:	Bob Wood 13 Bowland Crescent, Ward Green, Barnsley, South Yorks S70 5JP Telephone: (0246) 811585 (work) (0226) 85084 (home)
Membership Secretary:	Jack Cohen 30 Brancaster Road, Newbury Park, Ilford, Essex IG2 7EP Telephone: 01-597 1229
Editor:	Ron Geere 109 York Road, Farnborough, Hants GU14 6NQ
VIC Co-ordinator:	Mike Todd 27 Nursery Gardens, Lodgefield, Welwyn Garden City, Herts AL7 1SF
Discounts Officer:	John Bickerstaff 45 Brookscroft, Linton Glade Croydon CR0 9NA Telephone: 01-651 5436
Assistant Editor:	Tom Cranstoun

INDEPENDENT COMMODORE PRODUCTS USERS GROUP

Vol 5 No. 1 **Newsletter** Jan 1983

Europe's first independent magazine for PET users

Page	Contents

EDITOR'S NOTEBOOK

As I write this, Christmas has yet to come and computers are being advertised on television. Sales outstrip supply and a number of suppliers have closed their order books, Commodore included. UK stocks of the best-selling Vic-20 expired and Commodore estimated that in the three weeks prior to Christmas they could have sold a further 17,000 machines. UK Vics are made in West Germany. Incredibly, Vic sales have topped the million mark. Commodore are again considering UK manufacturing after closure of the short-lived Eaglescliffe factory, owing to the capacity of the German plant being extended to its limit and the huge increase in UK turnover.

Commodore's UK enterprise represented over 25% of the year's sales and the proportion of Commodore's contribution to world-wide micro-computer sales went up to 75%. When I first bought a computer, over four years ago, I couldn't help but wonder if I was about to back a loser. I could equally have opted for Apple or Tandy, for in those days one was limited for choice and there was little prior knowledge upon which to base a decision. I now have that feeling of satisfaction that comes from the knowledge of having invested wisely.

During 1983 we could well see the introduction of a number of useful products at 'Sinclair prices'. For below £ 70, expect to see the introduction of a modem with ancilliaries and a digitizer at 'affordable' prices. Perhaps then 'information technology' will arrive, after all, when you sit down and tot up the cost of getting involved in Prestel, a lot of enthusiasm is necessary to justify it.

Sign on an office wall: "Yesterday this was an Office of the Future, however there has been a power failure and..

--oOo--

A BUG IN VAL
(or WHEN IS A BUG NOT A BUG?)

By Mike Todd.

When does a quirk of the machine become a bug? That's often a very difficult question to answer. In this article I'm describing what some would call a bug, while others would say that it's not - it's just an oddity of the PET.

The argument against calling it a bug is that it doesn't adversely affect the operation of the PET and requires no action to put it right. On the other hand, it is an oversight in the writing of the BASIC interpreter - so maybe it is a bug after all! I'll leave you to decide.

If you've got a Commodore computer that has its screen RAM following immediately after the BASIC RAM space (e.g. a 32K PET or unexpanded Vic) then type the following:

```
10 A = VAL (TI$) : GOTO 10
```

With a Vic, you'll need the following to see the effect:

```
5  POKE 36879,11
```

RUN the program and you should see a flickering "@" in the top left hand corner of the screen. The top left hand character is constantly being changed to "@" and then put back to what it was. I'm told it may not happen with BASIC 4.

If you just want to know that it happens and that it's not too disastrous, then there's no need to read on. If you do want to know, here goes:

The "bug" is in the section of the interpreter which handles the VAL function. When a function like VAL is encountered, the expression in the brackets is first evaluated - in this case the string associated with TI$ is set up.

The resulting sequence of digits now needs to be converted to a floating point number and, as the operation is exactly the same as handling a string of digits during READ or INPUT, the same routine which converts a string into floating point (known as the FIN routine) is used.

This reads and converts the ASCII digits until a non-numeric terminator character is encountered. Amongst other things this could be a comma, a colon or a zero byte depending on circumstances and the VAL routine puts a zero byte terminator at the end of the string to be converted.

Unfortunately, VAL just whacks this into the byte following the string, over-writing whatever was there first. To stop other strings being corrupted by this, VAL sensibly keeps a note of the original value of this byte and restores it after the conversion has been done.

Going back to our example using TI$, the six characters of TI$ will be placed as the last six characters in RAM (in locations 32762-32767 on a 32K PET, or 7674-7679 on an unexpanded Vic) and then the terminator is tagged on - and goes into location 32768 on the PET or 7680 on the Vic.

This location is the first character on the screen, so a zero is effectively POKEd onto the screen, and this results in the "@" appearing. As I've already said, the VAL routine restores the original byte which results in the "@" simply flashing on the screen.

Because the string TI$ is only needed when it is accessed, it doesn't become a "permanent fixture" in the string space and so the next time it is set up it appears in the same place. So does the terminator, so we get a constantly flickering "@".

It can only occur when a VAL is executed on a string at the top of RAM and so it is rarely (if ever) noticed. It's just the exceptional condition of accessing TI$ constantly at the start of the program that makes it occur so frequently.

If a computed string already exists in RAM when TI$ is computed then TI$ will no longer be at the top and the "@" will not appear. For instance if a line 1 is included such as A$="**" + "**", then it won't occur, since the computed string "**" is built into RAM and stops TI$ being put at the top.

So there it is - is it a bug or not? I would say that it is, although not at all serious, especially considering how rarely it occurs. However, I can see one very special condition under which VAL would return an incorrect value. This requires the terminator to be written into a byte which has no RAM in it (eg the string being right at the top of RAM on an 8K PET) and the "noise" value of the byte into which the terminator is written happens to be an ASCII digit. In this instance, VAL would include such "noise" bytes and it's all very unlikely indeed.

--oOo--

(Actual Size)

AND ANOTHER BUG .....

The arithmetic routines also include a (minor?) bug which shows up when using the INT function. This has been known for a long time, but I thought I'd re-iterate it while on the subject of bugs. Try the following (PETs and Vics):

```
PRINT INT(10 * .1) ; INT(.1 * 10)
```

You will get the results 0 and 1 - yet, of course, they should both be the same since 10*.1=1 and .1*10=1

The reasons are complex but are tied up with a failure of the MICROSOFT interpreter to round the result of the expression before taking the INT of it. A forced "round" can be made by putting a dummy addition in such as:

```
PRINT INT(10 * .1 + 0)
```

--oOo--

MX-80 TIPS

By Barry Biddles

You have just printed a page, and torn it off. Now you want to print something else, you find that you either have to use friction feed or waste a sheet getting to the next top-of-page. Wind the paper down, manually, until it is about to come off the sprockets. At this point, not before, engage friction feed by pushing the release lever back. Now wind the paper down further, until you will be able to print as close to the top of page as you require. The first few lines are printed on friction, and you will find that the paper enters the sprocket guard and engages perfectly. As soon as possible after this, release the friction, otherwise the paper may eventually ruck or tear.

A 12" steel rule may be placed, on its edge, behind the two guides over the sprocket wheels of an MX-80F/T. It will fit securely in front of the two tractor feed removal levers, and forms a quickly applied tearing bar.

If you are handy with a file, and don't care about guarantees, no doubt you could make up a permanent fitting along these lines. For the rest of us, it is possible to obtain a replacement lid looking rather like the one on the MX 70, which incorporates a tear edge.

If you have one of those interfaces which let the last disk command through to your printer, and spoil the top of every file with rubbish like O:FILENAME,P,W, keep a piece of scrap paper handy. Pull the roller bar back and insert the scrap paper behind the print head and down as far as possible. Hold it with one hand while typing <RVS><O><P>, then <STOP>, with the other. This will clear the interface. Then you only have to worry about the top line of every LINKED file...

--oOo--

FORTH COLUMN

By Ron Geere

Forth is one of those languages that you either love or hate (well there's no point in being indifferent, is there ?). In this series I will not be attempting to 'convert' you to become an adherent of the language, but will set out to describe and examine some of its features so you may make your own judgement. With some user feedback, I may be able to assist in any Forth-related problems.

New-comers start here....

A computer language is a means for the user to communicate with the computer. Most computer languages have a fixed set of words, or instructions, which must conform to rigid grammar, or syntax. In the english language, when you want to describe something completely new, one has to invent a word for it and then define that word in terms of existing words. The definition is kept in a dictionary. So it is with Forth.

In Forth a word is anything between spaces. This means that frequently used words can be made short, often only one character. We are used to characters such as + or * but in Forth ! ' and . among others have special significance.

One's own words can, and should be descriptive of the task in hand. In Forth there may be several dictionaries so that one may have the same word with a different meaning in a different context as in english. For example, 'glasses' means different things to an optician and a publican. Long names can be used, (up to 31 characters) they are stored only once. Some dictionaries store only the first three characters plus the length.

What is Forth ?

Forth was invented by Charles H. Moore in the early '70's. In the early days it received little exposure. As a result the language had time to mature and, unlike BASIC, few 'dialects' exist. Since Forth is extensible, a required minimum standard word set exists upon which the user can build.

Forth is many things. It is a high-level language, an operating system, a list of words and definitions. It is both compiled and interpreted. It is interactive; it is fast, easy to test, quick to debug and test. The final result is not always easy to read, but enforces programmer to comment carefully. Forth is inherently a structured language.

Let us have a simple example. If one had an application to control a lamp, one could enter at the keyboard LAMP ON or LAMP OFF to control a bit at the user port. Now these three words must have been previously defined. Suppose LAMP stored the address of the port on the stack and ON were defined to put a '1' in that address. Similarly OFF puts a '0'. Thus a plain english statement LAMP ON can be made to control a lamp.

How does one define a word in the first place. In our example we could enter:-

```
: LIGHT LAMP ON ;
```

This expression will create, or compile, a new word LIGHT defined as LAMP ON so that on entering the word LIGHT it would be interpreted by searching in the dictionary and executing its definition. The colon ':' is Forth's way of saying 'make the following entry in the dictionary'. The semi-colon marks the end of the definition.

Mention was made of the stack. In fact Forth uses two stacks, the parameter stack, where values are stored while being passed from word to word, and the return stack which is used by the interpreter to store pointers to other words.

Forth uses what is called post-fix, or reverse-Polish notation for it mathematics, as in some calculators, e.g. 2 3 + to add 2 and 3. When a number is entered, it is put on the parameter stack. In the example, first 2, then 3 is pushed onto the stack. As mentioned before '+' is a Forth word. Its definition causes it to remove the top two numbers from the stack and replace them with their sum. In BASIC this is like saying X=2+3 and we need to PRINT X to know the result. The simple Forth word '.' prints the number corresponding to the value on the stack followed by

a space. In so doing, it deletes the number from the stack. Forth then outputs the prompt 'ok' to show it has done.

The dictionary is technically indirect threaded code. This means it is a list of addresses of the words comprising a definition. Each definition in the dictionary contains a pointer, or link, to the next word. Since words are defined in terms of other words one's program is eventually defined in terms of a single word, the program name. This is often the last word entered in the dictionary, so the search for it is short and fast. The other words in it's definition have already been found during compilation and entered as an address - no more searching required. The address list is executed similar to a sequence of assembler subroutine calls, e.g.

```
JSR ADDRESS1
JSR ADDRESS2
JSR ADDRESS3
```

except that in Forth the JSR (=GOSUB in BASIC) is not required, so that only 2/3 of the memory is used. Forth uses little memory in comparison to its power.

Resources:
If this introduction has wetted your appetite the following book is generally recommended: Starting Forth by Leo Brodie. It is available in the UK from Computer Solutions Ltd., Treway House, Hanworth Lane, Chertsey, Surrey, KT16 9LA for £ 15.60 as a paperback, or £ 19.10 hardback, inclusive of post & packing.

If you're spending money, various implentations of Forth may be obtained, but watch the price. PolyForth as its name implies is multi-user and costs. I don't think it is implemented on any Commodore machine. PET-Forth and Vic-Forth are available from Hampshire Data Systems, Unit 4, Lynx Estate, Yeovil, Somerset as a Vic ROM-pack or for PET on disk, both priced at £ 59.95 including VAT. Audiogenic supply a VicForth (VP076) at £ 24.95 including VAT and p & p.

Forth is also available from Intelligent Artifacts, and Datatronic AB. The latter's version is pricey, but comprehensive. FullForth+ comes from IDPC Co., and a value-for-money implementation is available from Supersoft. This version is a full FIG-Forth model with all the Forth-79 Standard extensions, complemented by an 80-page manual. Supersoft are at Winchester House, Canning Road, Wealdstone, Harrow, HA3 7SJ. Tel: 01-861 1166.

The Forth Interest Group provide Fig-Forth and this is a public-domain product. I hope to implement this on the PET/CBM and ultimately the Vic-20/C-64 and place it in the ICPUG Software Library in due course. (c) ICPUG 1983

R.D.G.

--o0o--

ROUND THE REGIONS

The Slough/Berks Region had a successful autumn and have settled down to regular meeting dates (if not venues). Second Thursdays of the month are the appointed time and the venues are chosen from Slough College, Crane Packing, Langley College and The Printers Devil [I've met him - Ed.], (Stoke Road, Slough). The club room of The Printers Devil has proved most convivial. The October meeting saw Commodore's Steve Beats put the new 64 through its paces, and those that missed it had the chance of an action replay at the November meeting of the North Hampshire Region. Slough's December meeting saw Compsoft's DMS package in action. Future meetings are to include a look at the 500 and 700 series machines and a visit from Graham Sullivan to talk about micros in education.

A new regional group has been formed at Chelmsford with a dominant Vic-20 interest. Meetings are on the first and third Tuesdays of each month at 7p.m. Contact for further details is Tony Surridge, 97, Shelley Road, Chelmsford, Essex. Tel: 0245 81878 (evenings).

--o0o--

VIC COLUMN

By Mike Todd.

We start the first Vic column of the New Year with what appears to be a Vic bug - but isn't really.

When accessing external output devices (in other words a printer or similar) it is not possible to use TAB(X) or commas to format the output. The simple reason is that the Vic uses the screen parameters to do the necessary arithmetic to calculate the next print position and so it is unlikely that the printer will respond correctly.

It's not really a bug, it's more a limitation (!!) because to provide the facility would require keeping a track of where the cursor is on all output devices. This is a virtually impossible task, especially considering that the Vic would have to know how the device handled control characters and so on since it needs to keep a count of printed characters only.

If you do decide to use TAB(X), or SPC(X) in a PRINT# command - don't have it as the first parameter after PRINT# as there's a bug which will produce a SYNTAX ERROR as follows:

```
10 OPEN 3,3
20 PRINT#3,TAB(10);"TESTSTRING"
```

This will not work, producing a SYNTAX ERROR in line 20, and would also fail if the TAB(10) were SPC(10). It only fails if this is the first item after the comma.

The problem is only likely to occur infrequently and there is a simple fix - change the line to:

```
20 PRINT#3,"";TAB(10);"TESTSTRING"
```

The null string at the start is ignored, but is sufficient to allow the syntax to work.

The bug is basically due to the fact that the routine which sets up the output file device is slightly different in the Vic than in the PET and it no longer preserves the accumulator and other registers on exit.

MATTERS ARISING

Now it's time to correct the bugs in the last Vic column! It seems to be the case that I spot the error just too late to put it right - maybe a function of the speed at which I'm having to work nowadays. I might inject at this point that, as well as reading me (and seeing my picture!) in VIC COMPUTING every other month, you'll be able to read me in PRACTICAL COMPUTING as from the February issue where I'm editing the PET/VIC OPEN FILE.

Anyway, enough of the personal plugs and down to the first problem on page 298 of the last Newsletter.

I went to some length to explain that there is a bug in the Vic which causes "DEVICE NOT PRESENT" error instead of "FILE NOT FOUND" errors if you try to OPEN a read file beyond an end-of-tape marker. I also said that the bug doesn't exist if you LOAD beyond the marker. Well, it appears that it does!

Also on page 298, I mentioned that the +0K re-configuration of the Vic caused problems when using the SUPER EXPANDER and I couldn't work out why. Well, as soon as the copy had gone to the printers I realised.

The SUPER EXPANDER automatically reconfigures the memory map of the Vic as soon as a GRAPHIC 1, 2 or 3 is issued. It puts the screen RAM back to 7680 ($1E00) where it would be on an unexpanded (or +3K) Vic and puts the character generator down at 4096 ($1000).

This means that, in a straightforward Vic with only +3K expansion nothing has really changed other than the reduction in RAM. However, if you've got 8K or more of RAM expansion, your BASIC program would have started at 4608 ($1200) and the

screen at 4096 ($1000). So the SUPER EXPANDER moves the BASIC program out the way at 8192 ($2000) and puts the screen RAM back to where it was.

This is fine if you've configured your Vic to have +8K or more of RAM, but if you've tried to configure it to have +0K RAM (so that the screen RAM is moved), the SUPER EXPANDER puts it back again and tries to put the BASIC program into empty space. The pointers also tell it there's no more RAM left - so you get an OUT OF MEMORY ERROR.

Aren't you glad you know now!

Finally, there's an error on page 310 in the discussion of the shift key flag (SHFFLG) - PEEK(653) would be 3 (and not 7) if the CTRL and CBM keys are pressed together.

SUPER EXPANDER - GRAPHIC 4

Having said that, with 8K or more of RAM expansion, the SUPER EXPANDER moves the BASIC program up out of the way and then moves the screen and character generator about, there appears to be a fifth GRAPHIC command (GRAPHIC 4) which will put everything back again.

Note that the reconfiguration only occurs when GRAPHIC 1,2 or 3 is called whilst in GRAPHIC 0 mode.

I've not had a lot of time to investigate this one - so more next time, I hope.

LIGHTPENS

I've been playing with the STACK lightpen recently and thought you might like to know my findings.

First of all, the lightpen seems very expensive at #25.00 - mind you, it is on a flexible cable and comes with a simple, but reasonably good, memory game to be played with the lightpen.

As I approached the TV with the pen, tension was placed on the curly cord and I found that the large plug on the end of the cable kept pulling out of the Vic's socket. I also found the lightpen was a little erratic in use. The reason is clear if you take a very close look at your colour TV screen.

The screen is covered by thousands of red, green and blue dots or stripes which make up the colour picture. But all the dots have a dark region between them and sometimes the lightpen is placed in one of these dark areas (the light sensitive tip is very small) and so doesn't trigger correctly. I found that moving the pen about 1-2mm from the screen made the world of difference, and I'm sure it would be possible to put a small collar over the end of the pen to keep it at this distance.

The lightpen also contains a touch sensitive switch which is used to communicate to the program that the spot on the screen has been selected and that the program should then go and retrieve the lightpen's co-ordinates.

The co-ordinates of the pen are simply read by PEEK(36870) for the horizontal position and PEEK(36871) for the vertical. The touch switch is detected by looking at bit 4 (the joy2 line) of location 37137; if PEEK(37137) AND 16 is zero then the switch is being touched. It is, of course, possible to use the RJOY and RPEN functions in the SUPER EXPANDER to get these values.

The range of values returned is theoretically 0-255, but in practice it is considerably narrower. Whatever value is returned will need to be scaled appropriately and this is probably best done by starting any lightpen program with a simple "calibration" test where the pen is placed at two known positions on the screen.

ANOTHER BUG?

Assuming that you've got a basic Vic or one with only 3K of expansion, try the following:

```
10 POKE 36879,11
20 A = VAL(TI$) : GOTO20
```

Line 10 just sets up a black screen so that we can see what's going on, and line 20 shows up the problem - you should get a flashing "@" in the top left hand corner.

The problem exists on PETs too and I've written an explanation elsewhere in the Newsletter.

ICPUG SERVICES

If you've seen the list of national officers of the group in the last Newsletter you'll realise that there are many services offered to members and many of the officials responsible have their addresses and phone numbers inside the front cover.

If you wish to contact any of the other officials whose addresses aren't published (on the subject of publicity, PRESTEL, COMAL, projects for instance) then any of the other officials would be happy to forward your letters. You'll also find that you get a far faster response if you enclose a stamped and addressed envelope.

Amongst the officials who are listed, Terry Devereux is the man to contact if you want to set up a regional group or want to know how to contact your nearest group.

Bob Wood is the software man and has quite a few Vic programs as mentioned on page 323. If you send him a S.A.E. he'll send you a list of what's available. He'll also accept any of your contributions.

Technical queries should go first to Jim Tierney who is our Technical Queries secretary and he'll pass them on to the relevant expert. (But don't forget the problems page!)

Jack Cohen handles membership and subscriptions and also provides the mailing list for the Newsletter distribution.

John Bickerstaff has the low down on discounts and if you're after anything for your Vic (or PET for that matter) contact him first in writing explaining what you want and he'll get back to you (probably by return of post) with the good news.

ADVENTURE SWAPS

Anyone who has played the Commodore Adventure games will realise how addictive they can be, but once you've completed a game (which could take days, weeks or even months!), it's unlikely that you'll want to play it again for some time.

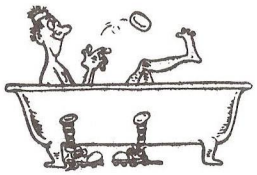
Brian Roberts of Luton has suggested that there may be many interested in swapping their "used" Adventure game cartridges with others and I think that this is a splendid idea.

Brian has agreed to act as a "clearing house" for the swaps and I would suggest that anyone who wants to do a swap should write to him giving details of what cartridges are being offered and what is wanted, but PLEASE don't send the cartridges at this stage.

Once he's got swaps matched up, he'll get in touch to arrange the swap, provided that you've also sent a stamped and addressed envelope.

His address is:

7, Cowdray Close,
Luton,
BEDFORDSHIRE



Incidentally, Brian has also come across what appears to be a bug in the PIRATES COVE game - it's not too serious and is related to stamps and a box. I won't say any more in case it spoils your enjoyment.

I've been playing PIRATES COVE over the last few days

and am totally hooked! It took me a long time to find out how to stop drowning in the lagoon and Brian had to give me a hint as to where to find the keys to open the doors. I wouldn't have thought of looking under the .... But that would be telling.

INPUT PROMPTS

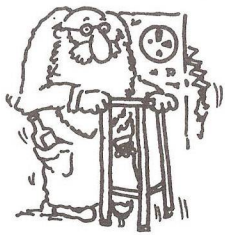
A while back, I mentioned that, under certain circumstances, the INPUT prompt can sometimes be included in the input string. For instance, try the following:

```
10 PRINT "PROMPT";
20 INPUT A$
```

RUN it, and when you see the "PROMPT? " and flashing cursor, move the cursor up and then down again, type something like TEST and press RETURN.

If you now examine A$ (type PRINT A$) you will see that it is contains the characters "PROMPT? TEST", which is likely to cause problems.

There are several things that cause this, and there is no truly easy way around the problem. However, the following may at least provide one solution.



```
10 PR$ = "PROMPT"
20 PRINT PR$;
30 INPUT A$
40 P=LEN(PR$)
50 IF LEFT$(A$,P)=PR$ THEN A$=MID$(A$,P+3)
```

This has the effect of checking A$ to see if the prompt string was included, and if so, only the characters after the prompt and question mark are included in A$. If it is necessary to input a numeric variable, use the same technique but add an extra line at the end A=VAL(A$) which will set A to the numeric value.

If you know that you only require, say, the first 5 characters of the input string, then you could add an extra line A$=LEFT$(A$,5) which will ignore anything after the 5th character. This could be used to good effect if there are any "spurious" characters on the same line as the input prompt.

Of course, you could always "simulate" INPUT using GET, although the programming required is much more complex but much more versatile too. A very simple method is:

```
10 A$=""
20 GET X$ : IF X$="" THEN 20
30 PRINT X$;
40 IF X$ <> CHR$(13) THEN A$=A$+X$ : GOTO 20
```

Line 10 clears the string which will be built up, line 20 gets single characters from the keyboard, line 30 echoes the characters so that you can see what you're typing and line 30 builds up A$ until the RETURN key (equivalent to CHR$(13)) is pressed.

There's nothing to stop you using a different character to terminate the input, or improving line 30 to provide a pseudo cursor. It is also possible to "trap" cursor controls so that the user cannot move the cursor around on the screen and it's even possible to restrict the input to only numbers, letters or specific combinations of letters and numbers.

AND FINALLY

I'm sorry that the Vic Column is a bit shorter than usual, but to make up for it there's a review of the BUTI programmer's aid elsewhere in the Newsletter. It's by Richard Allen, to whom I must apologise for not publishing it sooner.

If you've any comments on Vic products, why not write a review - but remember to make it as objective as possible and to cover all its features, not just the ones you like.

--oOo--

COMMODORE COLUMN

Lavatory Humour!

No this page isn't degenerating; just what some people get up to with their PETs. Reckitt & Colman have hooked a defenceless PET up to a pneumatic controller which, in turn, causes a whole room full of flush toilets to be activated at once. We have no idea why they are subjecting the computer to this work as the promotional material from IT82 offices only tells us that it is being done!!

New 8000 Series PETs Unleashed.

The 8000 series machines (8032/8096) have been released in a new case; that of the 500/700. This incorporates a tilt/swivel screen, seperate keyboard and the possibility of built-in disk drives - all complying with the new EEC regulations. Dealers appear to be excited about the new case, which will offer a few more years life for the product, as customers found the old style case 'dated'.

The following changes have been made during recasing:

1 IEEE-488 now on a standard connector.
2 User Port now on an IEEE-488 connector.
3 The second cassette port is no longer available.

Prices for these 'new' machines:

CBM 8032 - £ 995.00 excl VAT.
CBM 8096 - £ 1195.00 excl VAT.

It is possible that Commodore will make a 'conversion kit' available, for around £ 100.00, allowing oldies to be in step with the fashion.

Smalltalk.

Commodore intends to make available the XEROX developed SMALLTALK operating system/language, which combines high resolution graphics with a sophisticated editor, on the 700 in the near future. Datalink magazine feel that any implementation of small talk will have to be a cut down version as the screen resolution of the 700 is nothing like that used by XEROX.

Stacktalk.

Stack Computer Services Ltd. have produced a 40/80-column card to allow Vic-20 users write programs in a 40-column or 80-column format - without losing any of Vic's features. The card costs £115.00 and is being offered exclusively to VICSOFT (Commodore's own club) members.

Stack's card works by generating a second display independently of Vic's own. The Vic screen can be used simultaneously; therefore you don't lose colour etc. or the Stack board can be run on its own. It runs the full Vic character set in upper/lower case, graphic symbols and reversed field - it also incorporates the full Vic editing facilities.

The second display is software controlled in character size, line spacing and screen size, so that it is more flexible than it first appears. Simple key combinations allow you to change these parameters at will.

Take a Bow Commodore.

The world's first million selling computer has been announced - The Vic-20.!! I wonder if number 1,000,000 was made in gold!! USA sales top 850,000, European sales 200,000+ (50% of this in the UK).

Disk Desk.

It is possible that your dealer will have the latest Commodore product in stock - that bastion of the electronic office; a desk to put the computer onto. This is actually made by Commodore, in Canada, so if you're looking for matching office furniture or the specification attracts you then pop down to your dealer. The desk has an 'Arborite' top 1220 x 660 x 28 mm, steel legs and frame (don't get caught), adjustable levelling mechanism, ventilated disk compartment, and has a shelf with cable slots. As we go to press the price is unknown.

Teachers Pet.

Commodore have appointed an education advisor, Graham Sullivan, who has taken a one year break from his teaching routine - he was headmaster of Lowbrook County Primary School. Graham's job, as an independent advisor, is to promote educational uses within the company, to make Commodore aware of the needs within schools and, hopefully, for Commodore to try and fulfill those needs. Workshops and seminars are planned; suggestions put forward so far include the production of educational software, information sheets, contact with LEA advisors,and investigating turtle and other control applications. Graham is enthusiastic about the new 64 in terms of school use and the possibility of selling the 64 complete with software and tuition. Most of Graham's time will be spent on the road finding out what teachers want, then trying to convince Commodore to provide it.

T.C.

--oOo--

64K UPGRADE FOR A 2001-SERIES PET

By Nigel Peters

I spent a long time looking for a cheap way of expanding the memory of my 2001-series PET. At Christmas of last year I added another 8K of RAM, but I still needed more, in order to run a number of 32K programs, and, in particular, Superscript. In July of this year I acquired a 64K dynamic RAM card, as used in the Acorn Atom. The card

has its own 6502 processor and it replaces the PET's processor. The RAM was arranged as a single block of 64K, but what I wanted was two blocks of 32K each. To make the blocks, I took the most significant of the card's address lines, (A15), and attached it to a switch, so that I can hold it high or low depending on which block I want. A complication with the memory map is that the PET needs to be able to address ZERO PAGE on each of the two blocks. I got around this by masking out the bottom 1K of each block, and replacing that RAM with 1K from the PET's own on-board RAM.

The only problem that caused any trouble was the PET's clock. The CLOCK IN pin of the 6502 (pin 37) should oscillate at 1MHz, but on the early PETs the frequency is often lower than this. Because the RAM card is dependent on the clock signal, the contents of the RAM are lost if the clock is too slow! I solved this by changing the crystal which controls the clock speed for a more suitable one.

The card produces power supply problems, as my PET is one of those converted American ones in which the original PET transformer has been replaced by a pokey little British one. The extra load is almost all that it can manage! A further difficulty is that the 6502 on the new card is not buffered, and is now required to provide much more output current than originally intended. It gets quite hot, and gives up after about one hour's use, so if I want to run Superscript I have to remove anything that is consuming power unnecessarily, such as BASMON and PLUSDOS, and restrict myself to about an hour only!

I usually only use 32K of my additional RAM, but it is nice to know that I could use 64K if I wanted to! I am planning on putting a 4K buffer above the screen so that I can transfer data between the blocks. I have already replaced the switch with a line to the user port, so that I can change blocks under software control.

--oOo--

VISICALC - PRINTING THE FORMULAE

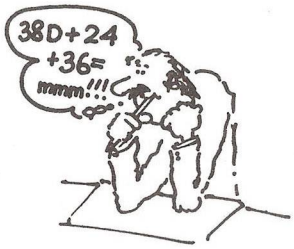
By Brian Grainger

I wonder how many of you have, like me, been under the misconception that one could not print to paper the formulae used on the spreadsheet when using VISICALC. I found this a source of great irritation and certainly thought an additional program was required. As some programs came out for this purpose when VISICALC first appeared this idea stuck.

Just recently our illustrious Chairman, Mike Ryan, pointed me in the direction of page 136 of the VISICALC manual. This article is the result.

The simple way of printing the formulae is to go in the file-saving mode but in response to the filename type ',tp,ca'. This will then print the formula to a Commodore printer. A sample printout is given below.

```
>D4:+B4*C4.....1
>C4:+C3+1.....2
>B4:+B3+1.....3
>A4:"ROW3.....4
>D3:+B3*C3.....5
>C3:+C2+1.....6
>B3:+B2+1.....7
>A3:"ROW2.....8
>D2:+B2*C2.....9
>C2:2.....10
>B2:1.....11
>A2:"ROW1.....12
>D1:/FR"COL3.....13
>C1:/FR"COL2.....14
>B1:/FR"COL1.....15
>A1:/--.....16
/W1.....17
/GOC.....18
/GRA.....19
/XH2.....20
```



```
/GC9.................21
/X>A1:>B1:/TV........22
/X>A1:>D1:;/GC9......23
/X>A1:>C1:/TV........24
>C2:/TH..............25
/X>A1:>D3:/WS........26
```

Let me try and explain the above listing. The spreadsheet which gave the above printout, (note the dots & numbers have been added by me for this explanation), was rather simple. I had a horizontal window bar in operation with titles fixed in both windows - they were fixed differently in each window though. Lines 1-16 are simply the formulae and formats used on each sheet entry. They pretty well follow what is typed except for the delimiter, ':', and the use of '"' to start a label. You will note that one entry is printed per line so if you have a lot of entries you need a good supply of paper. Another point to note is that labels in lower case come out strange when ASCII characters are used in the file name (,ca). This can be resolved by using PET characters instead (,cp) but then everything else comes wrong! Entries are printed in REVERSE order!

I haven't worked out what line 17 means. It always appears after the entry printout on all sheets I have tested. Line 18 indicates the order of calculation (by columns) and line 19 the recalculation mode (automatic). Again these are printed as they would be typed. Line 20 indicates a horizontal window bar has been set at line 2. Note that the 2 is a screen line and NOT a sheet row. This is more apparent when a vertical window bar is set, e.g. /XV12 which indicates a bar at screen column 12. It is impossible to get 12 entries on a 40-column CBM screen with standard width entries.

Everything from the window bar definition until the occurrence of ';' defines parameters for the 1st window. Line 21 gives the column width (9). Line 22 says that the entries from A1..B1 are fixed as titles vertically. Line 23 starts by indicating that the top left of the display for the 1st window is A1 and the cursor is at D1. The ';' now appears so the 1st window definition is complete.

The last item of line 23 indicates that the column width of the 2nd window is 9 as well. It can be different from the first window. Line 24 indicates that entries A1...C1 are titles fixed vertically. Line 25 is new and indicates entries to C2 are titles fixed horizontally. Finally line 26 indicates the top left displayed for the 2nd window is A1 and the cursor is at D3. Finally the windows are synchronised in scrolling. One point to note. If the cursor was to be positioned in the first window a further ';' would occur prior to the synchronised scrolling identification.

That covers most of the possibilities that will occur in practice. If the sheet is less complicated those items not included are omitted from the printout. Actually the print is an exact copy of the keystrokes required to form an exact copy of the spreadsheet. This in fact happens when loading a file. There is an /X command but it has strange effects. That is presumably why the format is in the order that is printed.

To finish this article I must give one word of warning. When one stores a file to the printer this way it becomes a default for all future file storage. One must redefine the disk and PET characters by appending ',td,cp' to the filename. For more explanation of filenames I suggest you read p.136 of the VISICALC manual.

--o0o--

NEWSLETTER BACK NUMBERS

The following are the current rates for ICPUG publications:

		ff
Compendium	UK	2.50
	overseas	3.50
Back issues of Newsletter	UK	1.25
	Europe & Eire	1.75
	outside Europe airmail	3.00
	surface mail elsewhere	1.75

The above may be obtained from the Membership Secretary, Jack Cohen.

--o0o--

TECHNICAL TIPS

Apologies !

I have been taken to task by Harry Broomhall for perpetuating a Myth... There is no such animal as "The-SAVE-with-replace bug".

Harry knows what he is talking about and all I can say, in mitigation, is that that snippet of information came from 'usually reliable sources - CBM'.

To repeat, there is no SAVE-with-replace bug! Personally I have always ignored this 'bug' and used SAVE"@0:file",8 as it is much more convenient than SCRATCH then SAVE.

To counterbalance the plethora of Vic & 64 information now pouring into the Newsletter we present two sections of interest to serious (no not Sirius!) business users; the SuperPET and 9060/9090 hard disk drives.

SuperPET.

Release 1.1 of the Waterloo languages, order code WCS1.1, is now available from your dealer. The package fixes all reported bugs from the original release, enhances the APL interpreter and adds COBOL to the languages available (with 339-page manual and 43 example programs) - all this for only £ 25.00 + VAT. The package also includes 50+ errata pages for the existing manuals.

SuperPET serial port.

This machine has an RS232 25-pin female D-connector port inside it, to gain access you have to undo the two screws at the bottom front of the machine. In RS232 terms the SuperPET is a DATA/TERMINAL. Some devices require only two connexions plus a ground to work - but the SuperPET requires all 9 connexions. If the device at the other end of the RS232, the DATA/SET device, cannot supply all the connexions then the plug at SuperPET's end should be jumpered as follows :-

Links	Pin Number	Name	Description
	1	gnd	Protective ground
	2	TxD	Transmitted Data
	3	RxD	Received Data
to CTS	4	RTS	Request to Send
from RTS	5	CTS	Clear to Send
to DCD	6	DSR	Data Set Ready
	7	gnd	Signal Ground
to DTR	8	DCD	Data Carrier Detect
from DCD	20	DTR	Data Terminal Ready

i.e. Pins 4 & 5 are linked and also pins 6,8 & 20.

This fools the SuperPET into thinking that the RS232 bus is connected properly. The technique can also be used on other RS232 devices.

Warning: Commodore state that Pin 13 ALWAYS has +5 volts on it, you should take care never to connect anything to this pin - or let stray wires touch it.

The SuperPET's serial bus is controlled by a 6551 Asynchronous Communication Interface Adaptor (ACIA for short). This chip features a software controlled baud rate (speed of working) generator; and can be driven in full or half duplex modes. The ACIA is seen by both microprocessors at the same address from $EFF0-$EFF3 (or in decimal 61424-61427) - the following registers being present :

Address	Write Access	Read Access
$EFF0	Fill transmitter	Read Receiver
$EFF1	Reset Chip	Read Status Reg
$EFF2	Command Register	Command Register
$EFF3	Control Register	Control Register

The control register is used to select the operating mode; word length, number of stop bits, baud rate etc.

Control bits	Function	Data (HEX)
7	Number of stop bits	0=1 Stop Bit
		1=2 Stop Bits
6-5	Set word length	0=8 bits
		1=7 bits

2=6 bits
3=5 bits

3-0 Select baud rate
15 different baud rates can be selected for example, the following provide:
06=300
07=600
08=1200
0E=9600

The command register is used to control parity generation/checking, receiver echo and transmit/receive functions:

Command bits	Function	Data (Binary)
7-5	Parity options	xx0= no parity
		001= odd parity, Tx + Rx
		011= even parity, Tx + Rx
		101= Mark parity on Tx, Rx disabled.
		111= Space parity on Tx, Rx disabled.
4	Echo /No echo	0= No echo
		1= Echo for Receive.
3-2	Transmitter control	00= Tx disabled, no Request-to-send (RTS)
		01= Tx enabled, RTS.
		10= Tx disabled,RTS.
		11= Tx disabled,RTS (Transmit BRK).
1	Receiver interrupt enable	0= Interrupt enabled.
		1= Interrupt disabled.
0	Data terminal ready	0= Disable Receiver/Transmitter.
		1= Enable Receiver/Transmitter.

The status Register allows the programmer to interrogate the ACIA, it is a read only register.

Status Bit	Function (1= Error detected, 0= no error)
0	Parity
1	Framing
2	Overrun
3	*Receiver data register full.
4	*Transmitter data register is empty
5	No data carrier.
6	Data set not ready.
7	*Interrupt

*=Not error conditions but neccessary for efficient programming.

9060/9090 Winchester Hard Disk Drives.

The latest information from COMMODORE includes an addenda for the manual of these disks which indicate the tremendous capacity; the HEADER command (or NEW disk command, from DOS support) will take appoximately 55 minutes for a 9060, and 1Hr 45mins for a 9090 to properly format the disk. These drives come fitted with DOS 3.0, in one of two ROM configurations (REV B,REV C), providing a number of changes which were considered essential in the use of hard disks:
The concept of 'cylinders' of data is introduced to CBM equipment for the first time. A cylinder is the same track on all surfaces of a disk drive (9060 having 4 surfaces, 9090-6). This speeds up data access if data is written in cylinders in the first place. Generally programmers will not worry about cylinder access; but its nice to know it's there if an application demands it.
Track numbering starts with track 0, all floppies start at track 1.
A bad sector list; which is an invaluable feature of hard disks, as it is hard to throw away Mbytes of data because one sector has gone down on the disk.
Larger relative files; through the concept of 'Super Side Sectors', which point to groups of side sectors - which in turn point to the data sector(s), it has been possible to arrange a relative file maximum file size of 86, 400 sectors. However,more practical limitations raise their head - there is only (ONLY says he!!) 29,376 sectors on a

9090 drive - the maximum number of relative records in a single file is still 65,536.

P.S. Each of the side sector groups pointed to by the Super Side Sectors is equivalent, in structure, to an entire DOS 2.1/2.5 relative file.

Directory header format, Track 0 - Sector 0.

Byte	Data	Description.
0-1	00-01	Track-sector pointer (TSP) to bad sector list.
2-3	00-255	Identifies DOS 3.0
4-5	76-10	TSP to first directory block.
6-7	76-20	TSP to disk name and ID block.
8-9	01-00	TSP to first BAM block.
10-11	48-49	Not identified.
12-255	00	Not used.

BAM Format.

Byte	Description.
0-1	TSP to next BAM block. ($FFFF = on last block)
2-3	TSP to previous BAM block. ($FFFF = on first block)
4	Lowest track number mapped in this BAM block.
5	Highest track number+1 mapped in this BAM block.
6-255	Bit map of up to 50 tracks, for one recording surface.

T.C.

--o0o--

THOUGHT FOR THE MONTH

The probability of anything happening is in inverse ratio to its desirability.

--o0o--

KEYCHIP A LOW PRICE PET AID.

By Alfred Rose.

KeyChip is a 4K chip which provides a large number of facilities for writing and debugging programs. All these functions are activated by pressing the left shift and one other key, none of them may be used from a BASIC program. It is available for Commodore PET 9" screen models (3000, 4000 and 2000-BASIC 2.0 series).

At a price of fifteen pounds including VAT and postage, it seemed too good to be true. Had the suppliers Wirt Microsytems really broken the price barrier, or just produced a shoddy piece of software? I was pleasantly surprised to find that KeyChip was very well packaged. The thirty page manual was typeset and excellently produced. There were two strips of labels which had to be stuck above the top row of keys to identify their new functions. The labels and the functions on the back cover of the manual were laminated with clear plastic, a welcome innovation to stop the onset of premature greyness. The general standard of presentation in this product puts the producers of exorbitantly priced hardware and software to shame.

KeyChip is compatible with Toolkit and both chips were used simultaneously during all my tests. The manual gives the SYS commands to activate KeyChip in the preferred UD3 socket, although it can be supplied for any free ROM socket.

The most interesting KeyChip function is the ability to scroll a BASIC program in both directions, either at high speed or one line at the time.

Simply pressing "left shift L" will activate the List function, causing the cursor to disappear and the program to be listed from the beginning. Once in the List mode, the screen is a window on the program which will fill all 25 lines of the screen. The program is still, however, quite difficult to read, which is where the List format functions come in. At the press of a key the line numbers can be reversed, or the second BASIC line is indented clear of the line number, or a space is created between BASIC lines or any combination of these functions.

One of the features of PET program listings, are the strange cursor-control symbols, e.g. reverse Q means cursor down. KeyChip can change these to more recognisable symbols. It can also fill spaces within inverted commas with reverse dollar symbols. This makes it possible to count spaces, e.g. when producing column headings.

After scrolling the program, pressing left shift will bring back the cursor and the program can be edited. Then "left shift R" will re-enter the list mode at exactly the same place and the program can be scrolled again. How the KeyChip manages to scroll the program in both directions, while keeping track of all the List formats I can't imagine. It even scrolls BASIC lines which are longer than 80 characters, and labels these "illegal" 3rd lines with a reverse arrow symbol!

In everyday use I found it very helpful to load the disk directories by using the DOS /$ and then, with KeyChip I was able to scroll both directories from the one loading.

As soon as KeyChip is activated all keys will repeat apart from RETURN and RVS/OFF. All "Repeat" parameters can be changed by POKEs, i.e. limiting repeat to the cursor-control keys, or changing the delay before repeat or the repeat rate.

Pressing left shift and one of the labelled keys at the top of the keyboard, accesses a large number of screen editing functions as described briefly below.

If the repeat key is too slow, then two dedicated keys make it possible for the cursor to move in half-screen jumps. This is useful for moving the cursor to the bottom of the screen or to the centre of a line.

There is a function to instantly regain control of the cursor when inverted commas have been entered. Similarly, just pressing left shift will release the cursor when the INSERT key has been used.


commodore
CBM™
Model 3016
computer
CRSR
CRSR
SCROLL
SCROLL
CLOSE
OPEN
LINE 40·80
DEL
DEL
DEL
ENTER
DEL LINE BASIC
DEL LINE BASIC
OFF RVS
SHIFT LOCK
SHIFT
RUN STOP
RETURN
SHIFT
CHAR SET
SAVE
RECALL
SCREEN SAVE
CLR HOME
CRSR
CRSR
INST DEL

The screen can be scrolled in either direction or closed/opened up at any point. The latter is useful when inserting a line in a program.

The screen can be erased below or above the cursor or from one line to any other. Lines can also be erased either left or right of the cursor. There is a similar function for BASIC lines which leaves the line numbers intact so edited lines can be re-entered immediately.

Screensave is a unique function to store up to 10 complete screens in memory. There are many features, such as swapping a stored screen with a current screen, storing any rectangular part of the screen or recalling a stored screen in reverse video. There are so many features that the manual takes six pages to list them all. Screensave is of special interest to users who do not have a printer and have to resort to pen and paper to note down program output. Parts of a program can also be altered while saving the original version in case proposed "improvements" make it worse. It is useful to be able to store small BASIC routines, e.g. to print out certain variables during debugging or for complex disk commands.

The conflict between the slower program with lots of spaces and REMs and the fast incomprehensible program is resolved with a routine to remove REMs and/or spaces.

Up to ten machine code subroutines can be activated at a touch of a button after a few POKEs to tell KeyChip where they are. The program produces a subroutine for "one key" BASIC commands. In fact the ASCII data given produces pictures of the author of KeyChip!

There are a number of screendump programs which print the whole screen on a Commodore printer but unless they are on a chip it, is very often impractical to use them. for example a chess game cannot be interrupted to print the positions of the pieces. The KeyChip Screenprint routine works perfectly with Petchess and Microchess. The chess or checkers program has to be loaded first, but not run until

KeyChip has been initialised. It can then be run normally and any stage of the game printed out.

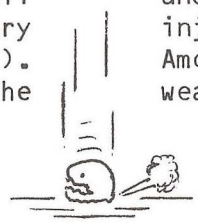
In summary after using KeyChip I consider that it is as useful to programming as the wordprocessor is to typewriting. It is a valuable addition to the PET a big time saver and excellent value. Further details from Wirt microsystems, 12, Alleyn Cres., London, SE21 8BN.

```
10 S=826:N=0:REM S=START OF SUBR.N=NUMBER OF KEY(0-9)
20 S=S-1:H=INT(S/256):L=S-H*256:N=N*2+904:POKEN;H:
   POKEN+1,L
30 FORA=S+1TOS+99:READ B:POKEA,B:IF B=0THEN50
40 NEXT
50 R=S+9:H=INT(R/256):POKE S+4,H
60 DATA169,1,160,1,32,28,202,96
70 REM BELOW MESSAGE IN ASCII-TERMINATNTE WITH 0
80 DATA29,250,184,204,157,157,157,17,213,46,32,46,201,
   157,157,157,157,157,17
90 DATA202,32,221,32,203,157,157,157,17,205,45,206,145,
   145,145,29,29,0
```

--o0o--

ATARI VERSUS COMMODORE

Atari has backed off and withdrawn its application for a temporary injunction against Commodore (Newsletter p283). Among the reasons possible for Atari's move is the weakness in the law over software copyright.



Atari now expect to have a full trial in which they claim that Commodore breached their copyright in Pacman by marketing a game called Jelly Monsters. Atari is already suing Philips over the same game and succeeded in the US. However the US judgement has no legal bearing in the UK.

--o0o--

DOUBLE-DENSITY PLOTTING IN COMAL

By Brian Grainger.

In the September 81 Newsletter I gave an example of using COMAL procedures recursively. In this article I want to show another use for COMAL procedures and functions, namely how to implement new commands. To do so I will write some routines to emulate my BEM1 double density plotting (refer to Newsletter May 81). This introduced two commands, SET(I,J) and RESET (I,J) and 1 function POINT(I,J).

There are two special lines in the routines and they have to be specially prepared. First of all type in the following COMAL program from the keyboard:

```
0010 DATA 32,108,124,225,126,127
0020 DATA 226,251,123,98,255,254
0030 DATA 97,252,236
0040 PROGSTART:=PEEK(23)+256*PEEK(24)
0050 POKESTART:=PROGSTART+152
0060 FOR I:=0 TO 14 DO
0070 READ K
0080 POKE POKESTART+I,K
0090 ENDFOR
0100 END
2020 DDCHARS$:="123456789012345"+CHR$(160)
```


Having entered the above program LIST it to disk by LIST"0:GENLINE.L". Then RUN the program, DEL -100 and RENUM 5020. If you LIST the result you will see a single line 5020.

Now ENTER"GENLINE.L", RUN the program again and DEL -100 again. LIST the result and you will see two identical lines 2020 and 5020.

Add to these two lines the rest of the program as given below. DO NOT OVERWRITE line 2020 or 5020.

```
0001 // written by b.d.grainger (icpug)
0002 // for comal version 0.12
0003 // prog. vers. 1.1 november 1982
0004 //
```

```
0005 // these routines allow double
0006 // density plotting by use of
0007 // set(x,y) and reset(x,y) command
0008 // where 1<=x<=80 and 1<=y<=50.
0009 // screen top left is x=1, y=1.
0010 // point(x,y) will be true if a
0011 // point exists at x,y, otherwise
0012 // it will be false.
0013 //
0014 // lines 2020 and 5020 are special
0015 // do not try to enter them from
0016 // the keyboard
0999 //
1000 proc set(x,y) closed
1010 find(x,y,value,screen,sdchar)
1015 if not (sdchar mod 2↑value-sdchar mod 2↑(value-1)) then
     sdchar:+2↑(value-1)
1020 plot(sdchar,screen)
1030 endproc set
1039 //                                  (c) ICPUG 1983
1040 proc reset(x,y) closed
1050 find(x,y,value,screen,sdchar)
1055 if sdchar mod 2↑value-sdchar mod 2↑(value-1) then
     sdchar:-2↑(value-1)
1060 plot(sdchar,screen)
1070 endproc reset
1079 //
1080 func point(x,y) closed
1090 find(x,y,value,screen,sdchar)
1100 if sdchar mod 2↑value-sdchar mod 2↑(value-1) then
     return 1
1110 return 0
1120 endfunc point
1999 //
2000 proc find(x,y,ref value,ref screen,ref sdchar) closed
2010 dim ddchars$ of 16
2020 ----------AS CREATED ABOVE-------------------------
```

```
2025 x:-1; y:-1
2030 value:=3; temp:=1
2040 if x mod 2 then value:=2; temp:=-1
2060 if y mod 2 then value:=value+temp
2080 x:=x div 2; y:=y div 2
2090 if x<0 or x>39 or y<0 or y>24 then error
2100 screen:=32768+40*y+x
2110 sdchar:=chr$(peek(screen)) in ddchars$
2115 if sdchar>0 then sdchar:-1
2120 endproc find
3999 //
4000 proc error
4010 print "plotting range error"
4020 stop
4030 endproc error
4999 //
5000 proc plot(sdchar,screen) closed
5010 dim ddchars$ of 16
5020 ----------as created above--------------------------
5030 poke screen,ord(ddchars$(sdchar+1))
5040 endproc plot
```

Having created the full routine LIST it to disk by LIST"O:DDPLOTTING.L". You can now enter this program to memory or add it to your programs as you wish. When prepassed, the three commands, SET, RESET and POINT will be available from the keyboard or to a program as if they were defined in the COMAL language.

Once more the power of COMAL procedures is shown. I might add that at a recent club night (North Herts Branch), a COMAL programmer Martin Simpson, added some further routines to draw straight lines in various directions. More of that next time perhaps. COMAL LOGO is not far away!

--oOo--

DISK FILE - SECTOR 4

By Mike Todd.

I've had a couple of queries since I wrote sector 3, so I'll handle these first.

Brian Grainger's article on "DEBUGGING AND DOSSING AROUND" (page 370 of the November '82 Newsletter) raised several points and there are two which I would like to take up.

Firstly, Brian has pointed out a problem in nomenclature between all the different DOS versions. They are often referred to as DOS1.2, DOS2.1 and so on; they are also referred to as DOS1, DOS2A and so on.

As Brian rightly points out, 2A, 2C are the formats of the disks themselves, but these references cannot be used to identify the different versions of DOS. For instance, the Vic disk drive records to format 2A, but uses a totally different version of DOS to the 4040 which also records to format 2A. The only way to identify the different versions of the Disk Operating System (which is what counts if you are going to access memory locations within the disk drive) is to use the terms DOS1, DOS2.1 and so on.

There are at least 5 versions of DOS2 (DOS 2.1 & 2.2 used in 3040/4040, the DOS's used in the 1540 & 1541 Vic drives and the 2031 PET single drive), yet they all work with 2A formats and their disks are fully interchangeable - but they all differ in internal memory usage.

So I would suggest that explicit reference to the DOS type be made when doing anything with the internal RAM or ROM of the disk drives, but the format can be specified if it is related to the disk itself and is not DOS-dependent.

Brian also throws down the gauntlet on p371. He wants a definitive answer to the structure of the U1 command.

I cannot be 100% sure that the following is true within the DOS versions that I've not seen, but it is certainly true for DOS1.0, DOS1.1, DOS1.2, DOS2.0, DOS2.1 and the VIC1540 DOS (2.7?) and since the coding in all versions is absolutely identical, I see no reason why the other DOS's (notably DOS2.2, DOS2.5 and DOS2.6) shouldn't be the same.

The U1/U2 commands are generically the same as the "B-x" commands which assume the general syntax as follows:

```
B-x: 999 T 999 T 999 T 999 T
```

where "x" is the command, "9" are ASCII decimal digits and "T" are terminator or separator characters. The presence of the colon following the command (whether "B-x" or "Ux") is optional and if it is absent, the parameter list is assumed to start at the fourth character, otherwise the first character following the colon is taken as the first character of the parameter list.

The separators are normally spaces, commas or cursor right hence the fact that any of the following will work:

```
PRINT#15,"B-R:";CH;DR;TR;SE = B-R: 99 99 99 99
PRINT#15,  "U1",CH,DR,TR,SE = U1   99     99      99     99
PRINT#15,  "U1:CH,DR,TR,SE" = U1:99,99,99,99
```

All of which are acceptable (assuming of course that you substitute the correct numbers in the last one!) and all "B-x" commands are treated in exactly the same way. In fact the parameter string is parsed as soon as the "B" is identified and the parameters stored in a table for use by the appropriate routines. The "Ux" commands are not "pre-parsed" but both "U1" and "U2" execute the same parsing routine immediately on entry.

There is no way that a byte-oriented parameter string can be acceptable to these commands, and the sequence "U1:";CHR$(CH);CHR$(DR);CHR$(TR);CHR$(SE) cannot function correctly, although it is accepted by the parsing routine.

However, the memory oriented commands "M-x" work in a totally different way and are not parsed. They are byte oriented commands and must follow this byte oriented format.

I don't know where the myth of the B-x and U1 commands having a byte oriented structure came from, but I have heard several people mention it and it even appears in the PET Bible by Ray West.

The other query I've had also stems from a myth which has been propagated by many "authoritative" voices and is the write-protect problem in DOS1 - many sources say, or at least imply, that the write protect facility should not be used with DOS1 under any circumstances.

Although I've covered the gory details several times before (p139 in July 1980 and p341 in November 1982 are the most notable), armed with the coding details in the last DISK FILE, I can give the precise reason why.

The actual formatting is performed by the FDC as would be expected, but the actual routine to do it is stored in the Interface Processor's ROM! The IP transfers this code into RAM before the FDC can execute it.

There is a fair bit of house-keeping and other chores executed before the routine starts to perform the formatting - eventually, the following code is executed:

```
        LDA #$DC
        STA  $4C    ;select WRITE NORMAL mode (p339 Nov '82)
        LDA #$08
        AND  $82    ;check write protect
        BEQ  START  ;branch if OK

        LDA #$08    ;flag WRITE PROTECT ERROR
        JMP ERROR   ;and exit

START   start of main formatting routine
```

You will see that write current is applied to the head BEFORE the write protect notch is checked! This is made particularly disastrous since the ERROR routine doesn't bother to turn it off and all subsequent operations assume that the head is in the READ mode.

The result is that any subsequent operations (which may occur on either drive and anywhere on the disk) are executed with a head in WRITE mode - and data is casually erased in the process. So, if the red error light comes on at any time during a disk format, or the format routine seems to stop stepping the head (the drive runs but the head stops its gentle "ticking") open the drive doors immediately and turn the disk unit off and on again. I'm afraid there is no easy solution to this bug (other than upgrading to DOS2.1).

This problem only occurs with DOS1 and only during the header routine (the DOS1 write routine does things in the correct order), the other DOS's seem to do things in the right order and will enter the ERROR routine in the read mode.

THE DISK FORMAT

Last time I promised to describe the exact layout of the data on the disk, unfortunately time has prevented me from producing a diagram showing how this is organised and this will have to wait until another time.

A format 1 or 2A disk has 35 concentric tracks numbered 1-35 with track 1 as the outermost track and 35 innermost while 2C has 77 tracks. It is this increase in track density that gives format 2C its increased capacity, together with a small increase in density within the track.

The tracks have a variable number of sectors according to where they are on the disk, with outermost tracks having the largest number of sectors since they are effectively "longer". The details are as follows:

--format 1-- track range	sector range	--format 2A-- track range	sector range	--format 2C-- track range	sector range	DISK ZONE
1-17	0-20	1-17	0-20	1-39	0-28	0
18-24	0-19	18-24	0-18	40-53	0-26	1
25-30	0-17	25-30	0-17	54-64	0-24	2
31-35	0-16	31-35	0-16	65-77	0-22	4

The disk zones are those referred to on p159 (May '82) and you will see that the only difference between 1 and 2A is that each of tracks 18-24 in 2A has one less sector. The result of this is that attempts to write to a 2A disk from DOS1 may try to access a non-existent sector and writing to a format 1 disk by DOS which expects a 2A formatted disk will not use the extra sectors.

Because of the potential problems, any DOS2 will not allow writing to a format 1 disk although format 1 and 2A disks are READ-compatable since the 2A DOSs will recognise a format 1 disk and accept the extra sectors.

Also note that the track numbering starts at 1 while the sector numbering starts at 0.

Each sector on the disk is made up of two blocks - a short "header" block (also referred to as the "address" block) and a longer data block.

The header block is made up of a SYNC character (as discussed earlier), a block identifier byte (BI) which for the header is 08, a checksum byte (CKS), the sector and track numbers (SCT and TRK) and finally the two bytes of the disk ID specified when the disk was initially formatted (IDL and IDH).

There then follows a gap (known as the HEADER GAP) and then the data block starts with another SYNC character and a block identifier (in this case 07). This is followed by 256 bytes of data (the first two of these are used as "link"

bytes, of which more at a later date) and finally there is a checksum digit. There is then an INTER-SECTOR GAP before the next header block starts.

When the disk is formatted by the NEW command, all the header blocks are written onto the disk, correctly positioned. On DOS1, dummy data blocks are also written containing all zeros but on later DOS versions, there is no data block, only a gap where it should be.

The size of the gap between a header and its data block is fixed by the DOS. It can't be too small or the SYNC character of the data block may have passed the head by the time the DOS has gets round to reading the data block. If it's too long, space on the track is being wasted and the capacity of the disk is reduced.

On DOS1, the inter sector gaps were also fixed in length, but this led to some problems on some tracks where the number of sectors was being pushed to its limit. If the speeds of the formatting drive and the drive writing data were significantly different, it is possible that the last sector on a track could "bleed" over into the first sector.

As well as reducing the density on tracks 18-24 (where the problems had occurred), the formatting routine was made intelligent enough to work out the total capacity of the track from the disk itself (by timing its rotation) and then work out the optimum sector spacings, having been given minimum values to work from.

Some non-Commodore disk drives use sector interleaving (that is the sectors are not in numeric order but are in a special sequence such as 0-3-6-9-12-15-18-1-4-7 and so on). This is because having read sector 0, the FDC may just miss sector 1 and have to wait for the disk to fully rotate before it can be read. This can cause long delays, but, by using interleaving, sector 1 is a couple of sectors away and should allow quicker access.

Commodore don't do this in this way, instead they have their sectors running consecutively, but use the software to choose the most efficient sector on which to write data.

Note that the FDC uses the header blocks to keep a check of which track the head is on, to find the required sector and also to maintain an ID check on the disk. The ID shown in the directory is only an "aide memoire" for the user to remind him of the ID on the disk and, as we're constantly reminding you, should never be altered.

THE JOB QUEUE

In order to communicate with the FDC, the IP uses the common RAM area to store the data and to pass read or write requests to the FDC. I will leave full discussion of exactly how the IP handles this for another time, but I will describe the principles involved from the FDC point of view.

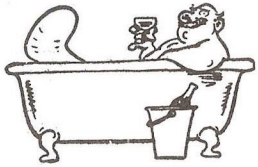
Each 256-byte buffer has an associated job request byte in a job queue table. Each buffer also has an entry in a header table and to read a specific sector into a buffer, the IP puts a job request byte into the queue, puts the header of the expected sector into the table and then waits.

Next time I hope to produce a diagram showing the location and organisation of all these tables and buffers.

Bit 7 of the job request byte is set to 1 to indicate to the FDC that this is a job request and the FDC returns a status code with bit 7=0 when the job is completed. It is 01 if the job was successful.

The way the the FDC handles this job queue is quite complex. It continually scans the job queue, picking out which job can be executed most efficiently, bearing in mind which drive motors are running, which track the head is on and even which sector will pass under the head in the next few milliseconds.

The codes used in the job queue are as follows. Note that the drive number of the requested job is contained in bit 0 so that a READ on drive 1 would have a job request byte of $81.



$80 - READ a sector
$90 - WRITE a sector
$A0 - VERIFY a sector
$B0 - SEEK any sector
$C0 - BUMP head to track 1
$D0 - JUMP straight into machine code in buffer
$E0 - EXECUTE code in buffer with head, motor etc set up

Jobs $80 and $90 are self explanatory, $A0 is only used at the end of a WRITE to make sure that data written reads back exactly as it was written. The SEEK command allows access to the ID of the disk and BUMP produces those horrid grating noises as the heads are returned to track 1 so that, from then on the disk drive knows where they are.

The JUMP command forces the FDC to execute the code in the buffer as soon as the command is found in the job queue (not implemented on DOS1), while EXECUTE waits for the drive to be available and the head to be positioned on the correct track.

It is not possible to use the M-R commands directly on the FDC address space and so a small routine must be written and transferred into one of the buffers and then executed, using the EXECUTE or JUMP commands. This routine would transfer FDC addressed memory into the common RAM area, making it accessible by the IP and therefore by the PET.

As well as having bit7=0, the error codes are in the range 1-16 ($01-$10) and, apart from the OKAY code, are converted into numbers in the range 20-29 for sending to the PET when the error channel is interrogated.

The following page shows these error codes:

FDC CODE	IP CODE	DESCRIPTION
$01	--	OKAY
$02	20	HEADER block not found - FDC has a limited number of tries to find the header of the requested sector
$03	21	No SYNC character detected within time limit.
$04	22	DATA block not found - once FDC finds the requested header it expects it to be followed by a DATA block with block ID=07.
$05	23	DATA block checksum error - once all data is read into buffer, a checksum is generated and compared with the checksum read from disk.
$07	25	WRITE VERIFY error - a READ of the data written is performed and compared with the data in the buffer.
$08	26	WRITE to WRITE PROTECTED disk attempted.
$09	27	HEADER block checksum error - whenever any header is read a checksum is generated and checked against the checksum on the disk.
$0A	28	DATA block too long - this was intended to identify the fact that a DATA block had over-written the next HEADER block but does not appear to be implemented in the FDC.
$0B	29	ID mismatch - returned if the IDs of the HEADER blocks read do not correspond with the ID given in the requested header.
$10	24	Byte decoding error - once all data is read into buffer, the FDC checks the ERROR flag which will be set if one or more bytes were incorrectly read.

LOCATION	DESCRIPTION
00	Interrupt counter - used to time motor turn off delay
01	Motor delay counter for drive 0
02	Motor delay counter for drive 1
03	Drive & head status flags and track log for drive 0
04	Drive & head status flags and track log for drive 1 Bits 0-5 track log Bit 6 head settled flag (0 if settled) Bit 7 motor up to speed flag (0 if up to speed)
05	Number of tracks for head to step for drive 0
06	Number of tracks for head to step for drive 0 Bits 0-6 number of tracks * 2 Bit 7 direction (0 if step IN) Note: 05 and 06 are decremented/incremented as head is stepped and are zero when the head is on the correct track
07	temp store for $40 during interrupt routine
08	checksum work byte .. smallest track difference
09	Smallest actual track difference found in job queue
0A	Difference between current track & track of job in queue
0B	Difference between current sect and sect of job in queue
0C	Smallest actual sector difference
0D	IDH of current header block
0E	IDL of current header block
0F	TRAK of current header block
10	SECT of current header block
11	CHKS of current header block
12	Current drive number
13	Current track number
14	Current sector number
15	Number of sectors on current track
16/7	Current buffer address (points to first byte in buffer)
18/9	Current header address (points to first byte of header)
1A	Current track number used during HEADER
1B	Not used
1C	Not used
1D	Retry counter after error occurs in HEADER
1E	Current job request code (bits 4-6 only)
1F	Current buffer number
20-3F	Hardware stack
40-4F	6522 VIA chip
80-8F	6530 RIOT chip (RAM Input Output Timer)
0400	IRQ timer constant (sets number of interrupts per second)
0401	Motor turn off delay constant
0402	Flag set during RESET sequence to tell IP that FDC is OK
0403-0411	Job queue (indexed by buffer number $00-$0E)

LOCATION	DESCRIPTION
0421-0498	Buffer header table (contains IDH IDL TRK SEC CKS for each buffer $00-$0E
0499-049C	Number of blocks per track in each zone (set up by IP)
049D	Header gap size (set up by IP)
049E	Inter sector gap minimum size (set up by IP)
049F	Disk format identifier (set up by IP)
0500-05FF	Buffer #0
0600-06FF	Buffer #1
0700-07FF	Buffer #2
0800-08FF	Buffer #3
0900-09FF	Buffer #4
0A00-0AFF	Buffer #5
0B00-0BFF	Buffer #6
0C00-0CFF	Buffer #7
0D00-0DFF	Buffer #8
0E00-0EFF	Buffer #9
0F00-0FFF	Buffer #A
1000-10FF	Buffer #B
1100-11FF	Buffer #C - BAM for drive 0
1200-12FF	Buffer #D - BAM for drive 1
1300-13FF	Buffer #E - Not used in DOS2.1 (workspace for IP)

FDC ROM ROUTINES

NB: These routines only give a rough guide and are not intended as a full programmer's guide.

FC00-FC03	Main holding loop and error routine vectors
FC04	"hold" routine for reset entry from IP
FC09	RESET - set stack pointer, check RAM, set I/O chips
FC54	Main holding loop - start scanning queue for job request
FC60	... JUMP code ($D0) found - so go and do it
FC65	... turn drive motor on & start turn off delay
FC8A	... exit loop if head is settled on current drive
FC92	Scan job queue for all jobs on current track
FCAC	... BUMP code ($C0) found - so go and do it
FCB3	... are we on the right track? exit to FCF4 if we are
FCBC	... note job whose track request is nearest current track
FCD3	Loop has given up - we're not on track of any job so ...
FCE0	... set heads to nearest track requested and goto FC54
FCF4	On right track, but is drive up to speed?
FCF8	If head & motor OK, work out zone & number sects on track
FD20	... EXECUTE ($E0) code was found so go and do it
FD1A	BUMP head to track 1: force head IN max number of traks
FD3F	Look for job on queue on current drive & track and nearest to current sector
FD98	... found one, set pointer to its header and go and do it
FDA5	Subroutine to set job parameters such as header pointer
FDBB	Force current sector to requested sector then goto FD3F

LOCATION	DESCRIPTION
FDC4	Identify job request (READ, WRITE & VERIFY)
FDC8	READ routine: find start of specified DATA block & read bytes into buffer. When complete check checksum at FE6B
FDDD	Subroutine to find start of specified DATA block ...
	... search for specified HEADER, wait for following block
	... check it's DATA block (BI=07) - ERROR #04 if not
FDF4	WRITE routine: check write protect - ERROR #08 if on
	... search for specified HEADER, wait ($049D) bytes ...
FE11	... write 3 SYNC bytes, DATA block ID (07) & write data
	... write checksum and terminate WRITE
FE4A	... turn WRITE request into VERIFY, then go to FDBB
FE57	VERIFY routine: find start of specified DATA block
	... read and compare with buffer - ERROR #07 if mismatch
FE6B	... read checksum ... ERROR #05 if not OK
FE76	... check ERROR flag .. ERROR #10 if set
FE82	Read any HEADER block and set track log from it ...
	... check for SEEK ($B0) request and goto FEBC if it is
	... otherwise, compare ID from disk with ID from IP
FEBC	SEEK routine: transfer header from disk into header table
FECB	ERROR #09 - checksum error in header
FECF	ERROR $0B - disk ID mismatch
FED3	Subroutine to find specified header: set retry counter
	... read first header available and compare with request
	... if they don't match, try again preset number of times
FF02	Subroutine to find ANY header block - will try only .X times, ERROR #02 if not found after .X tries
FF08	ERROR handler: flags error in job queue, reset stack & go back to join main holding loop at FC54
FF24	part FF02 routine - wait for any block & identify it
FF2C	Subroutine: identify SYNC within .X characters (used by HEADER routine only)
FF3E	Subroutine: search for any block - set time out delay ...
	... wait for SYNC until time out occurs
FF58	Subroutine: terminate WRITE mode ... restore READ
FF79	Subroutine: output byte from .X
FF82	INTERRUPT routine: save .A & .Y, reset IRQ timer ...
	... increment IRQ count in $00 & then, for each drive
FF8E	... handle turn off and up-to-speed delays & set flags
FFB7	... handle head stepping
FFE8	Masks for head and motor control bits
FFFC	RESET VECTOR - FC09
FFFE	IRQ VECTOR - FF82

Armed with all this information, here is a summary of the way that the FDC operates.

SCANNING THE JOB QUEUE

The queue is constantly scanned and as soon as a job request is found, it is checked. If it is the JUMP command then execution continues at the first byte of the appropriate buffer. If it's not, the appropriate drive is turned on and the up-to-speed delay is started. Then a check is made to see if the head has settled and the loop continues until a job whose head has settled has been found.

If the the job request was BUMP, the head is settled and the correct drive is set up, the FDC will drive the head back to track 1 and return to the main holding loop.

If not BUMP, a check is made to see if the head is at the same track of the selected job. If it is, a quick check is made to confirm that the drive is available and the routine exits to handle the job.

If the job is not on this track, the job queue is scanned for a job nearest the current track and a request is made for the interrupt routine to move the head to the required track. The loop is then rejoined and is only left when the head has arrived.

Once the correct track has been found, the necessary zone timings are set up and the number of sectors on the track is worked out. It is at this point that the EXECUTE command is identified and execution continues at the first byte of the appropriate buffer.

The first available header on the disk is then read, its checksum checked, and the FDC's own track record is updated. If the job request was SEEK, then the header read from the disk is transferred into the header associated with the current buffer and that job is complete.

The only jobs left are READ, WRITE and VERIFY and the FDC procedes to check that the ID just read from the disk is the same as the ID provided in the buffer header.

If all is well, a sector two sectors ahead is considered the current sector and the job queue is searched for any request for this sector. If not found, the job which is closest is chosen and the READ, WRITE or VERIFY routine is entered as approriate.

READING A BLOCK

A subroutine is first called which waits until the header of the requested sector is read. This is done by reading every HEADER block on the track until its contents exactly match the requested header.

The routine then waits for the SYNC pulse of the next block and reads the first byte, which should be the block identifier 07. If it's not, error 07 is generated.

Then, all 256 data bytes are read and placed into the correct buffer. A check is made of the checksum and of the ERROR flag and if all is well, the OKAY code is returned in the job queue.

WRITING A BLOCK

First, the write protect notch is checked and if it is not covered, the routine waits for the correct header to be found in exactly the same way as for READ.

Before starting to write the data, a delay loop waits to allow the inter-block gap to pass the head and then the drive is switched to write mode, three SYNC bytes are written followed by the DATA block identifier and then the 256 bytes from the buffer are written followed by their checksum.

Once complete, the WRITE routine cunningly turns its job code from $90 to $A0 and rejoins the main loop. This will

result in the data just written being verified on the next rotation of the disk.

VERIFY DATA

This acts exactly like the READ routine except that each byte read is compared with the buffer contents.

INTERRUPT SERVICING

The only section of the FDC I've not covered is the interrupt routine. This is called approximately 65 times a second (although it resets the IRQ timer from $0400 and so the timing can be altered) and has two distinct sections.

The first handles the drive motors. There are two delays associated with each motor. There is the up-to-speed delay which notes that a motor has just been turned on and sets bit 7 of 03/04 (drive 0/1) until the delay has expired.

The other delay is the one which turns the drive off if it is not accessed for a while.

The second half of the routine looks after the two head stepper motors. A signed byte in 05/06 (drive 0/1) is set up when the FDC wants the head moved. This byte indicates how many tracks, and in which direction, the head should move.

The interrupt handles this stepping and sets bit 6 of 03/04 (drive 0/1) until the head has settled on the track.

The rest of 03/04 is used by the FDC to keep a note of the actual track on which the head is currently positioned and this information is not used by the interrupt routine.

FINALLY

That's all for this DISK FILE. I don't know what'll be in the next one, but in the meantime here are some memory maps of RAM & ROM used by the FDC.

--ooo--

BASIC/COMAL DISK ID CHECKER

By Dennis Lyons and Brian Grainger

The idea for this article is twofold. Firstly to introduce a simple but useful program and secondly to show a comparison between a BASIC program and a similar program written in COMAL. Following warnings in earlier Newsletters about not duplicating disk IDs a friend of mine, Dennis Lyons, sat down and wrote two programs in BASIC. One to print all possible ID codes for disks to use as a chart. The second to keep a record of all IDs used and to check that new ones have not been used before. The two programs, as written in BASIC, follow:

PROGRAM 1

```
100 dim id$(1296)
110 print"<clr,19spaces>PROGRAM TO CHOOSE
    2 CHARACTER DISK ID
200 rem disk id check progrem
210 rem print used id's to disk and
220 rem read them back against
230 rem keyboard input
300 open8,8,8,"used ids,s,r"
320 input#8,a$:printa$:rem filler
330 i=i+1:input#8,id$:printid$,:id$(i)=id$
340 if st then 400:rem end of used id's
350 go to 330
400 rem ** input *
410 print
420 print"choose 2 character id": input id$
440 iflen(id$)<>2then410
500 rem * check with input array *
510 for x=1 to i
520 ifid$=id$(x)thenprint"id "chr$(34);id$;chr$(34)
    " already in use": x=i:goto410
530 next
600 rem * append new id to disk *
610 append#2,"used ids":print#2,id$:dclose
620 print"chosen id "chr$(34);id$;chr$(34);
    " now added to disk record  used ids
990 end
```

```
1000 open8,8,8,"@0:used ids,s,w"
1010 print#8,"id's used up so far"
1020 dclose
```

PROGRAM 2

```
100 REM * PROGRAM TO GENERATE A PRINT OF ALL POSSIBLE
    ID'S *
101 PRINT"<clr>PROGRAM TO GENERATE A PRINT OF ALL
    POSSIBLE ID'S
105 PRINT"<dn>PLEASE HAVE PRINTER READY
106 PRINT"<dn>PLEASE PRESS RETURN TO PRINT
107 GET A$:IFA$<>CHR$(13)THEN107
110 OPEN4,4:PRINT#4,CHR$(27);CHR$(69);
120 PRINT#4," DISK ID'S":CLOSE4
130 A$="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
140 FORX=1TOLEN(A$):OPEN4,4:PRINT#4,MID$(A$,X,1);" ";:
    C=C+1:CLOSE4:NEXT
150 FORX=1TOLEN(A$)
160 FORY=1TOLEN(A$)
170 OPEN4,4
180 PRINT#4,MID$(A$,X,1);MID$(A$,Y,1);" ";
190 C=C+1:CLOSE4
200 NEXT:NEXT
210 OPEN4,4:PRINT#4
220 PRINT#4,C;" ID'S GENERATED"
230 PRINT#4:CLOSE4
```

Having written the above programs in BASIC, Dennis decided to get to grips with COMAL- at long last! What better way to start but by writing the same programs in COMAL. You will see from the results below that the programs have become longer. This is partly due to improved layout and error checking from the original BASIC programs. However this is more than compensated for by the fact that the programs are far more understandable and easier to write. The way Dennis has used the very powerful procedures of COMAL to write the revised programs is a credit to him. Well, so much for the words, here are the COMAL programs:

PROGRAM 1 (Be sure to SAVE as "CHECK'DISK'IDS)
Note: to run this program put the program disk in drive 1

```
0100 // comal utility
0110 // by dennis lyons
0120 // ================================
0130 // program to store used disk id's
0140 // on disk and check a randomly
0150 // chosen disk id against disk
0160 // ================================
0170 dim id$ of 4, iden$(500) of 2, disk'error$ of 22
0180 count:=0; e#:=0 // error flag
0190 // ============================
0200 get'used'ids
0210 //
0220 choose'new'id
0230 //
0240 check'id                    (c) ICPUG 1983
0250 //
0260 add'id'to'disk
0270 //
0280 re'run //until '00'is entered
0290 //
0300 // ============================
0310 proc get'used'ids
0320 print "            disk id program    "
0330 print "used id's.."
0340 open file 2,"1:used id's",read
0350 get'disk'status
0360 if e# then
0370 e#:=0
0380 make'id'file
0390 close
0400 get'used'ids
0410 open file 2,"1:used id's",read
0420 endif
0430 repeat
0440 count:=count+1
0450 read file 2: id$
0460 print id$;
```

```
0470 iden$(count):=id$
0480 until eof(2)
0490 print " "
0500 close
0510 endproc get'used'ids
0520 //
0530 proc choose'new'id
0540 input "type in 2 character id (00 to end)": id$
0550 if id$="00" then end
0560 lenid:=len(id$)
0570 if lenid<>2 then
0580 choose'new'id
0590 endif
0600 endproc choose'new'id
0610 //                                    (c) ICPUG 1983
0620 proc check'id
0630 for check:=1 to count do
0640 if id$=iden$(check) then
0650 print "id ";chr$(34);id$;chr$(34);" already in use"
0660 for delay:=1 to 1000 do null
0670 re'run
0680 endif
0690 endfor check
0700 endproc check'id
0710 //
0720 proc add'id'to'disk
0730 open file 2,"1:used id's",append
0740 write file 2: id$
0750 close
0760 print "chosen id ";chr$(34);id$;chr$(34);
     " now added to disk"
0770 endproc add'id'to'disk
0780 //status procedure
0790 //use this after disk commands
0800 proc get'disk'status
0810 disk'error$:=status$
0820 if disk'error$(1:2)<>"00" then
0830 print "disk error #";disk'error$
0840 e#:=1
0850 endif
0860 endproc get'disk'status
0870 // make id file if not existing
```

```
0880 proc make'id'file
0890 close
0900 open file 2,"used id's",write
0910 write file 2: " "
0920 close
0930 endproc make'id'file
0940 //
0950 proc re'run
0960 chain "1:check'disk'ids"
0970 endproc re'run
```

PROGRAM 2

```
0100 //a comal program to generate
0110 //all possible disk ids
0120 //
0130 //set up strings & etc
0140 //                                (c) ICPUG 1983
0150 zone 3
0160 count:=0
0170 dim id$ of 36, starter$ of 1
0180 id$:="0123456789abcdefghijklmnopqrstuvwxyz";
     len'id:=len(id$)
0190 //-----------------------------
0200 input "press return to start*": starter$
0210 select output "lp"
0220 print "                              all possible disk ids"
0230 print'id1
0240 print'id2
0250 print
0260 print count;" disk ids"
0270 end
0280 proc print'id1
0290 //first print all single chr ids
0300 for first'id:=1 to len'id do
0310 print id$(first'id),
0320 count:=count+1
0330 endfor first'id
0340 endproc print'id1
0350 proc print'id2
0360 //next print all 2 chr ids
0370 for first'chr:=1 to len'id do
```

```
0380 for second'chr:=1 to len'id do
0390 print id$(first'chr);
0400 print id$(second'chr);
0410 print " ",
0420 count:=count+1
0430 endfor second'chr
0440 endfor first'chr
0450 endproc print'id2
```

I would just like to finish this article by mentioning that since writing these COMAL programs Dennis has been doing more work in COMAL. By his own admission he was not a perfect programmer in BASIC. With the help of Atherton's book and the CBM COMAL he has now 'seen the COMAL light' and considers the language essential and sufficient to write programs for his needs.

--o0o--

2031 DRIVE PATCH

There cannot be many of you trying to operate a 2031 single disk drive unit from an upgrade ROM (2.0) PET, but if so you may have a problem. The fix is a patch in the PET $F000-ROM as follows:

```
F17F   4C ED FF
F182   EA

FFED   AD 40 E8
FFF0   29 FB
FFF2   8D 40 E8
FFF5   A9 5F
FFF8   4C 87 F1
```

You can either program a substitute EPROM, or obtain one from Wilserv Industries, P.O. Box 456, Bellwamr, NJ 08031, USA. Price $15 (probably more in UK).

--o0o--

REGISTER EXCHANGE

By Andy Scott.

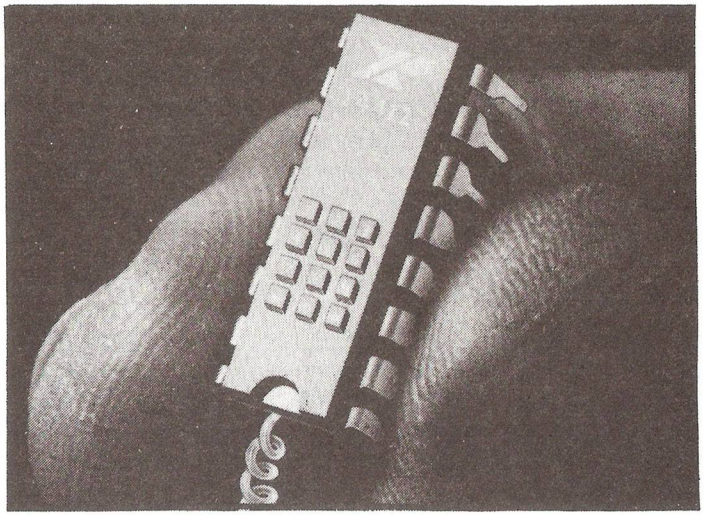
I required a subroutine which exchanged the contents of the accumulator with the Y register, without affecting the contents of other registers or memory locations. Similar programs can be written to swap the accumulator with the X register or swap both X and Y registers.

The stack is used for the various manipulations, as well as X/Y registers, Stack Pointer and Accumulator. After each subroutine call, the stack pointer and status register assume their original states. It is important that the interrupt flag is conditioned properly in order to stop interrupt routines writing over the middle of the stack.

```
PHP     ;PRESERVE STATUS REGISTER
SEI     ;PREVENT IRQ'S AFFECTING STACK
        ;
PHA     ;PRESERVE ACCUMULATOR
TYA
PHA     ;PRESERVE Y REGISTER
TXA
PHA     ;NOW X
        ; INCREMENT STACK BY 3
PLA     ;1.. OLD X REGISTER
PLA     ;2.. OLD Y REG
PLA     ;3.. OLD ACC
        ;
TAY     ;GOT OLD ACC NOW PUT IN Y.
        ;
TSX     ;ADJUSTING STACK POINTER
DEX     ; BACK TO WHERE
DEX     ;IT SHOULD HAVE BEEN
DEX
TXS     ;MAKE GOOD
        ;
PLA     ;GET OLD X
TAX
PLA     ;GET OLD Y (INTO ACC NOW)
PLP     ;THROWAWAY OLD ACC (IN Y NOW)
        ;
PLP     ;GET OLD STATUS BACK
RTS     ;GO HOME
```

Although the interrupt flag was set at the beginning of the routine (SEI instruction) it is not necessary to clear it at the end. The second from last instruction (PLP) restores the old status register - clearing or setting the interrupt flag to what it was before! The SEI instruction, following PHP is essential to ensure that the stack is uncorrupted.

--o0o--



No, I didn't mean put the whole thing on a chip!

THE PET INDEX

Mick Ryan's book "The PET Index" (reviewed p216, July '82 issue) is now reduced in price to only £ 6.95 plus £ 1.00 post & packing. Please send your order direct to Gower Publishing Company, Readout Publishing Co. Ltd., 8, Camp Road, Farnborough, Hants. The PET Index Disk (state 4040/8050) is £ 18.25 (incl. VAT and p & p), Book and Disk together £ 22.75 incl.).

--o0o--

Discount Corner

By John Bickerstaff

For discounts on CBM hardware (17.5%) and CBM software (20%) please write to me stating your requirements enclosing an SAE.

For 5.25" single-sided, single- or double-density disks with reinforced hub rings, please order directly from me enclosing payment by cheque made out to ICPUG. Ten disks cost £ 16.60. An addressed 'Jiffy Bag' with 56p in stamps should be sent with your order, which will usually be met by return of post.

Superscript is once again available to ICPUG members at a special price of £ 75.00 plus VAT. Superspell will also be available for the same price. Alternatively the two can be provided at £ 150.00 as one unit complete with a security key. Updates for your existing Supercript are £ 5.00 plus V.A.T. and new manuals at £ 11.00 each. Please apply to me in the first instance enclosing a stamped addressed envelope advising which format, 3040/4040 or 8050, you require. Only one copy of each program can be supplied to any member.

It is hoped that Ray West's book "Programming the PET/CBM" will soon be obtainable from Group secretaries. Otherwise, please write directly to me sending a cheque for £ 10.50 made out to ICPUG and a stamped 'Jiffy Bag' (Postage for 1 kilogram for first class is currently £ 1.57, or parcel postage £ 1.20 UK rates). The jiffy bag should be large enough to take the 10.25" x 7.5" x 1" thick book.

Please note my new address which is:-

45, Brookscroft,
Linton Glade,
Croydon CRO 9NA.
Tel: 01-651 5436

I will be pleased to search for discounts on any hardware or software required for your Commodore System, but please enclose a stamped self-addressed envelope for my reply. Jiffy bags can now be bought from the Post Office.

--ooo--

24-PIN 'OLD-ROM' 2001-SERIES CONVERSION TO BASIC 2.0 OR 4.0

By Mick Bignell

Owners of the old 2001-series PETs, fortunate enough to have the 24-pin ROMs fitted to their main logic board, may easily convert the pcb to accept the latter versions of BASIC (i.e versions 2.0 and 4.0) plus the additional sockets for ROM expansions such as TOOLKIT, POWER, ARROW etc.. These are only available on-board on the later dynamic RAM machines.

These instructions refer only to printed circuit boards marked:- 'PET MAIN LOGIC ASSY 320137' and when the conversion is complete it will not be possible to use the machine with the old BASIC 1 chips again.

Remove the power supply socket from J8, keyboard connector from J5 and the video socket from J7. Remove three screws holding the main logic board in position (2 at rear and 1 on right), while gently lifting the board release three nylon pillars (2 at front and 1 on left). Remove board from machine.

There are seven ROM sockets in row 'H' (conveniently the same as later PETs), remove the old 2K ROMs and discard. At this point it is possible to fit BASIC 2.0 4K ROMs directly into the sockets H1 to H4, leaving the H5 to H7 empty, with no further mods required. This is because BASIC 2.0 consumes the same amount of memory space as BASIC 1 ($C000-$FFFF). However the empty sockets cannot be used at all as they also address at some of these locations. Fitting an extra chip in any one of these sockets would cause a conflict on the data bus and prevent the machine from powering up.

On old PETs the hardware assumes that addressing anywhere in the range $9000 to $BFFF will be to an external device and therefore enables the data bus buffers associated with the memory expansion port. These buffers must be disabled if we are to use the on-board three spare sockets at these addresses. If not disabled, when the processor reads a ROM at these locations the buffers will switch on and again corrupt the data on the bus.

The mods to the PCB alter the three spare sockets to address in the $9000, $A000 and $B000 4K blocks and also disables the offending buffers anywhere within these 12K of locations.

HARDWARE CONVERSION.

N.B. All integrated circuit pins are numbered counting in an anti-clockwise direction. Looking from the top of any chip with the 'notch' end on the left, and the numbering the correct way up for reading, pins are counted from left to right on the lower edge and right to left on the upper edge.

On the top side of the pcb, cut track between pin 5 of chip G4 (74LS21) and 1K resistor/G3 pin 9 (74LS00), using a sharp modelling knife or scalpel. From the under-side of the board cut track between pins 4 & 5 of G4. On socket H7 pin 20 there is a track leading to a plated-through hole, from the upper side of the PCB, cut the track near this hole. Also repeat the same track cut for sockets H6 and H5.

Using some thin wire ('KYNAR' wire-wrapping wire is ideal), connect pin 20 of socket H5 to pin 13 of G1 (74154). Also the same pin 13/G1 should be connected to G4 pin 4. This part of the mod will cause socket H5 to address at $B000-$BFFF and disable the buffers. Pins 4 & 5 of G4 are spare inputs to a 'FOUR-INPUT AND' gate used to disable the data bus buffers when addressing ROM area.

One of these inputs we have now used for the $B000 socket. This leaves two sockets but only one remaining spare input on the gate G4. The logic we require to disable the buffers for two sockets with only one input is the 'OR' function on this gate's last spare input. As there are no spare 'OR' gates available this may be achieved using two high speed diodes of the 1N4148 type connected as a 'wired OR pair'. Solder together the two anodes of the diodes (anode = the end without the band) and each cathode (band end) to each of the plated-through holes connected to H6

pin 20 and H7 pin 20. Solder wire from the diodes common anode connections to pin 5 of G4. Solder wire from pin 20 of H6 to pin 10 of G1 and wire from pin 20 of H7 to pin 11 of G1.

THAT'S IT! Refit the board into the machine in the reverse order of dismantling. Fit the new ROMs as per table:-

Socket	H1	H2	H3	H4	H5	H6	H7
Start address	$C000	$D000	$E000	$F000	$B000	$9000	$A000
BASIC 2.0 ROMs	-01	-02	-24	-03	---	---	---
BASIC 4.0 ROMs	-20	-21	-29	-22	-23	---	---
Equivalent locations on dynamic RAM boards:	UD6	UD7	UD8	UD9	UD5	UD3	UD4

Any problems relating to this conversion or the interpretation of these instructions telephone Mick Bignell on 01-953-8385.

--o0o--

ICPUG SOFTWARE LIBRARY

Things are moving fast in the domain of the ICPUG Software Library, so much so that things are often out of date by the time one gets to hear of them. Many sources recommend the public domain program 'COPY/ALL' or its upgrades, now all of these are superseded by 'COPY/FAST'. 'COPY/FAST' will copy from ANY Commodore disk (hard or floppy) to any other (space permitting !), including relative files.

If you would like to assist in the cataloguing and collating of the programs in the library, please contact Carl Millin, 31, Northleigh Close, Loose, Maidstone, Kent, ME15 9RP.

--o0o--

UGLY BUG BALL TIME AGAIN

By Brian Grainger

It would seem the dreaded disk drives are out to bug me forever! After last time's identification of errors in the disk to tape saving program due to disk problems I found another one! This time because of the 2031 disk drive. Apparently this drive gets upset, when using BASIC2, if the error channel is accessed after the GET# command in certain circumstances. Thus in the revised line 2430 for the disk to tape program given last Newsletter remove the GOSUB2000. It now becomes:

```
2430 FORB=1TO255:GET#4,A$
```

Also on this drive, the disk buffers are in a different part of memory so line 2410 should be changed with this drive to read:

```
2410 PRINT#15,"M-R"CHR$(0)CHR$(3).
```

Moving away to the good(?) old 3040. It caused me great pain a few weeks ago when I could not ENTER, using this drive, some COMAL LISTed files. It worked perfectly during my preparation on a 4040. It didn't work when I came to giving a presentation at the COMAL User Group! Similarly early PET REVAS versions had problems with 3040 fitted with DOS1. The reason is that on this drive if you use secondary address 0 or 1, it expects a PRG file. Any other causes a filetype mismatch error. COMAL uses these addresses of course! So be warned. An up-to-date copy of PET REVAS to solve this problem can be obtained from Tom Cranstoun (address last Newsletter). COMAL will not be updated so get DOS2.

Talking of PET REVAS, users may have noted that a CPY immediate instruction gets translated into CPY 0 page. The solution is simple. IMMEDIATELY after LOADing PET REVAS you should find PEEK(2239) is 40. POKE2239,41 and SAVE IMMEDIATELY and all will be well in future.

--oOo--

SHOP WINDOW

Now I'm always pleased to announce a first in these pages, so when I see that Nottingham-based Keen Computers claim a first for their local area network for the CBM/PET series, I wonder if they have heard of Cluster One (at the first PET Show), Hydra and Commodore's Keynet, to cite but three. The Corvus Constellation is available on many machines and now interfaces to the PET using the Small Systems Engineering Corvus/IEEE-488 interface.

Small Systems also do a range of scientific and industrial interfaces. Their address is 18/19. Warren Street, London, W1P 5DB. Tel: 01-387 7388. Similar interfaces come from Biodata Ltd., 6, Lower Ormond Street, Manchester, M1 5QF. Tel: 061-236 1283. Also from Lee Dickens Ltd., Desborough, Kettering, Northants, NN14 2QW. Tel: 0536 760156.

I've always fancied an XY-plotter for producing diagrams for the Newsletter. One of the cheapest I have seen comes at £ 596 + VAT from J.J.Lloyd Instruments Ltd. The PD4 plotter is suitable for direct connection to the PET and has optional software available, including character generator. The address to contact is Brook Avenue, Warsash, Southampton, S03 6HP. Tel: Locks Heath (04895) 4221.

Now that the Commodore 64 is out, Oxford Computer Systems have produced a suite of cross-compilers which will compile 8000- or 4000-series source programs to C-64 compiled code and from Vic-20 to C-64. Also from OCS comes the Interpod, a 'universal' interface to convert between Vic-20 and C-64 serial interfaces and RS232 or IEEE-488. in either direction. The price is £ 95.95 + VAT from Oxford Computer Systems (Software) Ltd., The Old Signal Box, Hensington Road, Woodstock, Oxford, OX7 1JR. Tel: (0993) 812700. Commodore are reported to be negotiating with OCS regarding production of the Interpod. The PETSpeed BASIC Compiler from OCS is now available for the C-64 at £ 125.

If you need IEEE-488 cables and have fallen out with your friendly dealer, they may be obtained nowadays from many sources. Some of these are as follows: Cableforms and Connectors Ltd., 151, London Road, Camberley, Surrey, GU15 3JY. Fast Express Ltd., Bowater Road., London, SE18 5TF., Tel: 01-855 4344 or 9821. RS Components Ltd., P.O.Box 427, 13-17, Epworth St., London, EC2P 2HA. Tel: 01-250 4000 (queries), or 01-253 3040 (orders), and for real value, Supersoft - 2 metres for £ 32 + VAT. Tel: 01-861 1166.

Hackers and hardware enthusiasts may be interested in the catalogue from Control Universal which contains 6502 and memory boards, card frames and 65xx series interfaces. Control Universal are at Unit 2, Andersons Court, Newnham Road, Cambridge, CB3 9EZ. Tel: (0223) 358757.

Wordcraft has been available for some years now and in that time it has had several modifications. The latest version resides in ROM, rather than in overlays during use, and has some improvements in the screen editing functions, such as highlighting text that is to be moved.

'Video-Glasses' anti-glare screens are designed to reduce glare and enhance clarity at a price of £ 16.75, including post, packing and delivery from File Binders Ltd., 153-155, High Street, London, SE20 7DS.

If you are in the habit of pouring coffee over your keyboard, you can obtain a transparent silicone rubber keyboard cover which will fit all large-keyboard Commodore models for £ 6.00 + 90p VAT. Contact D.B.M. Products, P.O.Box 6, Melton Mowbray, Leicestershire, LE13 1YL.

Another chip to add commands to BASIC is offered, called Microbis. This adds 22 business-orientated commands to 8000-series machines - SORT can handle 1000 records in half a minute, KEYED random access commands, PAD a field with any character, FLASH selected screen messages, ENTER & validate data in screen windows, PACK & UNPACK data to conserve disk space, et al. The chip costs £ 46 from Microcomp, Homes Cottage, Reeds Lane, Liss, Hants. Tel: 073082 2512.

Your CBM/PET can act as a microprocessor development station for £ 999. A choice of processor modules from 6502, 6802, 6803 or 6809 housed in a half-width Eurocase, software controlled full-speed in-circuit emulator, user interrupts, hardware address trap and IEEE-488 interface are just some of the features. The software has a fast assembler with macro facilities and includes an extensive monitor/debug package. Details from Tylek Ltd., 2, Parkview Drive, Cashes Green, Stroud, Glos. Tel: 045 36 77257.

--o0o--

MATTERS ARISING

The program on p174 (July issue) to perform a printer presence check made references to a non-existent line 200. Furthermore the subroutines ended not in RETURN, but GOTO200. Naturally this caused some confusion but perhaps line 200 will help:

```
200 SYS50583:....rest of program...
```

This calls the back end of the CLR routine at the point where it sets the stack pointer to $FA after saving the RTS address. Hence any FOR-NEXT loops or subroutines left incomplete are cleared from the stack, yet no variables are lost, nor pointers altered. This enables nested subroutines to be aborted to some default point in the program, such as a menu of options. The call address for BASIC4.0 is 46610.

The article on p291, last issue, was not from an IEE circular, the item that was went astray at the printers. Pages 357 and 359 were transposed and the page number referred to in the editorial was 372.

Finally, the last term in the Karnaugh function (p349) should be 'D' and not its complement.

--o0o--

"THIS ADVERTISING SPACE RESERVED FOR RAM ELECTRONICS FLEET"

NEW VIC EXPANSION

Recently released from RAM Electronics of Fleet is a new super 32K RAM pack that, with the on-board capacity of the Vic's 3.5K gives a massive 35.5K+ bytes for use on your Vic. However, only 27K is directly useable by BASIC. The unit comes in an attractively styled custom injection moulded case, with three switches on the top, enabling it to be plugged directly or via RAM's 4-slot mother board.

The switches can enable combinations of 8K expansion blocks giving maximum flexibility when working with dedicated games, machine code or large programs.

Normal retail prices are:

32K RAM Pack	£ 69.95
16K RAM Pack	£ 44.95
4-slot Mother Board	£ 24.95

all prices include VAT. Contact the discounts officer for special ICPUG Member prices.

--oOo--

MEMBERS' PRIVATE SALES & WANTS

Our auditors, J.G.Feingold & Co. have for sale a CMC non-addressable interface PET to RS232 for £ 50.00 - further details phone 061-792 1257.

The editor has for disposal three 'Practical Computing' binders; offers welcome.

3000-series PET plus 3040 disk drive for sale, both upgradable, at £ 399 each. Regularly serviced with free diskettes and games. Contact R.D.Osner, Roxburghe International Ltd., 166, High St., Hounslow, Middx. Tel: 01-570 5783.

BASIC 2 Toolkit chip and manual for sale at £ 15; Disk-O-Pro Chip and manual at £ 25, Contact Laurie Faulkner, 136, Kingsway Road, Stoneygate, Leicester, LE5 5TT. Tel: 0533 704676.

--oOo--

COMAL CORNER

By Brian Grainger

The COMAL arena is changing again now after a quiet period. The previous version for BASIC4, 0.12, has been slightly modified. I have dubbed it revision 1 and have been distributing it as COMAL80-0.12/1. Anybody who wrote to me after my articles in the September Newsletter will have this version. Three differences from the earlier one have been noted. Firstly a bug where the error message remained on the screen even after error correction has been removed. Secondly, while the interpreter will accept a NEXT statement, (as in FOR...NEXT), it gets translated to ENDFOR which is the COMAL Kernal syntax. Using ENDFOR maintains the type of structure syntax which occurs with WHILE...ENDWHILE, CASE...ENDCASE. Finally another 500 bytes program space is made available, making 5612 in total. In conjunction with this BASIC4 update, the final released version of 1.02 arrived for the 8096. It is version 1.02J. Not having an 8096 I do not know what differences exist between 1.02A and 1.02J. These versions are, as far as I am aware, only available from me at present and not in the software library. We are in the process of bringing some order to the library chaos and until that is complete the best place to come for COMAL copies is myself.

I have found two further differences between COMAL version 0.11 and 0.12/1. The former rejected a shifted 'space' while entering programs. 0.12/1 will automatically convert this to an unshifted 'space'. This even occurs if the shifted 'space' occurs in a string constant, which can be a nuisance.

Another benefit of 0.12/1 is that more than one procedure can be called with a single COMAL statement. Previously only multiple arithmetic expressions were allowed in single statements. As an example the following statement is now valid:

```
IF NOT FINISHED THEN READ'SOME'MORE ; PROCESS
```

where READ'SOME'MORE and PROCESS are valid PROC names.

I have heard that a COMAL version 2 is to be made available, but as yet no details have appeared. I have a hunch it may be for the Commodore 500 or 700 series. I am, (at the beginning of December), eagerly awaiting a revised version of Borge Christensen's COMAL notes to see if this gives any clues. I have heard rumours of using disk space like virtual memory.

To finish this month's COMAL CORNER I want to give the current program status to aid the confused:

COMAL80IN, COMAL80EX is the BASIC4 split version at 0.11 standard. An 0.12 standard split version does not exist.

COMAL80-0.12/1 is the BASIC4 full version.

COMALB3 is the BASIC2 full version at 0.11 standard.

COMALB3IN, COMALB3EX is the BASIC2 split version at 0.11 standard.

COMAL4C1 is the version for users with cassettes only on BASIC4. It is aimed at giving an introduction to COMAL for these users. Very careful use is required so it is not suitable for teaching purposes. It is at 0.11 standard

COMAL3C1 is the version for users with cassettes only and BASIC2. Careful use is again required and it is at 0.11 standard.

No 0.12 standards for BASIC2 exist at present.

COMAL 80 R1.02J is the latest 8096 version.

--o0o--

8032 NOTES

By Martin Guy.

Both <SHIFT> keys and <2> puts you instantly into UPPER CASE/GRAPHICS mode. Both <SHIFT> keys and <2> and <CRT> gives SCROLL DOWN.

--o0o--

PROBLEMS & QUERIES

By Mike Todd.

I'm writing this only just as responses are coming in from the the previous problems page, but the signs are that people are rallying round. Some of these replies are worthy of printing in the Newsletter as complete articles and this I will endeavour to do next time.

It therefore strikes me that replies to the problems should continue to come via me so that I can pick out the details and give everyone the benefit of the expertise.

If you do have a query, please send it to me, and try to word it succinctly to save me having to condense a four-page letter into a paragraph!

I would suggest that the most common queries are likely to be along the lines of "I wonder if anyone has written a program to do ....." or "Does anyone know how to ....", and remember that there will inevitably be a delay before the query appears and replies are received.

Anyway, to this month's queries:

COMPUTHINK & IEEE

A reader writes (sounds a bit like Marj Proops!) "I have a 3032 with dual Computhink drives and an IEEE printer.

"I would now like to use a number of utilities like Visicalc, POWER etc., but these are not available for Computhink systems. Dealers offer little help, other than suggesting scrapping the Computhink drives and replacing with Commodore. However, I have a lot of software for the Computhink that I don't want to rewrite.

"What I propose is to add a single Commodore disk drive to the IEEE bus then, using a ROM pager, to switch out the Computhink ROM and patch in the CBM.

"Is this possible? Is there anyone who has such a multiple disk system with whom I could discuss the hardware modifications? Is there a better/cheaper alternative?"

EPSON PRINTERS MX80 & MX100

"I have an EPSON MX100 (identical to MX80 except faster and bigger) and have a couple of queries.

"(1) The auto line-feed and slashed zero features selectable on the DIP-switch inside are extremely awkward, especially as it requires the removal of the IEEE interface board everytime I want to change the switches. Does anyone know of a simple way of either extending the DIP switch to the outside world without voiding the warranty, or is there a software method using ESC sequence which EPSON don't publish?

"(2) Does anyone know exactly what the JPET/JNORM DIP switch on the IEEE board actually does? EPSON don't seem to know themselves!

"(3) Does anyone have a program which allows PET graphics characters to be sent to the printer using the bit image mode? I can see a way to do it, but haven't the time to write the program. In fact, are there any EPSON graphics programs around?

"(4) Does anyone have any technical data (mainly the instruction set) on the MPU used in the EPSON (a 8049). I've got a dump of the ROMs and would like to have a rummage around to see what's going on."

FINALLY

Finally, a query from me - does anyone know where I can get a Visible Music Monitor without the hardware. I would like the VMM but already have a very good DAC and don't want to have too buy another.

Well, that's it - I look forward to hearing from you.

--oOo--

REVIEW

COMAL HANDBOOK — Prentice/Hall,
LEN LINSAY — Hemel Hempstead.

This book on COMAL, the third to make an appearance in this country is undoubtedly what everybody has been waiting for. In a manner somewhat akin to Raeto West's treatment of BASIC, Len Lindsay has produced the definitive reference work for COMAL on the CBM.

The handbook starts with a quote from an anonymous Danish student- "I'm tired of BASIC but scared of Pascal"! This is followed by a brief introduction to COMAL by Borge Christensen.

The first few pages of the book proper are spent in outlining how the book is to be read and defining some common terms used throughout the book. These include disk file names, variable identifiers, the various versions of COMAL and a brief introduction to the editing facilities of the CBM machines.

The bulk of the text is the 160+ pages, each in a standard format, devoted to explaining each of the COMAL keywords. For each keyword a description of its purpose is given, the FULL syntax, some examples of use and a simple sample program using the keyword. In addition, the versions of COMAL to which it is applicable are also identified.

There are a number of appendices. The first describes the various COMAL structures in great depth. String handling, which is fundamentally different from Microsoft BASIC, is covered in the second. Another very useful appendix contains procedures for general use with COMAL programs. A number of these emulate functions in version 1.02 that are not available in 0.12 and therefore need a special procedure. The full COMAL Kernal definition is included in another appendix.

For those who are going to use COMAL regularly this book is a must. It is very comprehensive although some minor items are missing, most important being the

abbreviations one can use for some keywords. Although the book is biased towards the new COMAL versions 0.12, 1.02 some mention is made of 0.11 and 1.01 but the approach to these versions is not systematic. Nevertheless the book is essential to programmers using those COMAL versions as well.

My final observation is that the book is so comprehensive that sometimes one feels a small quick-reference is required to have ready access while programming. I'm working on that!

All in all this is an excellent REFERENCE book on COMAL and while not cheap (£ 15.15) represents very good value. By the time you read this it should be available from Prentice/Hall at 66, Wood Lane, Hemel Hempstead, Herts. (0442) 58531/212771.

B.D.G.

--o0o--

AVOIDING THE 'INPUT' CRASH

Those of you that are new to Microsoft BASIC will have found that when using the INPUT statement, pressing the <return> key on its own with no other input will cause a program to abort. Over the years various methods of avoiding this have been devised, each with various merits and disadvantages. Here is a little known method that will not abort with either a null input, or the <STOP> key:

```
100 N=16:REM N=3, 14 or 16
150 POKE N,1 : INPUT A : POKE N,0 : PRINT
```

The value of N is determined by which version of BASIC is in use (see Newsletter p199, July '82), the one shown is for BASIC4. Vic's BASIC has been revised to overcome the problem.

R.D.G.

--o0o--

MICROCHESS ON ICE - Part 2

By Walter Green

Note:- This article is intended to be read as a follow up to Part 1 on page 219 of Vol4 No.5. It only deals with the old ROM Microchess however.

In order to cope with the various Microchess versions, the previous article put the subroutine for 'Return to BASIC' at $1800 and the Duplicate chess piece set at $182D.

With the old-ROM Microchess put the subroutine at $17F0 and modify $05B9 and $05BA to $F0 and $17 respectively. The duplicate chess piece set should start at $181D. When the duplicate set is at this location POKE1188 with 20 or 24 will give the option of either set of chess pieces.

LOAD but do not RUN the modified Microchess. Enter in Direct Mode FORI=6173TO6206 : POKEI,187:NEXT. Now RUN and you will see an empty chessboard. FORI=5149TO5182 will do the same with the cassette as bought.

To set up a problem involving few pieces carry out the above and POKE the pieces; i.e. POKE5149,3 will put the White King on E1. ANY GAME AGAINST PET OR PLAYER CAN BE SAVED AND ONE CAN RETURN TO ANY POSITION TO PLAY ANOTHER MOVE. SEVERAL POSITIONS CAN BE STORED AND CALLED AT WILL. In order to do this I have pinched the following:

'*', 'X', 'N' and 'P'

Modify the text to give:-

```
05C0: 4C 50 18 ='*'= Memorise position v. PET
04ED: 4C 60 18 ='X'= Restore position v. PET
0549: 4C 70 18 ='N'= Memorise position v. opponent
0520: 4C C0 18 ='P'= Restore position v. opponent
```

The subroutine given as at $1850 in the previous article should be moved to an unused place in memory and $055D modified accordingly. Then 1850 can be replaced with:

```
1850: A2 21 B5 90 9D 90 18 CA
1858: 10 F8 4C D3 04 ...
```

Playing against PET, should you wish to return to the present position, simply type '*'. This will appear on the screen but will vanish as you press 'return'. Nothing more will happen and you should carry on playing. The subroutine copies the position into $1890. The same subroutine in reverse is entered at $1860. To return to this position, type 'X' AND MAKE YOUR NEXT MOVE. Your move will be made followed by the stored position. To illustrate this make a copy with SYS2062 and LOAD the copy. Now FORI=6173TO6206 : POKEI,187 : NEXT and RUN. The empty board will be displayed. Now make your move. The piece will appear out of the blue and move, followed by the rest of the board !

1870: A2 21 BD 1D 18 9D DD 18
1878: CA 10 F7 4C D3 04 ...

This puts a copy of the duplicate chess set at $18DD when you type 'N'. POKE a problem/end game and store it there. The same subroutine in reverse placed at $18C0 will restore it when 'P' is typed. Save with SYS2062 and with the copy loaded and RUN you have the choice of three positions.

Anyone with an old-ROM 8K PET would be advised to get a TIM Monitor relocated to $7183. This will make page $1B available for storing positions. The jump address in $1820/1 will then read 4C FE 1D.

When moves are POKEd, captured men must be removed by POKEing the value 187. A piece retains its original identity. Combining these two facts:-

WHITE moves- POKE6189,51 (=E2E4)
BLACK moves- POKE6205,68 (=D7D5)
WHITE moves- POKE6189,68:POKE6205,187

Note: After typing 'P' one must return to BASIC with '-' and then RUN to restore the memorised position v. opponent.

Brian Grainger Note! Since this article was received for the Newsletter further enhancements to Microchess have been made to enable, for example, PET to play by itself with no human intervention. Another enhancement is replay of a stored game. The author, at 151, The Hatherley, Basildon, Essex will be able to expand on these improvements should you be interested.

--o0o--

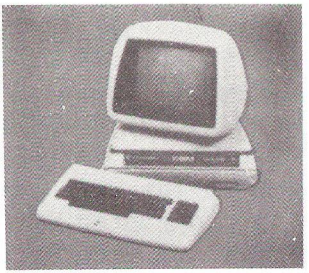
MULTIPLE KEY PRESS DETECTION

By Brian Grainger

I do not claim any originality for the ideas of this article. I have not seen anything on it in the ICPUG Newsletter so I thought I would bring it to the members attention.

The memory location $9B always holds the same contents as $E812. This location is the keyboard output value which indicates the column of the keyboard matrix in which a depressed key is located. Now the keyboard is scanned per group of 7 or 8 keys but the scan routine finishes with the scan of the bottom row of the matrix. Thus one can PEEK(155) to identify which keys of this 'bottom' row are pressed. It will not only detect a single key press but any combination of the bottom row keys. Check this table:

0 RVS
1 [
2 SPACE
3 <
4 RUN (if STOP key disabled)
5 detects no keys
6 .
7 =



For each key pressed check tne associated number from the above table and raise two to the power of that number. Do this for each key and sum the results. If you take away the final result from 256, that will be the value of PEEK(155). For example if the RVS and SPACE key are pressed simultaneously we have

2↑0+2↑2 = 5

This value from 256 gives 251 which will be the value of PEEK(155).

What uses does this have. Well sometimes in games it is useful to press more than one key at once and do different things dependent on the combination of keys pressed. One cannot use a GET command for this but if the right keys are chosen one could use PEEK(155).

To finish here is an odd 1-liner. Input the line, clear the screen, do a few cursor downs and RUN. The result on my 3032 upgraded PET is quite extraordinary. Presumably it is a timing problem:

```
10 PRINT"<home><4 spaces><home>";PEEK(155):GOTO10
```

--o0o--

SOME IEEE OBSERVATIONS

By Simon Tranmer

The following observations may save you the couple of hours it took me to find out what was happening. I have an 8032 normally connected to a 4040 and an 4022. On many occasions when using the machines, I know that I will not need the printer and therefore used not to bother to switch it on, however the IEEE cable was still connected. In these circumstances there is not usually any problem. You can load and save files and look at the directory of your disks with impunity. However if you try to do anything involving long disk access (e.g. assembling large programs) some very odd things start happening. You may not be able to find programs on the disk that you know are there, the files may read in corrupted when they are ok on the disk or any one of a number of errors occurs. It all seems to be related to the bus loading. Switch the printer back on and everything is ok.

One day when my printer was off by mistake I could not read any disk directory, and data transfer when it occurred was about 50 times slower than usual. I have seen the same thing happen with two disk drives connected and one switched off and also two PETs connected to one disk drive with one PET switched off. The moral of the story is, if you have a device on the bus leave it switched on, even if not in use, otherwise it may look as though your disk drives have died...

--o0o--

REVIEW - BUTI
BASIC UTILITY CARTRIDGE

Richard Allen.

The BUTI (pronounced "beauty") programmers utility ROM is available from AUDIOGENICS and it comes with 3K of RAM expansion adding 17 new commands to Vic's BASIC.

Once plugged in and the Vic switched on, the toolkit is automatically activated, although you can KILL the ROM and there is a SYS command to reactivate it again.

Commands include automatic line numbering and renumbering. There is an APPEND command to allow a program to to be added to the end of one in memory although this will not merge them, and DELETE takes out any lines you don't want in the program.

Typing DUMP after stopping a program, will display the variables used in the program. It's also possible to specify which type of variable you want to display. FIND enables you to find where a command or instruction occurs in your program and EDIT provides the facility to change commands or instructions.

There is a HELP command which clears the screen and shows a list of the new commands available. This is unlike other toolkit HELP commands which shows up the approximate area of a program line where an error has occurred. With BUTI, when an error message comes up, the line in question is automatically displayed.

STEP allows a program to be executed line by line and TRACE executes the program slowly. In both cases there is the option of having a reverse field window on the right displaying line numbers as they happen. And there is a hex to decimal and binary and a decimal to hex and binary converter.

When you type NEW on the Vic, you are not actually erasing the previous one completely. Instead you just reset

the memory pointers. Provided that you have not typed any further instructions on the Vic, by using the UNNEW command you can bring an old program that has just been NEW'd back into memory. If changes have been made since the NEW command then you'll just get garbage.

When you expand your Vic's memory, the locations alter and you have to make changes in your programs to enable them to function correctly. By using the VIC command you can configure the Vic to become a standard machine again. VIC3 expands it to the 3K memory limit and VIC8 raises it to the maximum memory you have available.

Most of the commands can be abbreviated. The manual suggest using just the first three letters, but I have found that they can be abbreviated in the same way as the normal Vic commands. That is, by typing the first letter (or in some cases the first two) and then, with shift pressed, typing the second (or third).

After using the BUTI, I have found it extremely useful. Unlike Commodore's utility cartridge, the BUTI does add to the Vic, though you are not able to use the function keys and there are some differences in the toolkits. I would not hesitate to recommend BUTI.

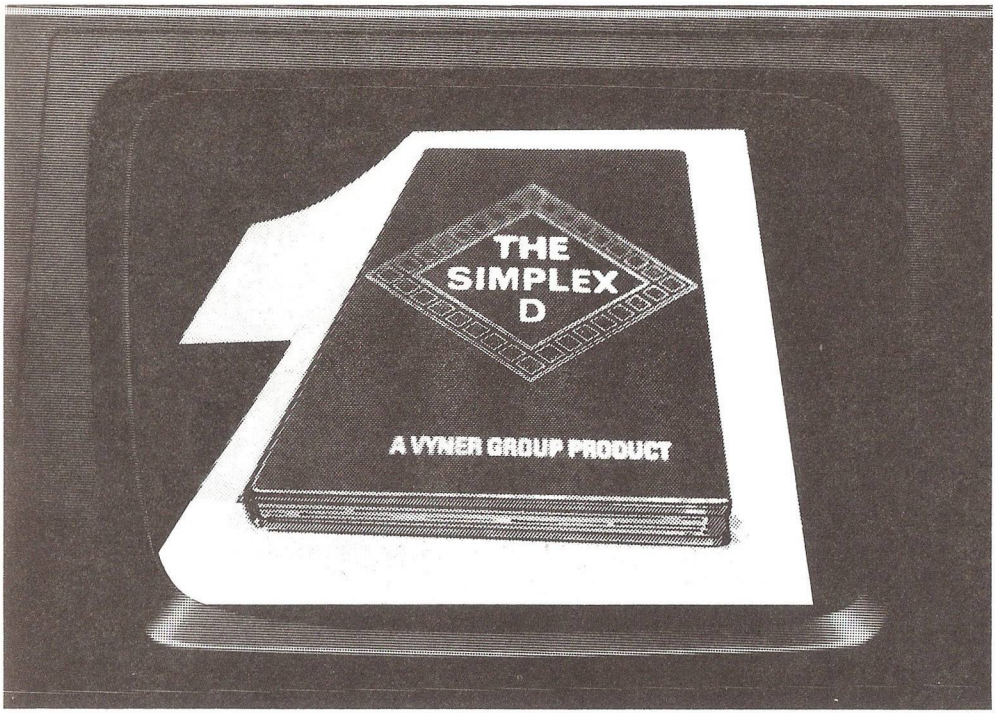
--oOo--

COMMENT: The BUTI costs £39.99 and now comes with a machine code monitor. My own feeling is that the Commodore Programmer's Aid provides very much the same commands (with the exception of UNNEW, VIC and hex-decimal-binary converter). It doesn't allow use to be made of the function keys, and the many extra commands in the Commodore Aid are very useful.

BUTI does at least come with 3K of RAM and a machine code monitor, so it may be considered to be better value.

M.R.T.

--oOo--



The Electronic Cash Book

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

Unique features:

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

Other features include . . .

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

MICRO-SIMPLEX

MICRO-SIMPLEX

OFFICIALLY APPROVED BY COMMODORE

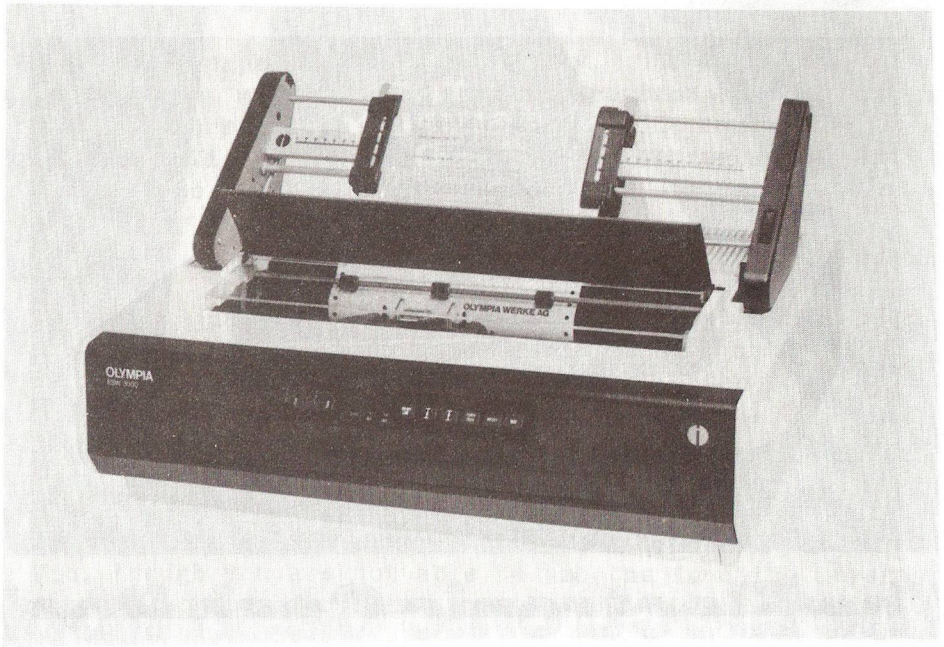
commodore COMPUTER

Contact Any Commodore Dealer or

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

Tel: 0625 615000

INTELLIGENCE REPORT



THE NEW OLYMPIA ESW 3000 HIGH SPEED DAISYWHEEL PRINTER.

Trade Distribution and Sales:
Intelligent Interfaces Ltd.,
P.O.Box 30,
Stratford on Avon,
Warwickshire.CV37 7BH.
Telephone:0789 295385.

CLUB SOFTWARE REVIEW - DDS SORT

By Brian Grainger

For this month's club software review I turn my attention back to Hertfordshire and Mick Bignell. He has made available for the public domain a program which sorts the disk directory into alphabetical order. It is extremely quick and has a few good features thrown in.

After loading DDS SORT you run it and it will tell you to put the disk requiring directory sorting in drive 0 and a spare disk for a scratch file to be placed in drive 1. Having done this and pressed a key the process starts. An option to keep the first program on the disk as the first program irrespective of name is given. This enables programs that are run by the RUN key to maintain their priority. One can also subdivide the directory sort so that files of the same type appear together. After having answered the two option questions the work is done in a flash and you have a nicely sorted directory.

This program works on BASIC2 or 4 and 3040 disks upgraded to DOS2, 4040 disks or 8050 disks. A super utility which is very handy.

--o0o--

4022 PRINTER - A SEQUEL

By Simon Tranmer

Both Dave Prentice and myself have changed the ROM in our 4022 printers as suggested on p171, July, '82 issue. The new ROM makes the printer bi-directional and the time saving on long listings is quite considerable. I would recommend the upgrade to all 4022 users. The part number of the new ROM for those interested is 901631-02 or 325301-01. Your printer must apparently have a 8Mhz crystal fitted for these ROMs to work.

--o0o--

SUPERSPELL - Simon Tranmer's new spelling checker

By Barry Biddles.

New to the market is Simon Tranmer's program, Superspell. Superspell is not a spelling corrector, but a spelling CHECKER. The difference is several hundred pounds and a lot of code! A word may be correct in one part of the text, but wrong in another, or its spelling may depend upon one of the many little rules of the english language which must terrify foreign school children. What Superspell does is to look up each word of the textfile (local or global) in its 31,000-word dictionary, and report words not found, so that the user can decide what to do. He may correct the word, using screen editing, and has the option of storing the result in a 'user dictionary'. This dictionary may easily be edited or corrected, and may finally be merged with the main dictionary.

Superspell also reports the number of words used, of great use to authors, and lists the different words used, with the number of occurrences of each. In addition, it supplies a paragraph count and a sentence count. Naturally, Superspell is a machine code program, since extreme speed is required for such a program. Even having said that, however, Superspell is exemplary among spelling checkers, in its speed of operation, handling a maximum-size Superscript file of about 2,000 words in about 90 seconds, and in the size of its dictionary, 31,000 words stored in only 85Kbytes! (That speaks of some rather clever coding. Perhaps Simon can be persuaded to tell us all about it some day?).

Superspell is to be made available on a 'dongled' disk which also contains Superscript, for the PET (also available as 'Easyspell' for the C-64). Interested non-members should contact Precision Software. Members should contact John Bickerstaff for details of the special prices arranged through ICPUG for members. John's address is 48, Martin Down Road, Whitstable, Kent. CT5 4PR. Tel: (0227) 272702. Precision Softw are are at 4, Park Terrace, Worcester Park, Surrey, KT4 7JZ. Tel: 01-330 7166.

--oOo--

Well I still think we should write to "Jim'll Fix It"...

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